Apéndice A. Archivos configuración Sistema de Análisis de Errores en Programas

En este apéndice incluimos varios archivos de configuración utilizados por el Sistema de Análisis de Errores en Programas.

A.1 Archivo de configuración unidad de compilación simple

El Sistema de análisis de errores en programas (descrito en el Capítulo 7. Sistema de análisis de errores de programas: PBA) utiliza determinados archivos para configurar Unidades de Compilación. Una Unidad de Compilación, permite indicar al sistema los directorios donde se encuentran los archivos fuente y donde tiene que dejar los resultados; las herramientas que debe utilizar, las opciones de estas herramientas y en que orden debe utilizarlas; incluso permite condicionar el uso de determinadas herramientas a que la ejecución de otras proporcione determinados resultados. El formato de estos archivos es XML del tipo de los archivos build.xml de la herramienta Apache Ant [Ant 2003].

En este apartado incluimos un archivo que ejecuta el compilador estándar de Java, concretamente el incluido en el JSDK 1.4.2 y una aplicación que se encarga de analizar los resultados y guardarlos en la base de datos del Sistema de análisis de errores en programas.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="miproyecto" default="compilar" basedir=".">
    <description>
        Nombre     : Sistema de compilacion Java
        Versión    : 1.0
       Parametros : -Druta="ruta del proyecto"
                    -Dusuario="usuario que lanza la compilacion"
                    -Dcompilacion="codigo de la compilacion"
    </description>

    <!-- Propiedades generales del sistema -->
    <property file="path.properties"/>
    <!-- Directorio temporal -->
    <property name="tmp" location="${usuarios}/${usuario}/tmp"/>
    <!-- Directorio donde se guarda informacion temporal en esta compilacion -->
    <property name="tmp.log" location="${tmp}/log${compilacion}"/>
    <!-- Directorio donde se guardan las fuentes del proyecto -->
    <property name="ruta.src" location="${ruta}/src"/>

    <!--Definiciones de las ampliaciones que vamos a usar-->
    <taskdef name="myjavac" classname="proyecto.utilidad.ant.JavacTask"/>
    <taskdef name="analizador"
        classname="proyecto.utilidad.ant.AnalizadorTask"/>
```

```
    <target name="inicio" description="Inicializaciones necesarias">
        <!-- Creamos el directorio temporal -->
       <mkdir dir="${tmp}"/>
       <mkdir dir="${tmp.log}"/>
    </target>

    <target name="compilar" depends="inicio"
        description="Compilamos los ficheros fuente">
        <myjavac output="${tmp.log}/javac.log" debug="true"
            home="${javac.home}">
            <src path="${ruta.src}"/>
            <include name="**/*.java"/>
        </myjavac>
        <analizador herramienta="javac-sdk-1.4.2"
                    fichero="${tmp.log}/javac.log"
                    filtro="false"
                    usuario="${usuario}"
                    compilacion="${compilacion}"
                    property="errores"/>
    </target>

</project>
```

A.2 Archivo unidad de compilación completa

Se pueden crear tantas unidades de compilación como se crean convenientes y cada una tiene un identificador único con lo que se pueden utilizar como bloque que unifica todo el trabajo que debe realizar el sistema sobre el conjunto de archivos de un proyecto.

En este apartado incluimos un archivo que ejecuta el compilador estándar de Java y varias herramientas de análisis estático de errores. Las herramientas utilizadas se describen en el apartado (Apartado 7.6. Herramientas de búsqueda de errores utilizadas en el proyecto). Una vez que las herramientas han terminado de realizar su tarea, una aplicación que se encarga de analizar los resultados y guardarlos en la base de datos del Sistema de análisis de errores en programas.

Esta Unidad de Compilación es la que se ha utilizado para analizar y capturar los avisos del (Capítulo 10. Clasificación de usuarios basada en la detección de errores).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="completa" default="busqueda_errores" basedir=".">
    <description>
        Nombre     : Sistema de compilacion Java
        Versión    : 1.0
       Parametros : -Druta="ruta del proyecto"
                    -Dusuario="usuario que lanza la compilacion"
                    -Dcompilacion="codigo de la compilacion"
    </description>

    <!-- Propiedades generales del sistema -->
    <property file="path.properties"/>
    <!-- Directorio temporal -->
    <property name="tmp" location="${usuarios}/${usuario}/tmp"/>
    <!-- Directorio donde se guarda informacion temporal en esta compilacion -->
    <property name="tmp.log" location="${tmp}/log${compilacion}"/>
    <!-- Directorio donde se guardan las fuentes del proyecto -->
    <property name="ruta.src" location="${ruta}/src"/>

    <!--Definiciones de las ampliaciones que vamos a usar-->
    <taskdef resource="net/sf/antcontrib/antcontrib.properties"/>
    <taskdef name="findbugs"
        classname="edu.umd.cs.findbugs.anttask.FindBugsTask"/>
    <taskdef name="myjavac" classname="proyecto.utilidad.ant.JavacTask"/>
    <taskdef name="analizador"
        classname="proyecto.utilidad.ant.AnalizadorTask"/>
    <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>
```

```
    <taskdef name="antic" classname="proyecto.utilidad.ant.AnticTask"/>
    <taskdef name="jlint" classname="proyecto.utilidad.ant.JlintTask"/>

    <target name="inicio" description="Inicializaciones necesarias">
        <!-- Creamos el directorio temporal -->
       <mkdir dir="${tmp}"/>
       <mkdir dir="${tmp.log}"/>
    </target>

    <target name="compilar" depends="inicio"
        description="Compilamos los ficheros fuente">
        <echo message="" file="${tmp.log}/javac.log"/>
        <myjavac output="${tmp.log}/javac.log" debug="true"
            home="${javac.home}">
            <src path="${ruta.src}"/>
            <include name="**/*.java"/>
        </myjavac>
        <analizador herramienta="javac-sdk-1.4.2"
                    fichero="${tmp.log}/javac.log"
                    filtro="false"
                    usuario="${usuario}"
                    compilacion="${compilacion}"
                    failOnError="true"
                    property="errores"/>
    </target>

    <target name="busqueda_errores" depends="inicio,compilar"
        description="Buscamos en el cógigo errores potenciales">
    <if>
        <equals arg1="${errores}" arg2="0" casesensitive="false"/>
        <then>
            <parallel>
                <sequential>
                    <echo message="" file="${tmp.log}/jlint.log"/>
                    <jlint home="${jlint.home}" output="${tmp.log}/jlint.log"
                           omite="synchronization" failOnError="true">
                        <src path="${ruta.src}"/>
                        <include name="**/*.class"/>
                    </jlint>
                    <analizador herramienta="jlint"
                                fichero="${tmp.log}/jlint.log"
                                usuario="${usuario}"
                                compilacion="${compilacion}"/>
                </sequential>
            </parallel>
            <parallel>
                <sequential>
                    <echo message="" file="${tmp.log}/antic.log"/>
                    <antic home="${antic.home}" output="${tmp.log}/antic.log"
                        java= "true" failOnError="true">
                        <src path="${ruta.src}"/>
                        <include name="**/*.java"/>
                    </antic>
                    <analizador herramienta="antic"
                                fichero="${tmp.log}/antic.log"
                                usuario="${usuario}"
                                compilacion="${compilacion}"/>
                </sequential>
            </parallel>
            <parallel>
                <sequential>
                    <echo message="" file="${tmp.log}/findbugs.log"/>
                    <findbugs home="${findbugs.home}" output="emacs"
                        outputfile="${tmp.log}/findbugs.log"

omitVisitors="DontCatchIllegalMonitorStateException,FindInconsistentSync2,FindJSR
166LockMonitorenter,FindMismatchedWaitOrNotify,FindNakedNotify,FindReturnRef,Find
RunInvocations,FindSpinLoop,FindTwoLockWait,FindUnconditionalWait,FindUnreleasedL
ock,FindUnsyncGet,InitializationChain,LazyInit,LockedFields,MutableLock,MutableSt
aticFields,SerializableIdiom,StartInConstructor,SwitchFallthrough,WaitInLoop,Find
InconsistentSync2"
                        timeout="60000">
                        <sourcepath path="${ruta.src}"/>
                        <class location="${ruta.src}"/>
                    </findbugs>
                    <analizador herramienta="findbugs-0.7.3"
                                fichero="${tmp.log}/findbugs.log"
```

```
                                usuario="${usuario}"
                                compilacion="${compilacion}"/>
                </sequential>
            </parallel>
            <parallel>
                <sequential>
                    <echo message="" file="${tmp.log}/pmd.log"/>
                    <pmd
rulesetfiles="${pmd.home}\rulesets\reglas_pmd_predefinidas.xml">
                        <formatter toFile="${tmp.log}/pmd.log"
                            type="net.sourceforge.pmd.renderers.EmacsRenderer"/>
                        <fileset dir="${ruta.src}">
                            <include name="**/*.java"/>
                        </fileset>
                    </pmd>
                    <analizador herramienta="pmd-1.6"
                                fichero="${tmp.log}/pmd.log"
                                usuario="${usuario}"
                                compilacion="${compilacion}"/>
                </sequential>
            </parallel>
        </then>
        <else>
            <echo message="Numero de errores de compilacion = ${errores}"/>
        </else>
    </if>
    </target>

</project>
```

A.3 Esquema XML para validación de los archivos de configuración de estadísticas

Este archivo permite validar cualquier archivo de configuración de estadísticas mediante cualquier parser XML que soporte XML-Schema. En el apartado 7.7 se describen todas las posibilidades que permite este XML-Schema.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
    Document   : estadisticas.xsd
    Description:
        XML Schema de los ficheros de estadísticas.
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Definición del elemento "estadisticas" -->
<!-- Puede tener anidados uno o más elementos que representan tipos
     de estadisticas. Cada uno de estos tipos de estadísticas están
     representados por un elemento distinto.
     Para el proyecto se concivieron dos tipos de estadísticas distintas:
        - Errores más frecuentes.
        - Media de errores.
     Pero pueden ser ampliables en un futuro
-->
    <xs:element name="estadisticas">
        <xs:complexType>
            <xs:choice maxOccurs="unbounded">
                <xs:element ref="ErroresFrecuentes"/>
                <xs:element ref="MediaErrores"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>

<!-- Definición de los tipos de estadísticas -->
<!-- Definición de la estadística de "Errores más frecuentes".
     Esta posee un atributo "numero" que representa el número de errores
     más frecuentes que se desean mostrar, por defecto es 5.
     Su otro atributo "manejador" indica la clase que maneja esta estadistica.
```

```
     El valor por defecto es "proyecto.estadisticas.ErroresFrecuentes", pero
     puede ser cambiada por los usuarios.
-->
    <xs:element name="ErroresFrecuentes">
        <xs:complexType>
            <xs:group ref="orden_estadistica"/>
            <xs:attribute ref="numero"/>
            <xs:attribute ref="manejador"
                default="proyecto.estadisticas.ErroresFrecuentes"/>
        </xs:complexType>
    </xs:element>

    <xs:attribute name="numero" default="5">
        <xs:simpleType>
            <xs:restriction base="xs:positiveInteger">
                <xs:maxInclusive value="10"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>

<!-- Definición de la estadística de "Media de Errores".
     Esta posee un atributo "evolucion" que representa el número de periodos
     para los que se desea conocer este valor, por defecto es 1.
     Su otro atributo "manejador" indica la clase que maneja esta estadistica.
     El valor por defecto es "proyecto.estadisticas.MediaErrores", pero puede
     ser cambiada por los usuarios.
-->
    <xs:element name="MediaErrores">
        <xs:complexType>
            <xs:group ref="orden_estadistica"/>
            <xs:attribute ref="evolucion"/>
            <xs:attribute ref="manejador"
                default="proyecto.estadisticas.MediaErrores"/>
        </xs:complexType>
    </xs:element>

    <xs:attribute name="evolucion" type="xs:positiveInteger" default="1"/>

<!-- El atributo "manejador" tiene como misión indicar cual es el manejador
     de las estadisticas.
-->
    <xs:attribute name="manejador" type="xs:string"/>

<!-- Todos los tipos de estadisticas tendrán una estructura base:
        - Periodo       (una aparición)
        - Usuarios      (una aparición)
        - Errores       (una o más apariciones)
        - Compilaciones (0 o 1 apariciones)
-->
    <xs:group name="orden_estadistica">
        <xs:sequence>
            <xs:element ref="periodo"/>
            <xs:element ref="usuarios"/>
            <xs:element ref="errores" maxOccurs="unbounded"/>
            <xs:element ref="compilaciones" minOccurs="0"/>
        </xs:sequence>
    </xs:group>

<!-- Definición de los periodos -->
<!-- Habrá tres formas posibles de indicar los periodos:
        - Mediante un intervalo temporal relativo (unidad dia).
            El final es hoy menos los dias que se indiquen como retardo.
        - Mediante un intervalo temporal absoluto (unidad dia).
            El final es la fecha indicada. El formato de la fecha es:
                AAAA-MM-DD (año - mes - día)
        - Mediante un intervalo de compilaciones absoluto (unidad compilacion).
            El final es la compilación indicada.
     Siempre habrá que especificar la duración del periodo, cuya unidad será la
     que se use en el inicio.
     El inicio de un periodo de final "f" y de duración "d" será "f-d"
-->
    <xs:element name="periodo">
        <xs:complexType>
            <xs:sequence>
                <xs:choice>
                    <xs:element ref="tiempo"/>
                    <xs:element ref="compilacion"/>
```

```
                </xs:choice>
                <xs:element ref="duracion"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="tiempo">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="relativo"/>
                <xs:element ref="fecha"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>

    <xs:element name="compilacion" type="xs:nonNegativeInteger"/>
    <xs:element name="duracion" type="xs:nonNegativeInteger" default="1"/>

    <xs:element name="relativo" type="xs:nonNegativeInteger" default="0"/>
    <xs:element name="fecha" type="xs:date"/>

<!-- Definicion del conjunto de usuarios -->
<!-- El elemento usuarios sirve para indicar los usuarios para los cuales se
     va a realizar el estudio estadistico.
     La forma de inclurlos es indicando dentro de elemento anidado usuario
     el código de usuario que se desee.
     El elemento usuarios tiene dos atributos booleanos:
        - "autor". Se usa para indicar que se haga la estadística para
                   el usuario que lanza el estudio estadístico.
                   Por defecto es "true".
        - "todos". Se usa para indicar que se haga la estadítica a todos los
                   usuarios del sistema.
                   Por defecto es "false".
-->
    <xs:element name="usuarios">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="usuario" maxOccurs="unbounded" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute ref="autor"/>
            <xs:attribute ref="todos"/>
        </xs:complexType>
    </xs:element>

    <xs:element name="usuario" type="xs:string"/>

    <xs:attribute name="autor" type="xs:boolean" default="true"/>
    <xs:attribute name="todos" type="xs:boolean" default="false"/>

<!-- Definición de los conjuntos de errores -->
<!-- Se puede indicar que errores se quere usar en la muestra de estadísticas.
     En primer lugar se pueden meter condiciones de dos tipos:
        - elementos "incluir" - usar los errores que cumplan una condición.
        - elementos "excluir" - usar los errores que no cumplan una condición.
     Si es expecifica más de una condición se deberán indicar como se relacionan
     esas condiciones, para ello se emplean los elementos "and" y "or".
     Las condiciones pueden ser de tres tipos:
        - "codigo".      Donde se indica el código del error.
        - "tipo".        Donde se indica un tipo de errores.
        - "herramienta". Donde se indica la herramienta que generó los errores.
-->
    <xs:element name="errores">
        <xs:complexType>
            <xs:group ref="condiciones_error" minOccurs="0"/>
        </xs:complexType>
    </xs:element>

    <xs:group name="condiciones_error">
        <xs:sequence>
            <xs:choice>
                <xs:element ref="incluir"/>
                <xs:element ref="excluir"/>
            </xs:choice>
            <xs:group ref="mas_condiciones" minOccurs="0"
               maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
```

```
    <xs:group name="mas_condiciones">
        <xs:sequence>
            <xs:choice>
                <xs:element name="and"/>
                <xs:element name="or"/>
            </xs:choice>
            <xs:choice>
                <xs:element ref="incluir"/>
                <xs:element ref="excluir"/>
            </xs:choice>
        </xs:sequence>
    </xs:group>

    <xs:element name="incluir">
        <xs:complexType>
            <xs:group ref="tipos_error"/>
        </xs:complexType>
    </xs:element>

    <xs:element name="excluir">
        <xs:complexType>
            <xs:group ref="tipos_error"/>
        </xs:complexType>
    </xs:element>

    <xs:group name="tipos_error">
        <xs:choice>
            <xs:element ref="codigo"/>
            <xs:element ref="tipo"/>
            <xs:element ref="herramienta"/>
        </xs:choice>
    </xs:group>

    <xs:element name="codigo" type="xs:integer"/>
    <xs:element name="tipo" type="xs:string"/>
    <xs:element name="herramienta" type="xs:string"/>

<!-- Definición de los conjuntos de compilaciones -->
<!-- Se puede indicar que unidades de compilación se quieren usar a la hora de
     hacer las estadísticas.
-->
    <xs:element name="compilaciones">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="unidad_compilacion" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="unidad_compilacion" type="xs:string"/>

</xs:schema>
```

A.4 Archivo ejemplo de configuración de estadísticas

Este archivo permite configurar la información incluida en un informe de resultados de análisis (Ver apartado 7.7. Estadísticas).

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
    Document   : ejemplo.xml
    Description:
        Ejemplo de como hacer los ficheros de estadísticas
-->

<estadisticas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:noNamespaceSchemaLocation="estadisticas.xsd">
```

```
    <ErroresFrecuentes numero="4">
        <periodo>
            <tiempo>
                <relativo>5</relativo>
            </tiempo>
            <duracion>14</duracion>
        </periodo>
        <usuarios todos="true"/>
        <errores>
            <excluir>
                <tipo>error de compilación</tipo>
            </excluir>
        </errores>
        <compilaciones>
            <unidad_compilacion>completa</unidad_compilacion>
        </compilaciones>
    </ErroresFrecuentes>

    <MediaErrores evolucion="5">
        <periodo>
            <compilacion>1000</compilacion>
            <duracion>200</duracion>
        </periodo>
        <usuarios autor="false" todos="true"/>
        <errores>
            <incluir>
                <herramienta>pmd-1.6</herramienta>
            </incluir>
        </errores>
        <errores/>
    </MediaErrores>

    <MediaErrores>
        <periodo>
            <tiempo>
                <fecha>2004-09-17</fecha>
            </tiempo>
            <duracion>7</duracion>
        </periodo>
        <usuarios todos="true"/>
        <errores>
            <excluir>
                <herramienta>jlint</herramienta>
            </excluir>
        </errores>
        <errores>
            <incluir>
                <herramienta>jlint</herramienta>
            </incluir>
            <or/>
            <incluir>
                <tipo>aviso de sincronización</tipo>
            </incluir>
        </errores>
    </MediaErrores>

    <ErroresFrecuentes numero="8">
        <periodo>
            <compilacion>10</compilacion>
            <duracion>10</duracion>
        </periodo>
        <usuarios/>
        <errores/>
    </ErroresFrecuentes>

    <ErroresFrecuentes numero="5">
        <periodo>
            <compilacion>10</compilacion>
            <duracion>10</duracion>
        </periodo>
        <usuarios/>
        <errores>
            <incluir>
                <codigo>13019</codigo>
            </incluir>
            <or/>
            <incluir>
```

```
                <codigo>13000</codigo>
            </incluir>
        </errores>
    </ErroresFrecuentes>
</estadisticas>
```

Apéndice B. Información sobre los errores frecuentes

En este capítulo se incluye información detallada sobre los errores más relevantes encontrados en el Capítulo 10 de los errores más frecuentes en los distintos cursos de la titulación de Ingeniería en Informática.

A.1 Errores que aparecen en todos los cursos

Evitar literales duplicados (13080)

Tipo de aviso:

Herramienta que detecta este aviso: PMD

El código que contiene literales String duplicados puede mejorarse declarando un String como campo constante.

Ejemplo:

The same String literal appears 8 times in this file; the first occurrence is on line 38

The same String literal appears 8 times in this file; the first occurrence is on line 41

```
…
public void escribir(int a[])
{
       escribir("{");
      for(int i=0; i<a.length; i++)
       escribir(a[i] + (i!=a.length-1 ? ", " : ""));
      escribir("}");
}

public void escribir(char a[])
{
      escribir("{");
      for(int i=0; i<a.length; i++)
       escribir(a[i] + (i!=a.length-1 ? ", " : ""));
      escribir("}");
}

public void escribir(long a[])
{
      escribir("{");
      for(int i=0; i<a.length; i++)
          escribir(a[i] + (i!=a.length-1 ? ", " : ""));
      escribir("}");
}
…
```

Evitar variables locales que no se utilizan (13084)

Tipo de aviso: Aviso de código no usado. Aviso sobre zonas de código no usadas en el programa.

Herramienta que detecta este aviso: PMD

Quitan claridad a la hora de leer el código y dificultan su mantenimiento.

Ejemplo:

```
Avoid unused local variables such as 'juegoRol'
public Principal() {
       realizarPruebas();
       Juego juegoRol= new Juego();
}
```

Ejemplo2:

Avoid unused local variables such as 'destreza'

Avoid unused local variables such as 'fuerza'

Avoid unused local variables such as 'magia'

Avoid unused local variables such as 'tipo'

Avoid unused local variables such as 'ataque'

```
public void arma(int i) {
       int opc, opcArma, fuerza, destreza, magia, n;
       String tipo;
      boolean ataque;
      p.escribirnl("\tTienes "+jugador.getnArmas()+" armas");
      p.escribirnl("\t\tPulsa 1 si quieres recogerla u otro NUMERO para
dejarla");
      opc = t.leerEntero();
      if(opc == 1) {
       if(jugador.getnArmas() >= 3) {
                        p.escribirnl("Tienes que dejar un arma para coger la
nueva");
                        n = mostrarArmas('x');
                        do {
                               p.escribirnl("\nEscribe el numero de arma que
deseas borrar:");
                               opcArma = t.leerEntero();
                        }while((opcArma > n)||(opcArma <= 0));
                        jugador.borrar(opcArma-1);
       }
       jugador.push((Arma)aventura.getSuceso(i));
      }
}
```

Evitar comparaciones innecesarias de expresiones booleanas (13039)

Tipo de aviso: Aviso de código innecesario. Aviso sobre zonas del código que son innecesarias o redundantes.

Herramienta que detecta este aviso: PMD

Se debería evitar el uso de operaciones de igualdad con operandos booleanos. Nunca se deberían utilizar los literales true o false en cláusulas condicionales. Esto permite mejorar la legibilidad del programa simplificando la expresión.

Ejemplo:

Avoid unnecessary comparisons in boolean expressions

```
    boolean insertado = false;
    for(int i = 0;i < 3; i++){
      if (inventario[i] == null && insertado == false){
          inventario[i] = objeto;
          insertado = true;
        }
    }
```

Evitar los bloques "catch" vacíos (13000)

Tipo de aviso: Aviso de estructura de control. Aviso de un posible error en una estructura de control de flujo del programa.

Herramienta que detecta este aviso: PMD

Ejemplo:

Avoid empty catch blocks

```
public void cerrarFlujo(){
    try{
      ficheroErrores.close();
    }catch(IOException e){}
  }
```

A.2 Errores que aparecen exclusivamente en un curso

Una sentencia vacía (punto y coma) no constituye el cuerpo de un bucle (13016)

Tipo de aviso: Aviso de código innecesario. Aviso sobre zonas del código que son innecesarias o redundantes.

Herramienta que detecta este aviso: PMD

A no ser que el punto y coma sea utilizado para sustituir el cuerpo de un bucle constituye problablemente un error.

Ejemplo:

An empty statement (semicolon) not part of a loop

```
…
    switch (estadoActual) {
      case INICIO:
        …
        break;
      case BASEDATOS:
        …
      case ERROR:
        gestorDatos.borrar();
        throw new ErrorSemantico();
      default:
        break;
    }
    ;
…
```

Este campo "final" debería de ser convertido en "static" (13046)

Tipo de aviso: Aviso de diseño. Aviso de un posible fallo a la hora de diseñar las aplicaciones, puede ser interesante pensar en una Refactorización del código.

Herramienta que detecta este aviso: PMD

Si a un campo final se le asigna una constante de tiempo de compilación, debería de convertirse en "static", esto ahorra la sobrecarga de cada objeto.

Ejemplo:

This final field could be made static

```
public class Buffer {
       //atributos
       private final int TAM_BUFFER=100;
       private final int CAR_RESPALDO=10;
       …
}
```

Campos no modificados deberían de ser estáticos (11021)

Tipo de aviso: Aviso de diseño. Aviso de un posible fallo a la hora de diseñar las aplicaciones, puede ser interesante pensar en una Refactorización del código.

Herramienta que detecta este aviso: FindBugs

Esta clase contiene una instancia de un campo "final" que es inicializado a un valor estático en tiempo de compilación. Se podría convertir en un campo "static".

Ejemplo:

SS: Unread field: modulo1.Buffer.CAR_RESPALDO; should this field be static? (M)

```
public class Buffer {
       //atributos
       private final int TAM_BUFFER=100;
       private final int CAR_RESPALDO=10;
       …
}
```

Un campo de la clase nunca es leido (11023)

Tipo de aviso: Aviso de código no usado. Aviso sobre zonas de código no usadas en el programa.

Herramienta que detecta este aviso: FindBugs

Se asignan valores a una variable; pero esos valores nunca se utilizan para nada por tanto o hay algún problema o esta variable y todo el código que la utiliza sobra.

Ejemplo:

Unread field: Menu.plantillaActual (M)

```
public class ElegirPlantilla {
       …
   String plantillaActual="";

public ElegirPlantilla(JRootPane rootPane) {
    …

plantillaActual=file.devolverLineaFicheroReferencia(plantillaElegida,System.getPr
operty("user.dir")+"\\Plantillas\\Plantillas.txt",1);
   }
}
```

Comparación redundante entre una referencia y NULL (11011)

Tipo de aviso: Aviso de código innecesario.

Herramienta que detecta este aviso: FindBugs

Si es evidente que una referencia es distinto de null, no hace falta compararla.

Ejemplo:

Redundant comparison to null in ImagePreview.loadImage() (M)

```
…
    //Obtiene la imagen del fichero seleccionado
    ImageIcon tmpIcon = new ImageIcon(file.getPath());

    if (tmpIcon != null) {
           if (tmpIcon.getIconWidth() > 90) {
              //Obtiene la imagen del fichero
           thumbnail = new ImageIcon(tmpIcon.getImage().getScaledInstance(90, -
1,Image.SCALE_DEFAULT));
            } else {
                thumbnail = tmpIcon;
            }
       }
    }
```

El valor de la variable referenciada podría ser NULL (12005)

Tipo de aviso: Referencia NULL. Podría haber problemas al utilizar la variable ya que podría ser NULL.

Herramienta que detecta este aviso: JLint

Se utilizan variables sin comprobar si son NULL o no, después de sentencias alternativas que han podido dejar sin asignar la variable.

Ejemplo:

Value of referenced variable 'tip' may be NULL.

```
…
               Tipo tip=null;
               Simbolo s=null;
               if (ei.isUnCampo()){
                      tip=estruct_padre.getCampo(ei.getNombre());
               }else{
                      …
                      }
                   if (s!=null)
                      tip=s.gettipo();

               }
…
               if (tip.getClass()==Funcion.class){
                      System.out.println("Error     ID     es     una     funcion
"+ei.getNombre()+" : "+ei.getN_linea());
                      System.out.println("\n ¿olvidas los \"( )\" ?");
                      System.exit(-501);
               }
…
```

Llamadas innecesarias a métodos (11003)

Tipo de aviso: Aviso de código innecesario. Código que realmente no es necesario, dificulta la comprensión del código y puede afectar al rendimiento.

Herramienta que detecta este aviso: FindBugs

Java garantiza que las constantes cadena idénticas serán representadas por el mismo objeto cadena. Por tanto, se puede emplear directamente la constante cadena directamente: “”

Ejemplo:

Sugerencias.mostrarSugerencias(String) invokes dubious new String() constructor; just use "" (M)

```
public void mostrarSugerencias(String nombreClase){

       String texto=new String();
       nombreFichero="Sugerencias/Sugerencias_"+ nombreClase + ".htm";
       // Crea el texto a partir del fichero de sugerencias y lo muestra en el
diálogo
       FicherosHTML file=new FicherosHTML();
       texto=file.devolverTextoFichero(nombreFichero);
       //Muestra el texto obtenido a partir del fichero
       JOptionPane.showMessageDialog(this,
texto,"Sugerencias",JOptionPane.INFORMATION_MESSAGE);

}
```

Evitar cláusulas return en bloques finally (13011)

Tipo de aviso: Aviso de excepciones. Aviso de un posible problema a la hora de lanzar o detectar excepciones en el código.

Herramienta que detecta este aviso: PMD

La introducción de return dentro de un bloque finally puede hacer que tengamos excepciones no controladas.

Ejemplo:

Avoid returning from a finally block

```
…
       }finally {
          //Se cierra el statement
          if (stmt!=null)  {
            try { stmt.close();
            }catch (SQLException e){
                System.out.println("Error cerrando Statement");
                System.out.println(e.getMessage());
                return -3;
                }
            }
            return 0;
         }//Fin de finally
 }//Fin de actualizarBaseDatos
```

A.3 Errores que aparecen en distintos cursos

Evitar que una variable local o parámetro oculte un componente de la clase (12002)

Tipo de aviso: Aviso de ocultación. Aviso provocado por una posible ocultación de campos o métodos en una clase.

Herramienta que detecta este aviso: JLint

Es una práctica habitual que en los constructores se utilicen como parámetros formales los mismos nombres que componentes de la clase. Dificulta la lectura del código.

Ejemplo:

Local variable 'jugador' shadows component of class 'juego_de_roll/Viaje'.

```
public class Viaje {
       …
       Jugador jugador;
```

```
…
public Viaje(int pasos,Jugador jugador) {
        this.pasos=pasos;
       this.jugador=jugador;
listaViaje=new Elemento[pasos];
        …
}
…
```

Un componente de la clase está ocultando a otro de una clase base (12001)

Tipo de aviso: Aviso de ocultación. Aviso provocado por una posible ocultación de campos o métodos en una clase.

Herramienta que detecta este aviso: JLint

Un campo en una clase derivada tiene el mismo nombre que otro campo en la clase base. Esto puede causar problemas porque estos dos campos apuntan a diferentes localizaciones y los métodos de la clase base no pueden acceder a uno de los campos; mientras que los métodos de la clase derivada (y a su vez las derivadas de esta) no podrán acceder al otro campo. A veces esto es lo que el programador espera; pero en cualquier caso no mejora la legibilidad del programa.

Ejemplo:

Component 'mag' in class 'el_viaje_a_mordor/Jugador' shadows one in base class 'el_viaje_a_mordor/Personaje'.

Comparación de Strings por medio de sus referencias (12014)

Tipo de aviso: Aviso de comparación. Aviso de posibles problemas a la hora de llevar acabo una comparación.

Herramienta que detecta este aviso: JLint

Se está utilizando los operadores == o != para comparar Strings. El operador == sólo devuelve true si los operandos apuntan al mismo objeto; por tanto, puede devolver false aunque los dos operandos contengan lo mismo. Normalmente en la mayoría de las ocasiones se quiere comprobar que dos Strings contengan el mismo valor y por tanto se debería emplear el método “equals”.

Ejemplo:

Compare strings as object references.

```
if(juego.devolverObjetosJugador(i).tipo=="Ataque"){
…
```

Evitar llamadas a métodos redefinibles durante la construcción (13044)

Tipo de aviso: Aviso de inicialización. Aviso sobre un posible problema en las inicializaciones o en los constructores.

Herramienta que detecta este aviso: PMD

Llamar a métodos redefinibles durante la construcción tiene el riesgo de invocar métodos sobre un objeto que todavía no está completamente construido. Esta situación puede ser difícil de percibir. Puede hacer que una subclase sea incapaz de construir su superclase, o

forzar a repetir todo el proceso de construcción dentro de ella, perdiendo la capacidad de llamar a super(). Si el constructor por defecto tiene una llamada a un método redefinible, puede hacer imposible la instanciación de la subclase.

Ejemplo:

Avoid calls to overridable methods during construction

```
…
public Juego() {
        …
        vida=(int)java.lang.Math.round( java.lang.Math.random() * 100);
        …
        //juego preparado...
        int opcion=0;
        interfaz.presentarAtributosJugador(miJugador);
        while (opcion != InterfazUsuario.MENU_SALIR) {
                if (opcion!=InterfazUsuario.MENU_CAMBIAR_MANO)
                        darPaso();
                if (miJugador.estasVivo()&&!estamosEnMordor()){
                        interfaz.presentarEstadoActual();
                opcion =
                        interfaz.menuGeneral(aventuraActual);
                switch (opcion) {
                        …
                        }
                }
        }
}
```

La jerarquía de llamadas a constructores presenta un interesante dilema. ¿Qué ocurre si uno está dentro de un constructor y se invoca a un método que utiliza el enlace dinámico del objeto que se está construyendo? Dentro de un método ordinario se puede imaginar lo que ocurrirá –la llamada que conlleva el enlace dinámico se resuelve en tiempo de ejecución, pues el objeto no puede saber si pertenece a la clase dentro de la que está el método o a alguna clase derivada de esta. Por consistencia, se podría pensar que esto es lo que debería pasar dentro de los constructores.

Este no es exactamente el caso. Si se invoca a un método de enlace dinámico dentro de un constructor, se utiliza la definición superpuesta de ese método. Sin embargo, el efecto puede ser bastante inesperado, y puede ocultar errores difíciles de encontrar.

Conceptualmente, el trabajo del constructor es crear el objeto (lo que es casi una proeza). Dentro de cualquier constructor, el objeto entero podría formarse sólo parcialmente – sólo se puede saber que se han inicializado los objetos de clase base, pero no se puede saber qué clases se heredan. Una llamada a un método de enlace dinámico, sin embargo, se "sale" de la jerarquía de herencias. Llamada a un método de una clase derivada. Si se hace esto dentro del constructor, se llama a un método que podría manipular miembros que no han sido aún inicializados – lo que ocasionará problemas.

Una cabecera de metodo no debería declarar que lanza la excepción "Exception" (13078)

Tipo de aviso: Aviso de excepciones. Aviso de zonas de código que pueden ser propensas a lanzar excepciones, o bien de zonas donde se está haciendo un mal tratamiento de estas.

Herramienta que detecta este aviso: PMD

Esto no proporciona ninguna información sobre las excepciones que realmente puede lanzar el método. Estos interfaces imprecisos hacen que estos métodos sean difíciles de documentar y de entender. Es mejor declarar una subclase de "Exception".

Ejemplo:

A signature (constructor or method) should'nt have Exception in throws declaration

```
public void insertar(Almacenable x, Nodo refLista)
   throws Exception {
        raiz = insertar(x, refLista, raiz);
}
```

La sentencia "switch" debe de tener una etiqueta "default" (13040)

Tipo de aviso: Aviso de estructura de control. Aviso de un posible error en una estructura de control de flujo del programa.

Herramienta que detecta este aviso: PMD

Ejemplo:

Switch statements should have a default label

```
    …
    switch (nivel) {
      case 0:
        if (nombreElemento.compareToIgnoreCase("basedatos") == 0) {
              …
        }
        break;

      case 1:
        if (atributos == 0) {
              …
        else {
              …
        }
        break;

      case 2:
        …
}
```

Evitar reasignación de parámetros (13042)

Tipo de aviso: Aviso de código innecesario. Aviso sobre zonas del código que son innecesarias o redundantes.

Herramienta que detecta este aviso: PMD

Ejemplo:

Avoid reassigning parameters such as 'nb'

```
  private NodoBinario insertar(Almacenable x, Nodo refLista, NodoBinario nb)
throws Exception {
    if (nb == null) {
      nb = new NodoBinario(x, refLista);
    }
    else if (x.comparadoCon(nb) < 0) {
      nb.izq = insertar(x, refLista, nb.izq);
    }
    …
  }
```

A.4 Errores derivados de las convenciones de código

Evitar el uso de sentencias "if…else" sin llaves (13019)

Tipo de aviso: Aviso de estilo. Aviso de una violación de los convenios de estilo de codificación del Lenguaje. Estas normas pueden ayudar a evitar errores en algunas ocasiones.

Herramienta que detecta este aviso: PMD

Explicación:

Tanto la parte "then" como la "else" de una sentencia "if" deberían de ser siempre bloques y estar entre llaves. Esto hace más fácil añadir sentencias sin añadir errores accidentalmente como consecuencia de olvidar añadir las llaves.

Ejemplo de este aviso:

Avoid using 'if...else' statements without curly braces

```
if (estamosEnMordor())
        interfaz.avisarEnMordor(miJugador.getNombre());
else
        interfaz.avisarMuerto();
```

Este es el aviso que se da más frecuentemente con mucha diferencia respecto a los demás y además es común para todos los grupos de proyectos. Es lógico que sea así ya que los alumnos no reciben ninguna instrucción sobre este convenio y en la evaluación tampoco se tiene en cuenta.

Solución:

Simplemente utilizar siempre llaves para la sentencia (aunque sea una sola) correspondiente de "if…else".

Evitar el uso de sentencias "if" sin llaves (13017)

Tipo de aviso: Aviso de estilo. Aviso de una violación de los convenios de estilo de codificación del Lenguaje. Estas normas pueden ayudar a evitar errores en algunas ocasiones.

Herramienta que detecta este aviso: PMD

Es análogo a lo que decíamos en el aviso anterior (13019), si la sentencia correspondiente del "if" está entre llaves, aunque sea una única sentencia, se reduce el riesgo de que al añadir más sentencias nos olvidemos las llaves e introduzcamos un error en el código.

Ejemplo:

Avoid using if statements without curly braces

```
if (opcion!=InterfazUsuario.MENU_CAMBIAR_MANO) darPaso();
```

Los alumnos no tienen en cuenta este convenio de la misma forma que el 13019 por tanto es lógico que se produzca este error.

El nombre de un método no empieza por minúscula (13071)

Tipo de aviso:

Herramienta que detecta este aviso: PMD

Los nombres de los métodos deberían empezar siempre por minúscula y no deberían contener el carácter guión bajo. Si se respetan las convenciones estándar de nombres, el código es más fácil de leer y de mantener.

Ejemplo:

Method name does not begin with a lower case character.

```
public void CrearObjeto(int num){
…
}
```

Evitar el uso de sentencias "for" sin llaves (13020)

Tipo de aviso: Aviso de estilo. Aviso de una violación de los convenios de estilo de codificación del Lenguaje. Estas normas pueden ayudar a evitar errores en algunas ocasiones.

Herramienta que detecta este aviso: PMD

Ejemplo:

Avoid using 'for' statements without curly braces

```
    …
    for(byte i=0; i<inventario.length; i++)
      if(inventario[i]!=null)
        if(inventario[i].getTipo().equalsIgnoreCase(tipo))
          lista += i+" "+inventario[i].getNombre()+'\n';
    …
```

Los nombres de los métodos no deberían contener el carácter guión bajo (13089)

Tipo de aviso: Aviso de nombre. Aviso sobre un problema potencial por no seguir las normas estándares de estilo para los nombres o por una mala elección de los nombres de campos, métodos o clases que pueden llevar a confusiones.

Herramienta que detecta este aviso: PMD

Ejemplo:

Method names should not contain underscores

```
private boolean inicio_elemento() throws Error {
…
}
```

Los nombres de variables deberían comenzar con minúscula (13088)

Tipo de aviso: Aviso de nombre. Aviso sobre un problema potencial por no seguir las normas estándares de estilo para los nombres o por una mala elección de los nombres de campos, métodos o clases que pueden llevar a confusiones.

Herramienta que detecta este aviso: PMD

Ejemplo:

Variables should start with a lowercase character

```
public class ArbolBB {

  //Nodo padre del árbol
  private NodoArbol raiz;
```

```
  boolean Encontrado = false;
    …
}
```

Apéndice C. Avisos por proyecto e informes de análisis

El sistema de análisis de programas desarrollado en el Capítulo 7. Sistema de análisis de errores de programas: PBA permite recopilar los avisos generados por las diferentes herramientas de análisis estático tras el análisis de los proyectos correspondientes. Como se explica en el Capítulo 10. Clasificación de usuarios basada en la detección de errores hemos utilizado el sistema para analizar distintos proyectos agrupados por distintas etapas en el aprendizaje de la programación. En aquel capítulo se hizo un análisis exhaustivo de los 10 errores más frecuentes para cada grupo de proyectos. En este capítulo incluimos, en primer lugar, las tablas completas de resultados del análisis que incluyen todos los avisos generados ordenados por el código que se le ha asignado. Por otra parte, también se incluyen dos informes de errores frecuentes generados directamente por el sistema de análisis de programas.

A.1 Datos generales para todas las tablas

Para realizar el estudio hemos utilizado seis conjuntos de proyectos correspondientes a dos asignaturas de la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo de la Universidad de Oviedo.

Los grupos de proyectos analizados se describen en la Tabla 23 (esta tabla ya se incluyó en el Capítulo 10; pero se vuelve a incluir para mayor facilidad de consulta).

Tabla 23. Grupos de proyectos que se utilizarán en el experimento y descripción de las asignaturas a las que pertenecen

Identificador	Asignatura	Curso y duración		Tipo	Año de las muestras
mpmod1	Metodología de la programación	1º	2º cuatrimestre	Troncal	Curso 2002-2003
mpmod2	Metodología de la programación	1º	2º cuatrimestre	Troncal	Curso 2002-2003
edi3mod1	Estructura de datos y de la información	2º	Anual	Troncal	Curso 2003-2004
edi3mod2	Estructura de datos y de la información	2º	Anual	Troncal	Curso 2003-2004
edi4mod1	Estructura de datos y de la información	2º	Anual	Troncal	Curso 2004-2005
edi4mod2	Estructura de datos y de la información	2º	Anual	Troncal	Curso 2004-2005
bd4mod2	Bases de Datos	3º	Anual	Troncal	Curso 2004-2005
pl4	Procesadores de Lenguaje	4º	Anual	Troncal	Curso 2004-2005
pfc	Proyecto Fin de Carrera	5º		Obligatorio	Distintos cursos

En las tablas de datos se incluyen las siguientes columnas:

- Código error. Es el código asignado al aviso correspondiente.
- Descripción del error. Descripción del aviso detectado.

- Número errores. Número total de avisos de este tipo en el grupo de proyectos analizado.
- Núm. proyectos con errores. Número de proyectos que han generado este tipo de aviso.
- Media de errores por proyecto. Es el número medio de avisos de un tipo determinado en los proyectos que contienen este tipo de error.
- Porcentaje de proy. errores. Es el porcentaje de proyectos que contienen este tipo de avisos respecto al total de proyectos analizados para este grupo.

A.2 Datos completos del análisis de mpmod1

En la Tabla 24 se muestran los datos de todos los avisos detectados en el grupo mpmod1, módulo 1 de la asignatura de Metodología de la Programación del curso 2002-2003, ordenados por código, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 24. Datos completos del análisis de mpmod1

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
1	class <class name> is public, should be declared in a file named <class name>.java	6	6	1,00	3,28%
4	cannot resolve symbol symbol : class <A> location: class <B>	2	1	2,00	0,55%
5	cannot resolve symbol symbol : method <a> location: class <B>	10	3	3,33	1,64%
6	cannot resolve symbol symbol : variable <a> location: class <B>	3	3	1,00	1,64%
8	illegal start of expression	1	1	1,00	0,55%
9	;' expected'	1	1	1,00	0,55%
12	<identifier> expected	1	1	1,00	0,55%
13	)' expected'	37	11	3,36	6,01%
19	incompatible types found : <type found> required: <type required>	2	1	2,00	0,55%
25	unexpected type required: <type name> found : <type name>	7	1	7,00	0,55%
34	package <package name> does not exist	2	1	2,00	0,55%
37	unreachable statement	4	3	1,33	1,64%
38	duplicate class: <class name>	1	1	1,00	0,55%
41	<class name> is not abstract and does not override abstract method <method name> in %	6	1	6,00	0,55%
10005	May be wrong assumption about operators priorities	3	1	3,00	0,55%
10006	May be wrong assumption about logical operators precedence	7	4	1,75	2,19%
10010	May be wrong assumption about bit operation priority	3	1	3,00	0,55%
10011	May be wrong assumption about loop body	51	32	1,59	17,49%
10012	May be wrong assumption about IF body	40	23	1,74	12,57%
10013	May be wrong assumption about ELSE branch association	79	41	1,93	22,40%
10016	Possible miss of BREAK before CASE/DEFAULT	17	14	1,21	7,65%
11002	This method might ignore or drop an exception. In	2	1	2,00	0,55%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
	general, exceptions should be handled or reported in some way, or they should be thrown out of the method.				
11003	Innecesary calls to methods.	7	3	2,33	1,64%
11004	This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by	199	75	2,65	40,98%
11006	Self assignment of field or local variable	32	15	2,13	8,20%
11010	Problems than can cause a NullPointerException.	8	3	2,67	1,64%
11011	This method contains a redundant comparison of a reference value to null.	16	5	3,20	2,73%
11014	This constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructors parameters.'	62	12	5,17	6,56%
11015	Useless control flow statement	6	5	1,20	2,73%
11017	Problems with confusing named methods.	2	2	1,00	1,09%
11020	This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.	159	26	6,12	14,21%
11021	This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.	124	25	4,96	13,66%
11022	This field is never used. Consider removing it from the class.	189	54	3,50	29,51%
11023	This field is never read. Consider removing it from the class.	174	68	2,56	37,16%
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	3	1	3,00	0,55%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	1	1	1,00	0,55%
12000	Method <method name> is not overridden by method with the same name of derived class <class name>.	18	10	1,80	5,46%
12001	Component <name> in class <class name> shadows one in base class <super class name>.	1828	106	17,25	57,92%
12002	Local variable <variable name> shadows component of class <class name>.	487	104	4,68	56,83%
12004	Method <method name> can be invoked with NULL as <position> parameter and this parameter is used without check for null.	1	1	1,00	0,55%
12005	Value of referenced variable <variable name> may be NULL.	97	7	13,86	3,83%
12011	Comparison always produces the same result.	39	20	1,95	10,93%
12012	Compared expressions can be equal only when both of them are 0.	5	4	1,25	2,19%
12014	Compare strings as object references.	458	76	6,03	41,53%
12027	Zero operand for <operation symbol> operation.	17	8	2,13	4,37%
12028	Inequality comparison can be replaced with equality comparison.	19	13	1,46	7,10%
12029	Index [<min>,<max>] may be out of array bounds.	11	7	1,57	3,83%
13000	Avoid empty catch blocks	172	159	1,08	86,89%
13001	Avoid empty if' statements'	16	8	2,00	4,37%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13006	Avoid modifying an outer loop incrementer in an inner loop for update expression	1	1	1,00	0,55%
13016	An empty statement (semicolon) not part of a loop	197	159	1,24	86,89%
13017	Avoid using if statements without curly braces	1504	153	9,83	83,61%
13018	Avoid using while' statements without curly braces'	17	10	1,70	5,46%
13019	Avoid using if...else' statements without curly braces'	3426	160	21,41	87,43%
13020	Avoid using for' statements without curly braces'	1745	161	10,84	87,98%
13036	All methods are static. Consider using Singleton instead.	11	10	1,10	5,46%
13038	Avoid unnecessary if..then..else statements when returning a boolean	24	15	1,60	8,20%
13039	Avoid unnecessary comparisons in boolean expressions	693	110	6,30	60,11%
13040	Switch statements should have a default label	357	116	3,08	63,39%
13042	Avoid reassigning parameters such as <name>"	58	35	1,66	19,13%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	109	63	1,73	34,43%
13044	Avoid calls to overridable methods during construction	1277	151	8,46	82,51%
13046	This final field could be made static	101	25	4,04	13,66%
13051	The default label should be the last label in a switch statement	4	3	1,33	1,64%
13052	A non-case label was present in a switch statement	1	1	1,00	0,55%
13059	Avoid duplicate imports such as <package>"	2	2	1,00	1,09%
13060	Avoid importing anything from the package java.lang"	30	10	3,00	5,46%
13061	Avoid unused imports such as <package>"	7	6	1,17	3,28%
13071	Method name does not begin with a lower case character.	371	43	8,63	23,50%
13072	Class names should begin with an uppercase character and not include underscores	35	12	2,92	6,56%
13073	Abstract classes should be named AbstractXXX"	168	105	1,60	57,38%
13075	Classes should not have non-constructor methods with the same name as the class	1	1	1,00	0,55%
13078	A signature (constructor or method) shouldnt have Exception in throws declaration'	2	1	2,00	0,55%
13080	The same String literal appears <number> times in this file; the first occurrence is on line <line number>	684	161	4,25	87,98%
13081	Avoid instantiating String objects; this is usually unnecessary.	7	3	2,33	1,64%
13082	Avoid calling toString() on String objects; this is unnecessary	1	1	1,00	0,55%
13083	Avoid unused private fields such as <name>"	26	14	1,86	7,65%
13084	Avoid unused local variables such as <name>"	572	160	3,58	87,43%
13085	Avoid unused private methods such as <name>"	1	1	1,00	0,55%
13087	Variables that are not final should not contain underscores.	145	23	6,30	12,57%
13088	Variables should start with a lowercase character	336	38	8,84	20,77%
13089	Method names should not contain underscores	172	17	10,12	9,29%
13090	Variables that are final and static should be in all caps.	13	5	2,60	2,73%
13091	Class names should not contain underscores	19	5	3,80	2,73%

A.3 Datos completos del análisis de mpmod2

En la Tabla 25 se muestran los datos de todos los avisos detectados en el grupo mpmod2, módulo 2 de la asignatura de Metodología de la Programación del curso 2002-2003, ordenados por código, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 25. Datos completos del análisis de mpmod2

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
1	class <class name> is public, should be declared in a file named <class name>.java	1	1	1,00	0,75%
5	cannot resolve symbol symbol : method <a> location: class <B>	5	3	1,67	2,26%
13	)' expected'	413	59	7,00	44,36%
37	unreachable statement	29	24	1,21	18,05%
10005	May be wrong assumption about operators priorities	10	1	10,00	0,75%
10006	May be wrong assumption about logical operators precedence	2	2	1,00	1,50%
10007	May be =' used instead of '=="	2	2	1,00	1,50%
10010	May be wrong assumption about bit operation priority	8	2	4,00	1,50%
10011	May be wrong assumption about loop body	19	14	1,36	10,53%
10012	May be wrong assumption about IF body	122	47	2,60	35,34%
10013	May be wrong assumption about ELSE branch association	116	49	2,37	36,84%
10014	Suspicios SWITCH without body	2	2	1,00	1,50%
10016	Possible miss of BREAK before CASE/DEFAULT	11	11	1,00	8,27%
11001	The return value of this method should be checked.	6	3	2,00	2,26%
11002	This method might ignore or drop an exception. In general, exceptions should be handled or reported in some way, or they should be thrown out of the method.	9	6	1,50	4,51%
11003	Innecesary calls to methods.	28	8	3,50	6,02%
11004	This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by	97	42	2,31	31,58%
11006	Self assignment of field or local variable	23	12	1,92	9,02%
11010	Problems than can cause a NullPointerException.	23	18	1,28	13,53%
11011	This method contains a redundant comparison of a reference value to null.	28	19	1,47	14,29%
11012	The method creates an IO stream object, does not assign it to any fields, pass it to other methods, or return it, and does not appear to close the stream on all paths out of the method. It is generally a good idea to use a finally block to ensure that str	43	33	1,30	24,81%
11014	This constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructors parameters.'	2	2	1,00	1,50%
11015	Useless control flow statement	11	5	2,20	3,76%
11017	Problems with confusing named methods.	2	2	1,00	1,50%
11020	This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.	29	1	29,00	0,75%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
11021	This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.	19	4	4,75	3,01%
11022	This field is never used. Consider removing it from the class.	63	25	2,52	18,80%
11023	This field is never read. Consider removing it from the class.	78	37	2,11	27,82%
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	10	2	5,00	1,50%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	10	7	1,43	5,26%
12001	Component <name> in class <class name> shadows one in base class <super class name>.	45	12	3,75	9,02%
12002	Local variable <variable name> shadows component of class <class name>.	546	72	7,58	54,14%
12004	Method <method name> can be invoked with NULL as <position> parameter and this parameter is used without check for null.	2	2	1,00	1,50%
12005	Value of referenced variable <variable name> may be NULL.	192	32	6,00	24,06%
12011	Comparison always produces the same result.	69	35	1,97	26,32%
12014	Compare strings as object references.	311	43	7,23	32,33%
12027	Zero operand for <operation symbol> operation.	3	2	1,50	1,50%
12028	Inequality comparison can be replaced with equality comparison.	12	7	1,71	5,26%
12029	Index [<min>,<max>] may be out of array bounds.	8	4	2,00	3,01%
13000	Avoid empty catch blocks	244	78	3,13	58,65%
13001	Avoid empty if' statements'	53	17	3,12	12,78%
13002	Avoid empty while' statements'	2	2	1,00	1,50%
13016	An empty statement (semicolon) not part of a loop	92	70	1,31	52,63%
13017	Avoid using if statements without curly braces	1753	112	15,65	84,21%
13018	Avoid using while' statements without curly braces'	129	33	3,91	24,81%
13019	Avoid using if...else' statements without curly braces'	5097	114	44,71	85,71%
13020	Avoid using for' statements without curly braces'	622	75	8,29	56,39%
13036	All methods are static. Consider using Singleton instead.	13	9	1,44	6,77%
13038	Avoid unnecessary if..then..else statements when returning a boolean	21	12	1,75	9,02%
13039	Avoid unnecessary comparisons in boolean expressions	791	73	10,84	54,89%
13040	Switch statements should have a default label	163	75	2,17	56,39%
13042	Avoid reassigning parameters such as <name>"	124	45	2,76	33,83%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	41	36	1,14	27,07%
13044	Avoid calls to overridable methods during construction	1179	109	10,82	81,95%
13046	This final field could be made static	20	5	4,00	3,76%
13059	Avoid duplicate imports such as <package>"	2	2	1,00	1,50%
13060	Avoid importing anything from the package java.lang"	1	1	1,00	0,75%
13061	Avoid unused imports such as <package>"	5	3	1,67	2,26%
13071	Method name does not begin with a lower case	575	49	11,73	36,84%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
	character.				
13072	Class names should begin with an uppercase character and not include underscores	19	9	2,11	6,77%
13073	Abstract classes should be named AbstractXXX"	63	59	1,07	44,36%
13078	A signature (constructor or method) shouldnt have Exception in throws declaration'	181	24	7,54	18,05%
13080	The same String literal appears <number> times in this file; the first occurrence is on line <line number>	529	107	4,94	80,45%
13081	Avoid instantiating String objects; this is usually unnecessary.	33	6	5,50	4,51%
13082	Avoid calling toString() on String objects; this is unnecessary	4	3	1,33	2,26%
13083	Avoid unused private fields such as <name>"	25	15	1,67	11,28%
13084	Avoid unused local variables such as <name>"	609	115	5,30	86,47%
13085	Avoid unused private methods such as <name>"	10	7	1,43	5,26%
13086	Avoid unused formal parameters such as <name>"	5	5	1,00	3,76%
13087	Variables that are not final should not contain underscores.	83	8	10,38	6,02%
13088	Variables should start with a lowercase character	294	41	7,17	30,83%
13089	Method names should not contain underscores	94	6	15,67	4,51%
13091	Class names should not contain underscores	10	3	3,33	2,26%

A.4 Datos completos del análisis de edi3mod1

En la Tabla 26 se muestran los datos de todos los avisos detectados en el grupo edi3mod1, módulo 1 de la asignatura de Estructura de Datos y de la Información del curso 2003-2004, ordenados por código, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 26. Datos completos del análisis de edi3mod1

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
5	cannot resolve symbol symbol : method <a> location: class <B>	3	1	3,00	3,70%
9	;' expected'	1	1	1,00	3,70%
14	missing method body, or declare abstract	10	1	10,00	3,70%
26	not a statement	1	1	1,00	3,70%
37	unreachable statement	1	1	1,00	3,70%
38	duplicate class: <class name>	2	1	2,00	3,70%
10005	May be wrong assumption about operators priorities	3	2	1,50	7,41%
10006	May be wrong assumption about logical operators precedence	1	1	1,00	3,70%
10010	May be wrong assumption about bit operation priority	3	2	1,50	7,41%
10011	May be wrong assumption about loop body	2	2	1,00	7,41%
10012	May be wrong assumption about IF body	15	7	2,14	25,93%
10013	May be wrong assumption about ELSE branch association	26	7	3,71	25,93%
10016	Possible miss of BREAK before CASE/DEFAULT	9	2	4,50	7,41%
11002	This method might ignore or drop an exception. In general, exceptions should be handled or reported in some way, or they should be thrown out of the method.	12	7	1,71	25,93%
11003	Innecesary calls to methods.	15	1	15,00	3,70%
11004	This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by	43	10	4,30	37,04%
11006	Self assignment of field or local variable	1	1	1,00	3,70%
11010	Problems than can cause a NullPointerException.	5	2	2,50	7,41%
11011	This method contains a redundant comparison of a reference value to null.	10	6	1,67	22,22%
11012	The method creates an IO stream object, does not assign it to any fields, pass it to other methods, or return it, and does not appear to close the stream on all paths out of the method. It is generally a good idea to use a finally block to ensure that str	4	4	1,00	14,81%
11014	This constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructors parameters.'	1	1	1,00	3,70%
11015	Useless control flow statement	5	3	1,67	11,11%
11019	This method ignores the return value of one of the variants of java.io.InputStream.read() which can return multiple bytes.	1	1	1,00	3,70%
11020	This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.	4	3	1,33	11,11%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
11021	This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.	176	9	19,56	33,33%
11022	This field is never used. Consider removing it from the class.	20	10	2,00	37,04%
11023	This field is never read. Consider removing it from the class.	19	9	2,11	33,33%
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	2	2	1,00	7,41%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	4	4	1,00	14,81%
12000	Method <method name> is not overridden by method with the same name of derived class <class name>.	9	2	4,50	7,41%
12001	Component <name> in class <class name> shadows one in base class <super class name>.	14	4	3,50	14,81%
12002	Local variable <variable name> shadows component of class <class name>.	76	16	4,75	59,26%
12004	Method <method name> can be invoked with NULL as <position> parameter and this parameter is used without check for null.	2	2	1,00	7,41%
12005	Value of referenced variable <variable name> may be NULL.	27	5	5,40	18,52%
12011	Comparison always produces the same result.	26	9	2,89	33,33%
12012	Compared expressions can be equal only when both of them are 0.	2	1	2,00	3,70%
12014	Compare strings as object references.	71	11	6,45	40,74%
12027	Zero operand for <operation symbol> operation.	2	1	2,00	3,70%
12029	Index [<min>,<max>] may be out of array bounds.	8	6	1,33	22,22%
13000	Avoid empty catch blocks	90	13	6,92	48,15%
13001	Avoid empty if' statements'	11	2	5,50	7,41%
13002	Avoid empty while' statements'	2	1	2,00	3,70%
13009	Ensure you override both equals() and hashCode()	7	1	7,00	3,70%
13015	Do not use if' statements that are always true or always false'	4	2	2,00	7,41%
13016	An empty statement (semicolon) not part of a loop	15	10	1,50	37,04%
13017	Avoid using if statements without curly braces	693	19	36,47	70,37%
13018	Avoid using while' statements without curly braces'	31	11	2,82	40,74%
13019	Avoid using if...else' statements without curly braces'	2041	19	107,42	70,37%
13020	Avoid using for' statements without curly braces'	43	11	3,91	40,74%
13036	All methods are static. Consider using Singleton instead.	8	3	2,67	11,11%
13038	Avoid unnecessary if..then..else statements when returning a boolean	44	11	4,00	40,74%
13039	Avoid unnecessary comparisons in boolean expressions	24	7	3,43	25,93%
13040	Switch statements should have a default label	116	18	6,44	66,67%
13042	Avoid reassigning parameters such as <name>"	20	10	2,00	37,04%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	24	15	1,60	55,56%
13044	Avoid calls to overridable methods during construction	106	13	8,15	48,15%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13046	This final field could be made static	145	10	14,50	37,04%
13059	Avoid duplicate imports such as <package>"	1	1	1,00	3,70%
13060	Avoid importing anything from the package java.lang"	16	5	3,20	18,52%
13061	Avoid unused imports such as <package>"	8	4	2,00	14,81%
13062	No need to import a type thats in the same package'	12	4	3,00	14,81%
13071	Method name does not begin with a lower case character.	208	9	23,11	33,33%
13072	Class names should begin with an uppercase character and not include underscores	5	2	2,50	7,41%
13073	Abstract classes should be named AbstractXXX"	5	5	1,00	18,52%
13078	A signature (constructor or method) shouldnt have Exception in throws declaration'	204	11	18,55	40,74%
13080	The same String literal appears <number> times in this file; the first occurrence is on line <line number>	195	21	9,29	77,78%
13081	Avoid instantiating String objects; this is usually unnecessary.	24	2	12,00	7,41%
13082	Avoid calling toString() on String objects; this is unnecessary	2	1	2,00	3,70%
13083	Avoid unused private fields such as <name>"	15	9	1,67	33,33%
13084	Avoid unused local variables such as <name>"	150	21	7,14	77,78%
13085	Avoid unused private methods such as <name>"	4	4	1,00	14,81%
13087	Variables that are not final should not contain underscores.	66	10	6,60	37,04%
13088	Variables should start with a lowercase character	61	9	6,78	33,33%
13089	Method names should not contain underscores	115	12	9,58	44,44%
13090	Variables that are final and static should be in all caps.	9	2	4,50	7,41%
13091	Class names should not contain underscores	12	5	2,40	18,52%

A.5 Datos completos del análisis de edi3mod2

En la Tabla 27 se muestran los datos de todos los avisos detectados en el grupo edi3mod2, módulo 2 de la asignatura de Estructura de Datos y de la Información del curso 2003-2004, ordenados por código, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 27. Datos completos del análisis de edi3mod2

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
4	cannot resolve symbol symbol : class <A> location: class <B>	4	1	4,00	0,04
5	cannot resolve symbol symbol : method <a> location: class <B>	5	2	2,50	0,08
13	)' expected'	38	4	9,50	0,16
34	package <package name> does not exist	1	1	1,00	0,04
10005	May be wrong assumption about operators priorities	3	2	1,50	0,08
10006	May be wrong assumption about logical operators precedence	1	1	1,00	0,04
10010	May be wrong assumption about bit operation priority	3	2	1,50	0,08
10011	May be wrong assumption about loop body	3	3	1,00	0,12
10012	May be wrong assumption about IF body	21	10	2,10	0,4
10013	May be wrong assumption about ELSE branch association	27	8	3,38	0,32
10016	Possible miss of BREAK before CASE/DEFAULT	9	2	4,50	0,08
11001	The return value of this method should be checked.	3	1	3,00	0,04
11002	This method might ignore or drop an exception. In general, exceptions should be handled or reported in some way, or they should be thrown out of the method.	36	11	3,27	0,44
11003	Innecesary calls to methods.	7	5	1,40	0,2
11004	This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by	45	9	5,00	0,36
11006	Self assignment of field or local variable	2	2	1,00	0,08
11010	Problems than can cause a NullPointerException.	13	5	2,60	0,2
11011	This method contains a redundant comparison of a reference value to null.	11	7	1,57	0,28
11012	The method creates an IO stream object, does not assign it to any fields, pass it to other methods, or return it, and does not appear to close the stream on all paths out of the method. It is generally a good idea to use a finally block to ensure that str	5	4	1,25	0,16
11014	This constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructors parameters.'	3	3	1,00	0,12
11015	Useless control flow statement	7	5	1,40	0,2
11019	This method ignores the return value of one of the variants of java.io.InputStream.read() which can return multiple bytes.	1	1	1,00	0,04
11020	This field is never written. All reads of it will return the default value. Check for errors (should it have	4	3	1,33	0,12

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
	been initialized?), or remove it if it is useless.				
11021	This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.	241	13	18,54	0,52
11022	This field is never used. Consider removing it from the class.	49	14	3,50	0,56
11023	This field is never read. Consider removing it from the class.	43	14	3,07	0,56
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	2	2	1,00	0,08
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	7	6	1,17	0,24
12000	Method <method name> is not overridden by method with the same name of derived class <class name>.	8	3	2,67	0,12
12001	Component <name> in class <class name> shadows one in base class <super class name>.	31	8	3,88	0,32
12002	Local variable <variable name> shadows component of class <class name>.	126	19	6,63	0,76
12004	Method <method name> can be invoked with NULL as <position> parameter and this parameter is used without check for null.	2	2	1,00	0,08
12005	Value of referenced variable <variable name> may be NULL.	52	11	4,73	0,44
12011	Comparison always produces the same result.	27	11	2,45	0,44
12012	Compared expressions can be equal only when both of them are 0.	2	1	2,00	0,04
12014	Compare strings as object references.	73	10	7,30	0,4
12027	Zero operand for <operation symbol> operation.	7	3	2,33	0,12
12028	Inequality comparison can be replaced with equality comparison.	1	1	1,00	0,04
12029	Index [<min>,<max>] may be out of array bounds.	8	6	1,33	0,24
13000	Avoid empty catch blocks	333	18	18,50	0,72
13001	Avoid empty if' statements'	14	4	3,50	0,16
13002	Avoid empty while' statements'	3	2	1,50	0,08
13009	Ensure you override both equals() and hashCode()	11	1	11,00	0,04
13015	Do not use if' statements that are always true or always false'	4	2	2,00	0,08
13016	An empty statement (semicolon) not part of a loop	23	12	1,92	0,48
13017	Avoid using if statements without curly braces	865	19	45,53	0,76
13018	Avoid using while' statements without curly braces'	43	11	3,91	0,44
13019	Avoid using if...else' statements without curly braces'	2441	20	122,05	0,8
13020	Avoid using for' statements without curly braces'	121	17	7,12	0,68
13036	All methods are static. Consider using Singleton instead.	20	7	2,86	0,28
13038	Avoid unnecessary if..then..else statements when returning a boolean	51	13	3,92	0,52
13039	Avoid unnecessary comparisons in boolean expressions	40	9	4,44	0,36
13040	Switch statements should have a default label	125	18	6,94	0,72
13042	Avoid reassigning parameters such as <name>"	49	20	2,45	0,8

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	24	16	1,50	0,64
13044	Avoid calls to overridable methods during construction	76	16	4,75	0,64
13046	This final field could be made static	210	14	15,00	0,56
13047	Avoid instantiating Boolean objects; you can usually invoke Boolean.valueOf() instead.	12	3	4,00	0,12
13049	Object clone() should be implemented with super.clone()	3	1	3,00	0,04
13059	Avoid duplicate imports such as <package>''	11	3	3,67	0,12
13060	Avoid importing anything from the package java.lang''	26	10	2,60	0,4
13061	Avoid unused imports such as <package>''	24	7	3,43	0,28
13062	No need to import a type thats in the same package'	12	4	3,00	0,16
13071	Method name does not begin with a lower case character.	301	14	21,50	0,56
13072	Class names should begin with an uppercase character and not include underscores	13	3	4,33	0,12
13073	Abstract classes should be named AbstractXXX''	12	7	1,71	0,28
13078	A signature (constructor or method) shouldnt have Exception in throws declaration'	369	19	19,42	0,76
13080	The same String literal appears <number> times in this file; the first occurrence is on line <line number>	254	22	11,55	0,88
13081	Avoid instantiating String objects; this is usually unnecessary.	2	1	2,00	0,04
13082	Avoid calling toString() on String objects; this is unnecessary	2	1	2,00	0,04
13083	Avoid unused private fields such as <name>''	18	11	1,64	0,44
13084	Avoid unused local variables such as <name>''	221	22	10,05	0,88
13085	Avoid unused private methods such as <name>''	6	6	1,00	0,24
13086	Avoid unused formal parameters such as <name>''	2	2	1,00	0,08
13087	Variables that are not final should not contain underscores.	67	10	6,70	0,4
13088	Variables should start with a lowercase character	92	11	8,36	0,44
13089	Method names should not contain underscores	122	13	9,38	0,52
13090	Variables that are final and static should be in all caps.	10	3	3,33	0,12
13091	Class names should not contain underscores	13	5	2,60	0,2

A.6 Datos completos del análisis de edi4mod1

En la Tabla 28 se muestran los datos de todos los avisos detectados en el grupo edi4mod1, módulo 1 de la asignatura de Estructura de Datos y de la Información, ordenados por código, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 28. Datos completos del análisis de edi4mod1

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
1	class <class name> is public, should be declared in a file named <class name>.java	1	1	1,00	0,83%
4	cannot resolve symbol symbol : class <A> location: class <B>	13	2	6,50	1,65%
5	cannot resolve symbol symbol : method <a> location: class <B>	85	3	28,33	2,48%
6	cannot resolve symbol symbol : variable <a> location: class <B>	5	2	2,50	1,65%
9	;' expected'	32	2	16,00	1,65%
11	illegal character: <character in hexadecimal>	3	1	3,00	0,83%
13	)' expected'	13	7	1,86	5,79%
26	not a statement	11	3	3,67	2,48%
34	package <package name> does not exist	8	1	8,00	0,83%
37	unreachable statement	3	3	1,00	2,48%
38	duplicate class: <class name>	59	5	11,80	4,13%
10005	May be wrong assumption about operators priorities	5	2	2,50	1,65%
10006	May be wrong assumption about logical operators precedence	8	6	1,33	4,96%
10007	May be =' used instead of '=="	6	3	2,00	2,48%
10010	May be wrong assumption about bit operation priority	3	1	3,00	0,83%
10011	May be wrong assumption about loop body	24	16	1,50	13,22%
10012	May be wrong assumption about IF body	135	42	3,21	34,71%
10013	May be wrong assumption about ELSE branch association	323	58	5,57	47,93%
10016	Possible miss of BREAK before CASE/DEFAULT	40	23	1,74	19,01%
11001	The return value of this method should be checked.	3	2	1,50	1,65%
11002	This method might ignore or drop an exception. In general, exceptions should be handled or reported in some way, or they should be thrown out of the method.	62	29	2,14	23,97%
11003	Innecesary calls to methods.	166	19	8,74	15,70%
11004	This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by	60	26	2,31	21,49%
11006	Self assignment of field or local variable	14	9	1,56	7,44%
11007	Problems with the method finalize() use.	56	5	11,20	4,13%
11010	Problems than can cause a NullPointerException.	26	12	2,17	9,92%
11011	This method contains a redundant comparison of a reference value to null.	32	11	2,91	9,09%
11012	The method creates an IO stream object, does not assign it to any fields, pass it to other methods, or return it, and does not appear to	39	28	1,39	23,14%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
	close the stream on all paths out of the method. It is generally a good idea to use a finally block to ensure that str				
11014	This constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructors parameters.'	7	7	1,00	5,79%
11015	Useless control flow statement	26	13	2,00	10,74%
11017	Problems with confusing named methods.	3	3	1,00	2,48%
11019	This method ignores the return value of one of the variants of java.io.InputStream.read() which can return multiple bytes.	10	9	1,11	7,44%
11020	This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.	26	15	1,73	12,40%
11021	This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.	1021	73	13,99	60,33%
11022	This field is never used. Consider removing it from the class.	142	56	2,54	46,28%
11023	This field is never read. Consider removing it from the class.	228	75	3,04	61,98%
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	3	2	1,50	1,65%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	43	26	1,65	21,49%
12000	Method <method name> is not overridden by method with the same name of derived class <class name>.	1	1	1,00	0,83%
12001	Component <name> in class <class name> shadows one in base class <super class name>.	16	4	4,00	3,31%
12002	Local variable <variable name> shadows component of class <class name>.	483	88	5,49	72,73%
12003	Method finalize() doesnt call super.finalize().'	37	5	7,40	4,13%
12004	Method <method name> can be invoked with NULL as <position> parameter and this parameter is used without check for null.	2	2	1,00	1,65%
12005	Value of referenced variable <variable name> may be NULL.	183	28	6,54	23,14%
12011	Comparison always produces the same result.	37	22	1,68	18,18%
12012	Compared expressions can be equal only when both of them are 0.	32	10	3,20	8,26%
12014	Compare strings as object references.	156	29	5,38	23,97%
12027	Zero operand for <operation symbol> operation.	4	3	1,33	2,48%
12028	Inequality comparison can be replaced with equality comparison.	108	15	7,20	12,40%
12029	Index [<min>,<max>] may be out of array bounds.	25	11	2,27	9,09%
13000	Avoid empty catch blocks	347	67	5,18	55,37%
13001	Avoid empty if' statements'	100	33	3,03	27,27%
13002	Avoid empty while' statements'	7	4	1,75	3,31%
13004	Avoid empty finally blocks	1	1	1,00	0,83%
13006	Avoid modifying an outer loop incrementer in an inner loop for update expression	2	2	1,00	1,65%
13013	Avoid unnecessary return statements	16	3	5,33	2,48%
13015	Do not use if' statements that are always true or always false'	4	3	1,33	2,48%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13016	An empty statement (semicolon) not part of a loop	180	39	4,62	32,23%
13017	Avoid using if statements without curly braces	2547	92	27,68	76,03%
13018	Avoid using while' statements without curly braces'	133	47	2,83	38,84%
13019	Avoid using if...else' statements without curly braces'	7050	96	73,44	79,34%
13020	Avoid using for' statements without curly braces'	218	67	3,25	55,37%
13036	All methods are static. Consider using Singleton instead.	70	42	1,67	34,71%
13038	Avoid unnecessary if..then..else statements when returning a boolean	157	54	2,91	44,63%
13039	Avoid unnecessary comparisons in boolean expressions	334	42	7,95	34,71%
13040	Switch statements should have a default label	349	90	3,88	74,38%
13042	Avoid reassigning parameters such as <name>"	412	95	4,34	78,51%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	67	45	1,49	37,19%
13044	Avoid calls to overridable methods during construction	260	61	4,26	50,41%
13046	This final field could be made static	1192	77	15,48	63,64%
13053	Avoid empty finalize methods	3	2	1,50	1,65%
13056	Last statement in finalize method should be a call to super.finalize()	34	4	8,50	3,31%
13057	Explicit call of finalize method	16	1	16,00	0,83%
13058	If you override finalize(), make it protected	37	4	9,25	3,31%
13059	Avoid duplicate imports such as <package>"	19	6	3,17	4,96%
13060	Avoid importing anything from the package java.lang"	47	14	3,36	11,57%
13061	Avoid unused imports such as <package>"	11	5	2,20	4,13%
13062	No need to import a type thats in the same package'	7	3	2,33	2,48%
13071	Method name does not begin with a lower case character.	1016	39	26,05	32,23%
13072	Class names should begin with an uppercase character and not include underscores	43	16	2,69	13,22%
13073	Abstract classes should be named AbstractXXX"	9	7	1,29	5,79%
13075	Classes should not have non-constructor methods with the same name as the class	3	3	1,00	2,48%
13078	A signature (constructor or method) shouldnt have Exception in throws declaration'	246	25	9,84	20,66%
13080	The same String literal appears <number> times in this file; the first occurrence is on line <line number>	628	100	6,28	82,64%
13081	Avoid instantiating String objects; this is usually unnecessary.	249	16	15,56	13,22%
13082	Avoid calling toString() on String objects; this is unnecessary	2	2	1,00	1,65%
13083	Avoid unused private fields such as <name>"	143	63	2,27	52,07%
13084	Avoid unused local variables such as <name>"	513	102	5,03	84,30%
13085	Avoid unused private methods such as <name>"	39	23	1,70	19,01%
13086	Avoid unused formal parameters such as <name>"	24	17	1,41	14,05%
13087	Variables that are not final should not contain underscores.	253	47	5,38	38,84%
13088	Variables should start with a lowercase character	306	51	6,00	42,15%
13089	Method names should not contain underscores	404	76	5,32	62,81%
13090	Variables that are final and static should be in all caps.	34	14	2,43	11,57%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13091	Class names should not contain underscores	8	4	2,00	3,31%

A.7 Datos completos del análisis de edi4mod2

En la Tabla 29 se muestran los datos de todos los avisos detectados en el grupo edi4mod2, módulo 2 de la asignatura de Estructura de datos y de la Información del curso 2004-2005, ordenados por código, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 29. Datos completos del análisis de edi4mod2

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
4	cannot resolve symbol symbol : class <A> location: class <B>	75	1	75,00	0,61%
5	cannot resolve symbol symbol : method <a> location: class <B>	2	1	2,00	0,61%
9	;' expected'	34	3	11,33	1,83%
13	)' expected'	58	24	2,42	14,63%
26	not a statement	2	2	1,00	1,22%
34	package <package name> does not exist	1	1	1,00	0,61%
37	unreachable statement	7	2	3,50	1,22%
40	cannot resolve symbol symbol : class <class name> location: interface <interface name>	1	1	1,00	0,61%
10005	May be wrong assumption about operators priorities	54	14	3,86	8,54%
10006	May be wrong assumption about logical operators precedence	14	10	1,40	6,10%
10007	May be =' used instead of '=="	6	5	1,20	3,05%
10010	May be wrong assumption about bit operation priority	29	3	9,67	1,83%
10011	May be wrong assumption about loop body	83	35	2,37	21,34%
10012	May be wrong assumption about IF body	226	63	3,59	38,41%
10013	May be wrong assumption about ELSE branch association	644	93	6,92	56,71%
10016	Possible miss of BREAK before CASE/DEFAULT	84	43	1,95	26,22%
11001	The return value of this method should be checked.	27	10	2,70	6,10%
11002	This method might ignore or drop an exception. In general, exceptions should be handled or reported in some way, or they should be thrown out of the method.	96	47	2,04	28,66%
11003	Innecesary calls to methods.	301	43	7,00	26,22%
11004	This code compares java.lang.String objects for reference equality using the == or != operators. Unless both strings are either constants in a source file, or have been interned using the String.intern() method, the same string value may be represented by	130	55	2,36	33,54%
11006	Self assignment of field or local variable	41	29	1,41	17,68%
11007	Problems with the method finalize() use.	56	5	11,20	3,05%
11010	Problems than can cause a NullPointerException.	42	25	1,68	15,24%
11011	This method contains a redundant comparison of a reference value to null.	76	35	2,17	21,34%
11012	The method creates an IO stream object, does not assign it to any fields, pass it to other	87	53	1,64	32,32%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
	methods, or return it, and does not appear to close the stream on all paths out of the method. It is generally a good idea to use a finally block to ensure that str				
11014	This constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructors parameters.'	45	31	1,45	18,90%
11015	Useless control flow statement	36	18	2,00	10,98%
11017	Problems with confusing named methods.	7	7	1,00	4,27%
11019	This method ignores the return value of one of the variants of java.io.InputStream.read() which can return multiple bytes.	14	12	1,17	7,32%
11020	This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.	50	28	1,79	17,07%
11021	This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.	2354	116	20,29	70,73%
11022	This field is never used. Consider removing it from the class.	315	94	3,35	57,32%
11023	This field is never read. Consider removing it from the class.	440	121	3,64	73,78%
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	29	7	4,14	4,27%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	118	62	1,90	37,80%
12000	Method <method name> is not overridden by method with the same name of derived class <class name>.	15	10	1,50	6,10%
12001	Component <name> in class <class name> shadows one in base class <super class name>.	346	36	9,61	21,95%
12002	Local variable <variable name> shadows component of class <class name>.	999	139	7,19	84,76%
12003	Method finalize() doesnt call super.finalize().'	37	5	7,40	3,05%
12004	Method <method name> can be invoked with NULL as <position> parameter and this parameter is used without check for null.	15	12	1,25	7,32%
12005	Value of referenced variable <variable name> may be NULL.	344	60	5,73	36,59%
12007	Result of operation <operation> is always 0.	7	3	2,33	1,83%
12011	Comparison always produces the same result.	149	64	2,33	39,02%
12012	Compared expressions can be equal only when both of them are 0.	64	17	3,76	10,37%
12014	Compare strings as object references.	244	62	3,94	37,80%
12015	Switch case constant <constant> cant be produced by switch expression.'	3	2	1,50	1,22%
12027	Zero operand for <operation symbol> operation.	26	12	2,17	7,32%
12028	Inequality comparison can be replaced with equality comparison.	88	25	3,52	15,24%
12029	Index [<min>,<max>] may be out of array bounds.	77	24	3,21	14,63%
13000	Avoid empty catch blocks	994	109	9,12	66,46%
13001	Avoid empty if' statements'	213	68	3,13	41,46%
13002	Avoid empty while' statements'	12	7	1,71	4,27%
13004	Avoid empty finally blocks	2	2	1,00	1,22%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13006	Avoid modifying an outer loop incrementer in an inner loop for update expression	1	1	1,00	0,61%
13008	Avoid unnecessary temporaries when converting primitives to Strings	1	1	1,00	0,61%
13009	Ensure you override both equals() and hashCode()	13	2	6,50	1,22%
13013	Avoid unnecessary return statements	18	10	1,80	6,10%
13015	Do not use if' statements that are always true or always false'	3	2	1,50	1,22%
13016	An empty statement (semicolon) not part of a loop	291	68	4,28	41,46%
13017	Avoid using if statements without curly braces	4687	127	36,91	77,44%
13018	Avoid using while' statements without curly braces'	245	67	3,66	40,85%
13019	Avoid using if...else' statements without curly braces'	12213	131	93,23	79,88%
13020	Avoid using for' statements without curly braces'	582	97	6,00	59,15%
13036	All methods are static. Consider using Singleton instead.	109	75	1,45	45,73%
13038	Avoid unnecessary if..then..else statements when returning a boolean	339	83	4,08	50,61%
13039	Avoid unnecessary comparisons in boolean expressions	723	77	9,39	46,95%
13040	Switch statements should have a default label	689	128	5,38	78,05%
13042	Avoid reassigning parameters such as <name>"	861	142	6,06	86,59%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	186	83	2,24	50,61%
13044	Avoid calls to overridable methods during construction	519	92	5,64	56,10%
13046	This final field could be made static	2614	123	21,25	75,00%
13049	Object clone() should be implemented with super.clone()	2	1	2,00	0,61%
13050	Non-static initializers are confusing	1	1	1,00	0,61%
13051	The default label should be the last label in a switch statement	2	2	1,00	1,22%
13053	Avoid empty finalize methods	3	1	3,00	0,61%
13056	Last statement in finalize method should be a call to super.finalize()	34	5	6,80	3,05%
13057	Explicit call of finalize method	15	3	5,00	1,83%
13058	If you override finalize(), make it protected	39	5	7,80	3,05%
13059	Avoid duplicate imports such as <package>"	53	18	2,94	10,98%
13060	Avoid importing anything from the package java.lang"	200	55	3,64	33,54%
13061	Avoid unused imports such as <package>"	100	38	2,63	23,17%
13062	No need to import a type thats in the same package'	98	5	19,60	3,05%
13071	Method name does not begin with a lower case character.	2249	95	23,67	57,93%
13072	Class names should begin with an uppercase character and not include underscores	56	18	3,11	10,98%
13073	Abstract classes should be named AbstractXXX"	122	67	1,82	40,85%
13075	Classes should not have non-constructor methods with the same name as the class	7	7	1,00	4,27%
13078	A signature (constructor or method) shouldnt have Exception in throws declaration'	539	43	12,53	26,22%
13080	The same String literal appears <number> times in this file; the first occurrence is on line <line number>	1172	144	8,14	87,80%
13081	Avoid instantiating String objects; this is usually unnecessary.	410	40	10,25	24,39%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13082	Avoid calling toString() on String objects; this is unnecessary	10	7	1,43	4,27%
13083	Avoid unused private fields such as <name>''	425	103	4,13	62,80%
13084	Avoid unused local variables such as <name>''	1109	147	7,54	89,63%
13085	Avoid unused private methods such as <name>''	108	55	1,96	33,54%
13086	Avoid unused formal parameters such as <name>''	40	28	1,43	17,07%
13087	Variables that are not final should not contain underscores.	790	136	5,81	82,93%
13088	Variables should start with a lowercase character	1834	137	13,39	83,54%
13089	Method names should not contain underscores	1688	146	11,56	89,02%
13090	Variables that are final and static should be in all caps.	124	43	2,88	26,22%
13091	Class names should not contain underscores	22	9	2,44	5,49%

A.8 Datos completos del análisis de bd4mod2

En la Tabla 30 se muestran los datos de todos los avisos detectados en el grupo bd4mod2, módulo 2 de la asignatura de Bases de Datos de tercer curso, ordenados por el número de errores que han sido detectados de cada tipo, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 30. Datos completos del análisis de bd4mod2

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13089	Method names should not contain underscores	924	15	61,60	68,18%
13091	Class names should not contain underscores	752	15	50,13	68,18%
13087	Variables that are not final should not contain underscores.	496	14	35,43	63,64%
13088	Variables should start with a lowercase character	324	10	32,40	45,45%
13078	A method shouldnt have Exception in throws declaration.'	269	15	17,93	68,18%
13071	Method name does not begin with a lower case character.	182	12	15,17	54,55%
13019	Avoid using "if...else" statements without curly braces	130	15	8,67	68,18%
13084	Avoid unused local variables.	113	13	8,69	59,09%
11023	This field is never read. Consider removing it from the class.	113	13	8,69	59,09%
13080	The same String literal appears several times in this file.	111	17	6,53	77,27%
38	duplicate class: <class name>	107	3	35,67	13,64%
13044	Avoid calls to overridable methods during construction	97	16	6,06	72,73%
13061	Avoid unused imports such as <package>"	58	13	4,46	59,09%
13059	Avoid duplicate imports such as <package>"	38	12	3,17	54,55%
13017	Avoid using "if" statements without curly braces	26	11	2,36	50,00%
11011	This method contains a redundant comparison of a reference value to null.	25	2	12,50	9,09%
13000	Avoid empty catch blocks	21	5	4,20	22,73%
13020	Avoid using "for" statements without curly braces	16	4	4,00	18,18%
13039	Avoid unnecessary comparisons in boolean expressions	15	5	3,00	22,73%
13083	Avoid unused private fields.	15	5	3,00	22,73%
13	")" expected	13	5	2,60	22,73%
11022	This field is never used. Consider removing it from the class.	13	7	1,86	31,82%
10013	May be wrong assumption about ELSE branch association	13	6	2,17	27,27%
13040	Switch statements should have a default label	12	5	2,40	22,73%
13036	All methods are static. Consider using Singleton instead.	7	5	1,40	22,73%
11006	Self assignment of field or local variable.	5	4	1,25	18,18%
13038	Avoid unnecessary if..then..else statements when returning a boolean	5	2	2,50	9,09%
10012	May be wrong assumption about IF body	5	3	1,67	13,64%
13072	Class names should begin with an uppercase character and not include underscores	5	2	2,50	9,09%
11014	Uninitialized read of field in constructor	4	3	1,33	13,64%
10011	May be wrong assumption about loop body	3	2	1,50	9,09%
13016	An empty statement (semicolon) not part of a loop	3	3	1,00	13,64%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
11007	Problems with the method finalize() use.	2	1	2,00	4,55%
13042	Avoid reassigning parameters.	2	1	2,00	4,55%
13056	Last statement in finalize method should be a call to super.finalize()	2	1	2,00	4,55%
13085	Avoid unused private methods.	2	2	1,00	9,09%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	2	2	1,00	9,09%
13058	If you override finalize(), make it protected	2	1	2,00	4,55%
13060	Avoid importing anything from the package java.lang"	2	2	1,00	9,09%
13090	Variables that are final and static should be in all caps.	2	1	2,00	4,55%
11002	Method might drop exception or ignore exception.	2	2	1,00	9,09%
11004	Comparison of String objects using "==" or "!="	2	1	2,00	4,55%
13073	Abstract classes should be named AbstractXXX"	1	1	1,00	4,55%
13081	Avoid instantiating String objects; this is usually unnecessary.	1	1	1,00	4,55%
11020	Unwritten field.	1	1	1,00	4,55%
34	package <package name> does not exist	1	1	1,00	4,55%
37	unreachable statement	1	1	1,00	4,55%

A.9 Datos completos del análisis de pl4

En la Tabla 31 se muestran los datos de todos los avisos detectados en el grupo pl4, de la asignatura de Procesadores de Lenguaje de cuarto curso, ordenados por el número de errores que han aparecido de cada tipo, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 31. Datos completos del análisis de pl4

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13019	Avoid using "if...else" statements without curly braces	427	4	106,75	80,00%
13017	Avoid using "if" statements without curly braces	350	4	87,50	80,00%
13087	Variables that are not final should not contain underscores.	262	2	131,00	40,00%
13071	Method name does not begin with a lower case character.	182	1	182,00	20,00%
13089	Method names should not contain underscores	98	4	24,50	80,00%
13088	Variables should start with a lowercase character	76	2	38,00	40,00%
13084	Avoid unused local variables.	64	3	21,33	60,00%
13090	Variables that are final and static should be in all caps.	57	4	14,25	80,00%
13080	The same String literal appears several times in this file.	50	4	12,50	80,00%
13081	Avoid instantiating String objects; this is usually unnecessary.	47	2	23,50	40,00%
13020	Avoid using "for" statements without curly braces	45	4	11,25	80,00%
13078	A method shouldnt have Exception in throws declaration.'	34	3	11,33	60,00%
11003	Innecesary calls to methods.	33	2	16,50	40,00%
13039	Avoid unnecessary comparisons in boolean expressions	30	1	30,00	20,00%
12005	Value of referenced variable may be NULL.	23	1	23,00	20,00%
13091	Class names should not contain underscores	21	1	21,00	20,00%
12001	Component in this class shadows one in base class.	21	3	7,00	60,00%
13000	Avoid empty catch blocks	20	2	10,00	40,00%
12014	Compare strings as object references.	18	2	9,00	40,00%
13001	Avoid empty if' statements'	18	2	9,00	40,00%
13061	Avoid unused imports such as <package>"	17	1	17,00	20,00%
13073	Abstract classes should be named AbstractXXX"	17	4	4,25	80,00%
12002	Local variable shadows component of class.	15	2	7,50	40,00%
13040	Switch statements should have a default label	11	4	2,75	80,00%
5	cannot resolve symbol symbol : method <a> location: class <B>	11	1	11,00	20,00%
11004	Comparison of String objects using "==" or "!="	9	2	4,50	40,00%
11022	This field is never used. Consider removing it from the class.	9	3	3,00	60,00%
13042	Avoid reassigning parameters.	8	4	2,00	80,00%
11023	This field is never read. Consider removing it from the class.	8	3	2,67	60,00%
13082	Avoid calling toString() on String objects; this is unnecessary	7	1	7,00	20,00%
13059	Avoid duplicate imports such as <package>"	7	2	3,50	40,00%
13018	Avoid using "while" statements without curly braces	7	1	7,00	20,00%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13036	All methods are static. Consider using Singleton instead.	6	3	2,00	60,00%
11015	Useless control flow statement	5	1	5,00	20,00%
13083	Avoid unused private fields.	4	1	4,00	20,00%
13009	Ensure you override both equals() and hashCode()	4	1	4,00	20,00%
13044	Avoid calls to overridable methods during construction	3	3	1,00	60,00%
13016	An empty statement (semicolon) not part of a loop	3	1	3,00	20,00%
13072	Class names should begin with an uppercase character and not include underscores	3	2	1,50	40,00%
13079	The catch clause shouldnt check the exception type - catch several exceptions instead'	2	1	2,00	20,00%
11002	Method might drop exception or ignore exception.	2	1	2,00	20,00%
11010	Problems than can cause a NullPointerException.	2	2	1,00	40,00%
11020	Unwritten field.	2	2	1,00	40,00%
12011	Comparison always produces the same result.	1	1	1,00	20,00%
12029	Index [<min>,<max>] may be out of array bounds.	1	1	1,00	20,00%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	1	1	1,00	20,00%
13060	Avoid importing anything from the package java.lang''	1	1	1,00	20,00%

A.10 Datos completos del análisis de pfc

En la Tabla 32 se muestran los datos de todos los avisos detectados en el grupo pfc: proyectos fin de carrera del último curso de la titulación de ingeniería en informática ordenados por el número de errores detectado, sólo aparecen en la tabla los avisos que tienen al menos una ocurrencia en uno de los proyectos.

Tabla 32. Datos completos del análisis de pfc

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13019	Avoid using "if...else" statements without curly braces	2352	4	588,00	44,44%
13017	Avoid using "if" statements without curly braces	1381	4	345,25	44,44%
13080	The same String literal appears several times in this file.	354	4	88,50	44,44%
13088	Variables should start with a lowercase character	221	4	55,25	44,44%
13061	Avoid unused imports such as <package>"	220	1	220,00	11,11%
13000	Avoid empty catch blocks	128	2	64,00	22,22%
13084	Avoid unused local variables.	119	4	29,75	44,44%
13081	Avoid instantiating String objects; this is usually unnecessary.	116	4	29,00	44,44%
13044	Avoid calls to overridable methods during construction	110	2	55,00	22,22%
13011	Avoid returning from a finally block	98	3	32,67	33,33%
6	cannot resolve symbol symbol : variable <a> location: class <B>	92	1	92,00	11,11%
13071	Method name does not begin with a lower case character.	88	2	44,00	22,22%
13039	Avoid unnecessary comparisons in boolean expressions	88	3	29,33	33,33%
13090	Variables that are final and static should be in all caps.	88	2	44,00	22,22%
13020	Avoid using "for" statements without curly braces	72	2	36,00	22,22%
13087	Variables that are not final should not contain underscores.	67	2	33,50	22,22%
13078	A method shouldnt have Exception in throws declaration.'	65	1	65,00	11,11%
13042	Avoid reassigning parameters.	54	3	18,00	33,33%
13059	Avoid duplicate imports such as <package>"	52	1	52,00	11,11%
13060	Avoid importing anything from the package java.lang"	47	1	47,00	11,11%
13040	Switch statements should have a default label	39	3	13,00	33,33%
13077	A catch statement should never catch throwable since it includes errors	35	1	35,00	11,11%
13083	Avoid unused private fields.	29	2	14,50	22,22%
12002	Local variable shadows component of class.	20	2	10,00	22,22%
13089	Method names should not contain underscores	19	1	19,00	11,11%
13016	An empty statement (semicolon) not part of a loop	17	1	17,00	11,11%
13001	Avoid empty if' statements'	16	1	16,00	11,11%
12014	Compare strings as object references.	15	2	7,50	22,22%
38	duplicate class: <class name>	13	1	13,00	11,11%
11003	Innecesary calls to methods.	12	2	6,00	22,22%
13018	Avoid using "while" statements without curly braces	12	1	12,00	11,11%

Código error	Descripción del error	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13047	Avoid instantiating Boolean objects; you can usually invoke Boolean.valueOf() instead.	11	1	11,00	11,11%
13073	Abstract classes should be named AbstractXXX"	10	1	10,00	11,11%
13036	All methods are static. Consider using Singleton instead.	8	2	4,00	22,22%
13079	The catch clause shouldnt check the exception type - catch several exceptions instead'	7	1	7,00	11,11%
13038	Avoid unnecessary if..then..else statements when returning a boolean	7	3	2,33	33,33%
13008	Avoid unnecessary temporaries when converting primitives to Strings	7	1	7,00	11,11%
11004	Comparison of String objects using "==" or "!="	6	2	3,00	22,22%
13013	Avoid unnecessary return statements	6	1	6,00	11,11%
13086	Avoid unused formal parameters.	6	2	3,00	22,22%
13009	Ensure you override both equals() and hashCode()	5	1	5,00	11,11%
13085	Avoid unused private methods.	5	2	2,50	22,22%
11011	Redundant comparison of a reference value to null.	4	2	2,00	22,22%
11023	This field is never read. Consider removing it from the class.	4	1	4,00	11,11%
12011	Comparison always produces the same result.	4	1	4,00	11,11%
13015	Do not use if' statements that are always true or always false'	3	1	3,00	11,11%
11025	This code seems to be using non-short-circuit logic (e.g., & or |) rather than short-circuit logic (&& or ||). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side	3	1	3,00	11,11%
13062	No need to import a type thats in the same package'	3	1	3,00	11,11%
34	package <package name> does not exist	3	1	3,00	11,11%
13091	Class names should not contain underscores	3	1	3,00	11,11%
5	cannot resolve symbol symbol : method <a> location: class <B>	2	1	2,00	11,11%
11026	This private method is never called. Although it is possible that the method will be invoked through reflection, it is more likely that the method is never used, and should be removed.	2	1	2,00	11,11%
27	.' expected'	2	1	2,00	11,11%
39	inconvertible types found : <type name> required: <type name>	2	1	2,00	11,11%
13043	A high ratio of statements to labels in a switch statement. Consider refactoring.	1	1	1,00	11,11%
1	class <class name> is public, should be declared in a file named <class name>.java	1	1	1,00	11,11%
13046	This final field could be made static	1	1	1,00	11,11%
4	cannot resolve symbol symbol : class <A> location: class <B>	1	1	1,00	11,11%
19	incompatible types found : <type found> required: <type required>	1	1	1,00	11,11%
12028	Inequality comparison can be replaced with equality comparison.	1	1	1,00	11,11%
13072	Class names should begin with an uppercase character and not include underscores	1	1	1,00	11,11%
11002	Method might drop exception or ignore exception.	1	1	1,00	11,11%

A.11 Página de Errores frecuentes completa

A continuación se incluye el informe generado por el Sistema de Análisis de Programas en el que se representan los 10 avisos más frecuentes para cada uno de los grupos de proyectos analizados. El formato de este informe está descrito en el apartado 7.9.2. Descripción del informe de estadísticas.

RESULTADOS DE LAS ESTADÍSTICAS

1 - Estadística de errores frecuentes

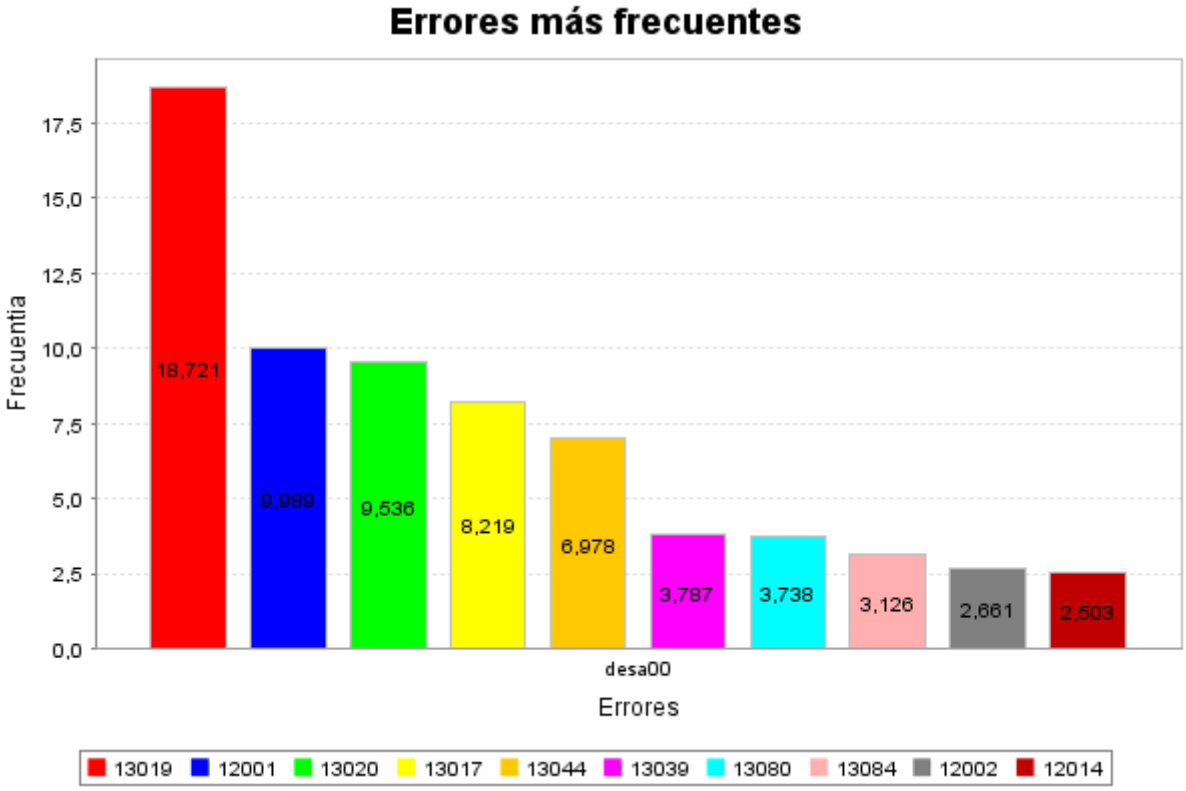
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24000 y acaba en la compilación 24182

Conjunto de Errores 1

Contenido : Todos los errores

Resultados para el usuario desa00



Código de Error	13019	12001	13020	13017	13044	13039	13080	13084	12002	12014
Frecuencia	18,721	9,989	9,536	8,219	6,978	3,787	3,738	3,126	2,661	2,503

Número de compilaciones : 183

2 - Estadística de errores frecuentes

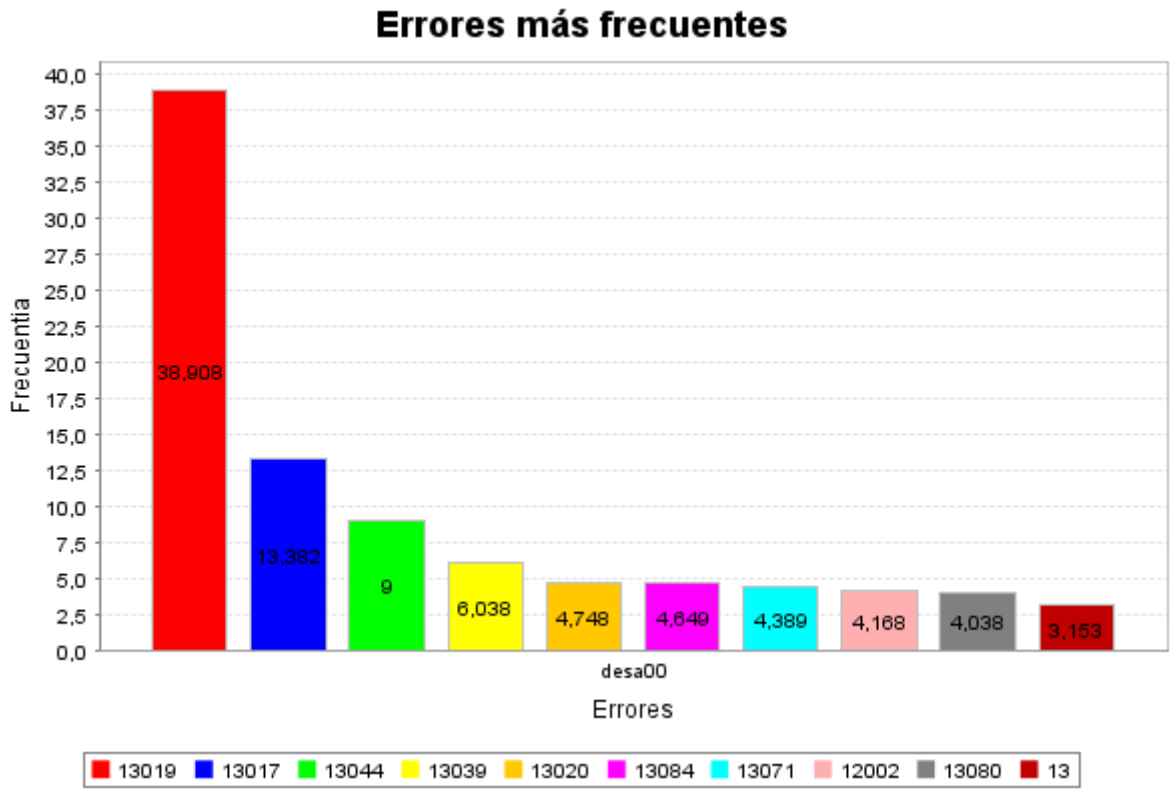
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24200 y acaba en la compilación 24330

Conjunto de Errores 1

Contenido : Todos los errores

Resultados para el usuario desa00



Código de Error	13019	13017	13044	13039	13020	13084	13071	12002	13080	13
Frecuencia	38,908	13,382	9,00	6,038	4,748	4,649	4,389	4,168	4,038	3,153

Número de compilaciones : 131

3 - Estadística de errores frecuentes

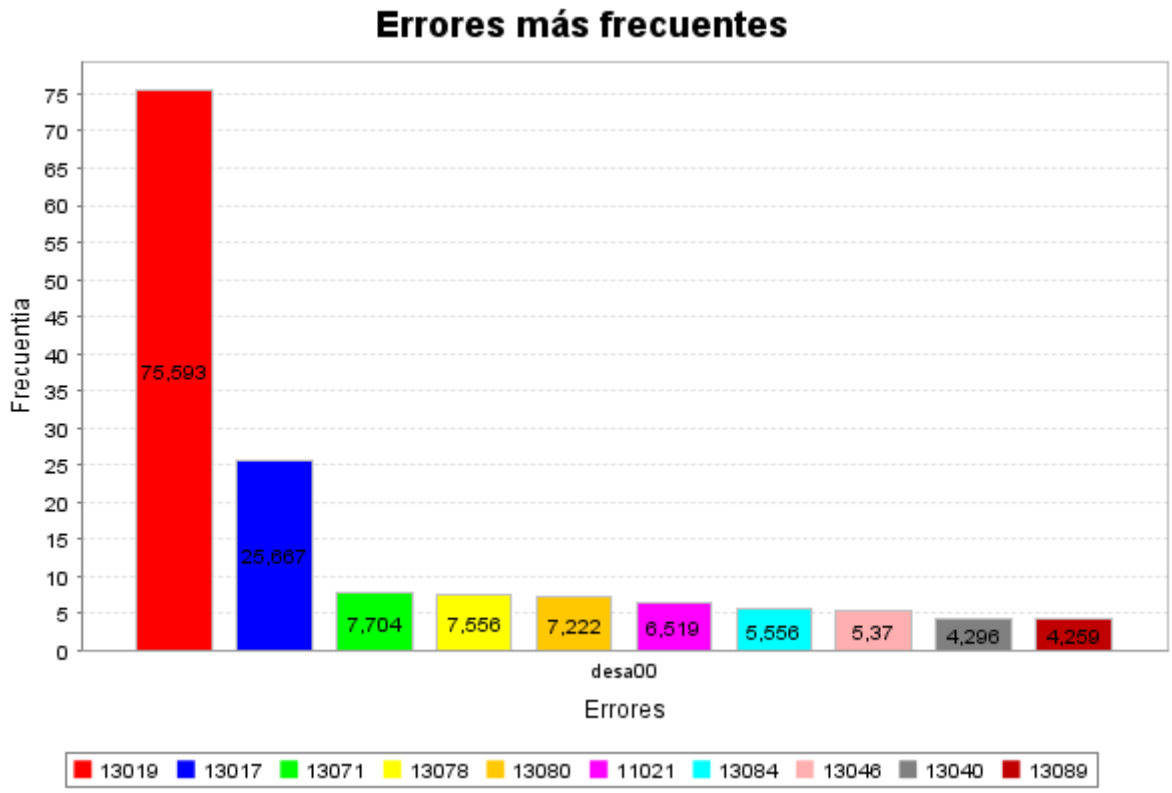
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24400 y acaba en la compilación 24426

Conjunto de Errores 1

Contenido : Todos los errores

Resultados para el usuario desa00



Código de Error	13019	13017	13071	13078	13080	11021	13084	13046	13040	13089
Frecuencia	75,593	25,667	7,704	7,556	7,222	6,519	5,556	5,37	4,296	4,259

Número de compilaciones : 27

4 - Estadística de errores frecuentes

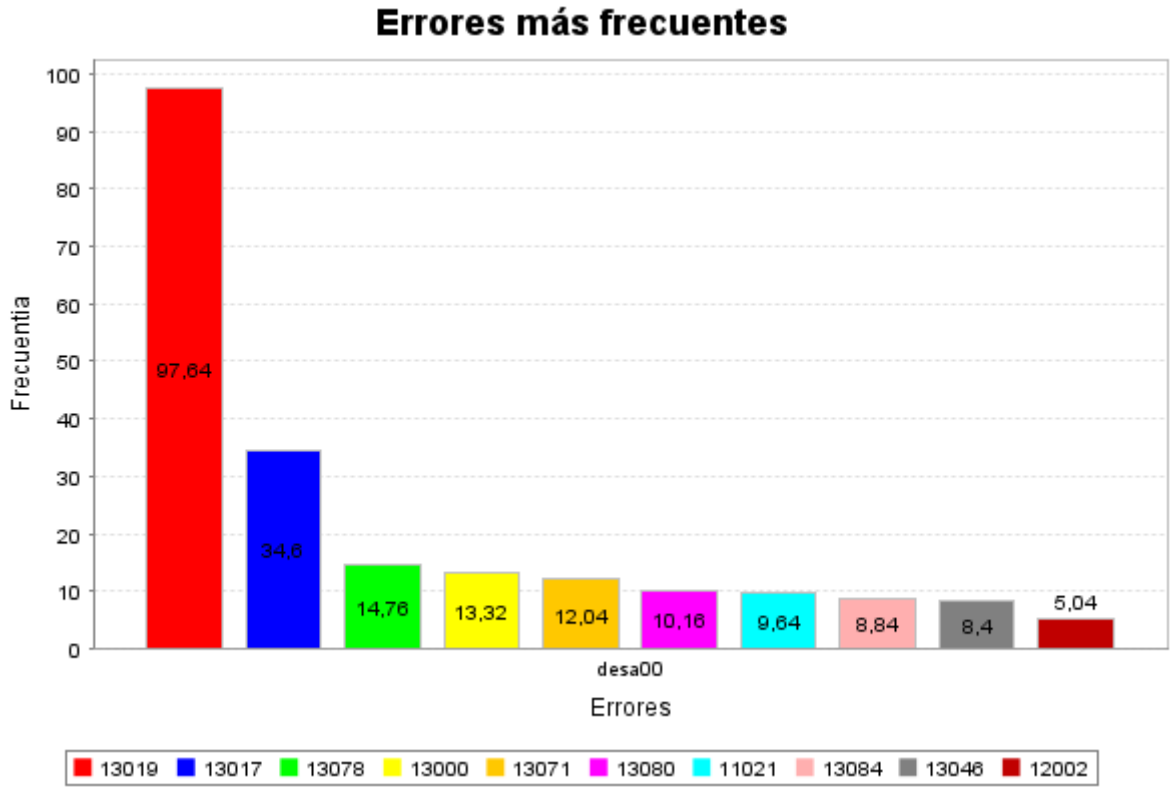
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24500 y acaba en la compilación 24524

Conjunto de Errores 1

Contenido : Todos los errores

Resultados para el usuario desa00



Código de Error	13019	13017	13078	13000	13071	13080	11021	13084	13046	12002
Frecuencia	97,64	34,60	14,76	13,32	12,04	10,16	9,64	8,84	8,40	5,04

Número de compilaciones : 25

5 - Estadística de errores frecuentes

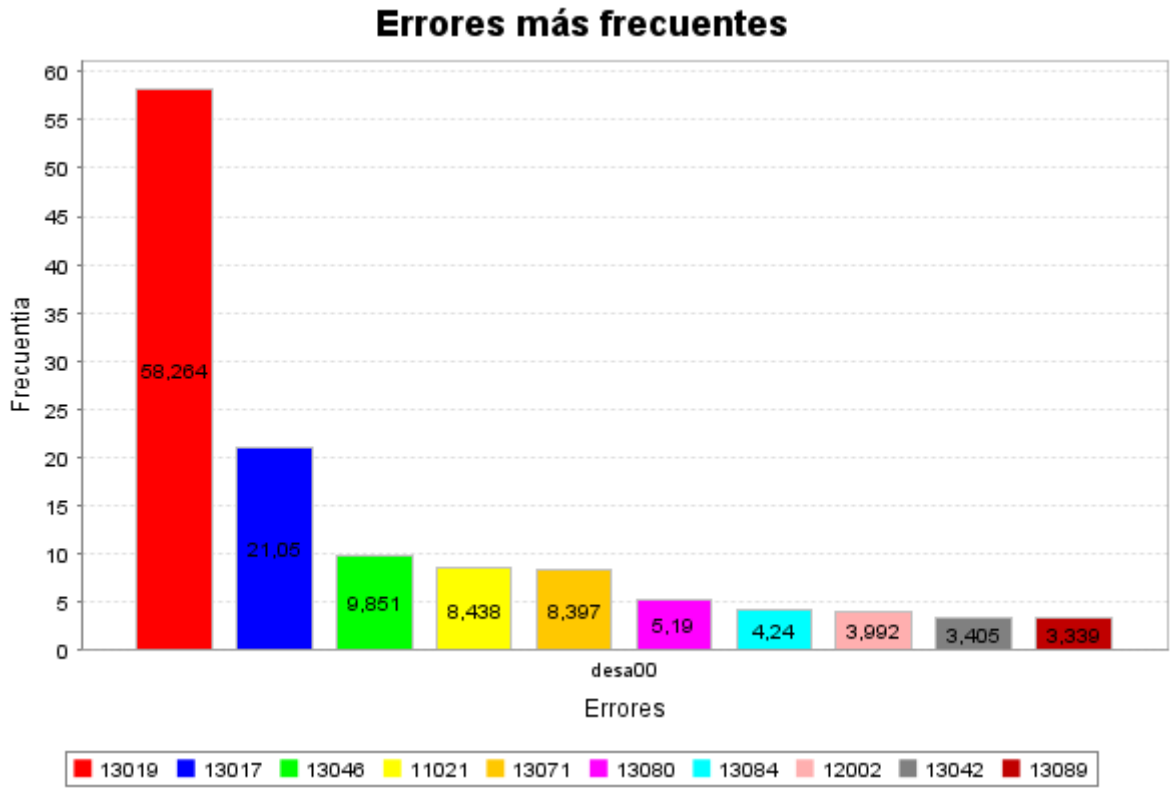
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24600 y acaba en la compilación 24720

Conjunto de Errores 1

Contenido : Todos los errores

Resultados para el usuario desa00



Código de Error	13019	13017	13046	11021	13071	13080	13084	12002	13042	13089
Frecuencia	58,264	21,05	9,851	8,438	8,397	5,19	4,24	3,992	3,405	3,339

Número de compilaciones : 121

6 - Estadística de errores frecuentes

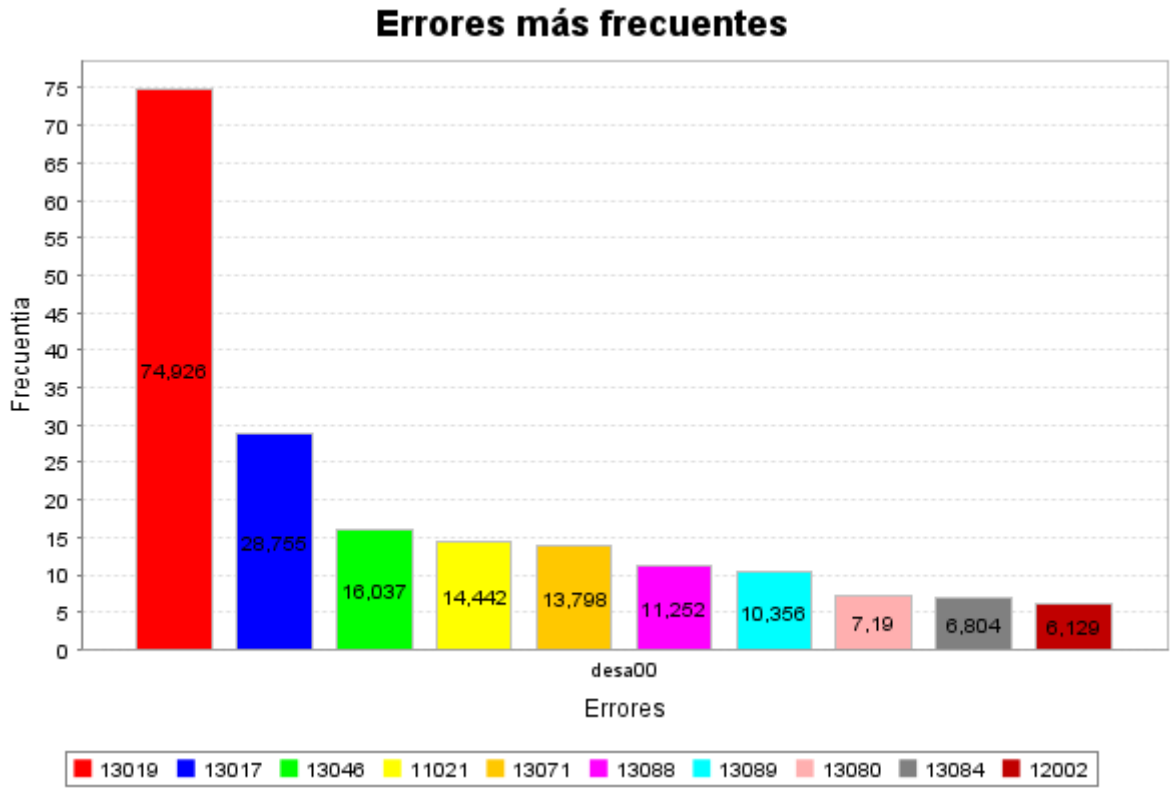
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24800 y acaba en la compilación 24963

Conjunto de Errores 1

Contenido : Todos los errores

Resultados para el usuario desa00



Código de Error	13019	13017	13046	11021	13071	13088	13089	13080	13084	12002
Frecuencia	74,926	28,755	16,037	14,442	13,798	11,252	10,356	7,19	6,804	6,129

Número de compilaciones : 163

A.12 Página errores frecuentes excluyendo avisos de incumplimiento de convenios de nombres y de código

A continuación se incluye el informe generado por el Sistema de Análisis de Programas en el que se representan los 10 avisos más frecuentes para cada uno de los grupos de proyectos analizados excluyendo aquellos avisos relacionados con incumplimiento de convenios para los nombres de paquetes, clases, métodos y variables; y convenios para la escritura del código fuente: indentación, uso de llaves, etc. Los avisos excluidos se incluyen en la Tabla 33. El formato de este informe está descrito en el apartado 7.9.2. Descripción del informe de estadísticas.

Tabla 33. Avisos eliminados en el análisis del apartado A.12

Código error	Descripción del error	Tipo de error
13019	Avoid using "if...else" statements without curly braces	Convenciones de código
13017	Avoid using "if" statements without curly braces	Convenciones de código
13071	Method name does not begin with a lower case character.	Convenciones de nombres
13020	Avoid using "for" statements without curly braces	Convenciones de código
13089	Method names should not contain underscores	Convenciones de nombres
13088	Variables should start with a lowercase character	Convenciones de nombres
13087	Variables that are not final should not contain underscores.	Convenciones de nombres
13091	Class names should not contain underscores	Convenciones de nombres
13073	Abstract classes should be named AbstractXXX"	Convenciones de nombres

RESULTADOS DE LAS ESTADÍSTICAS

1 - Estadística de errores frecuentes

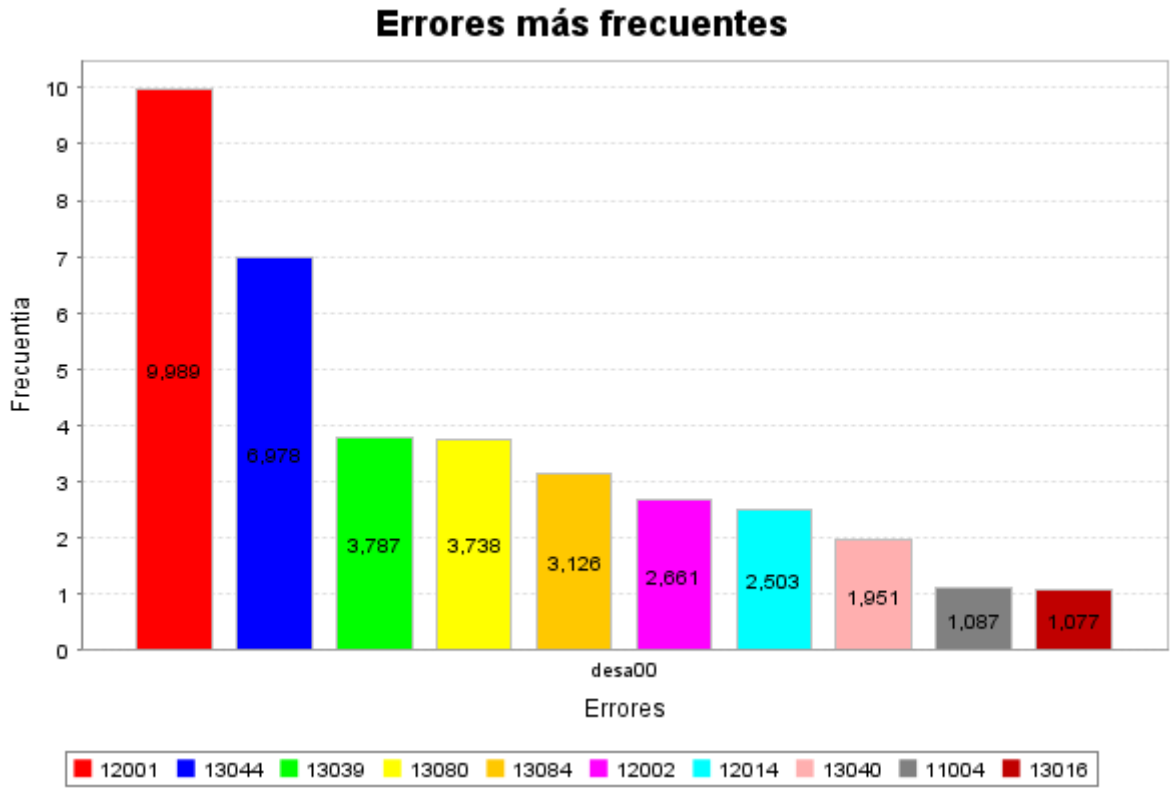
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24000 y acaba en la compilación 24182

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su código sea 13019 y que no cumplen que su código sea 13020 y que no cumplen que su código sea 13017 y que no cumplen que su código sea 13071 y que no cumplen que su código sea 13089 y que no cumplen que su código sea 13088

Resultados para el usuario desa00



Código de Error	12001	13044	13039	13080	13084	12002	12014	13040	11004	13016
Frecuencia	9,989	6,978	3,787	3,738	3,126	2,661	2,503	1,951	1,087	1,077

Número de compilaciones : 183

2 - Estadística de errores frecuentes

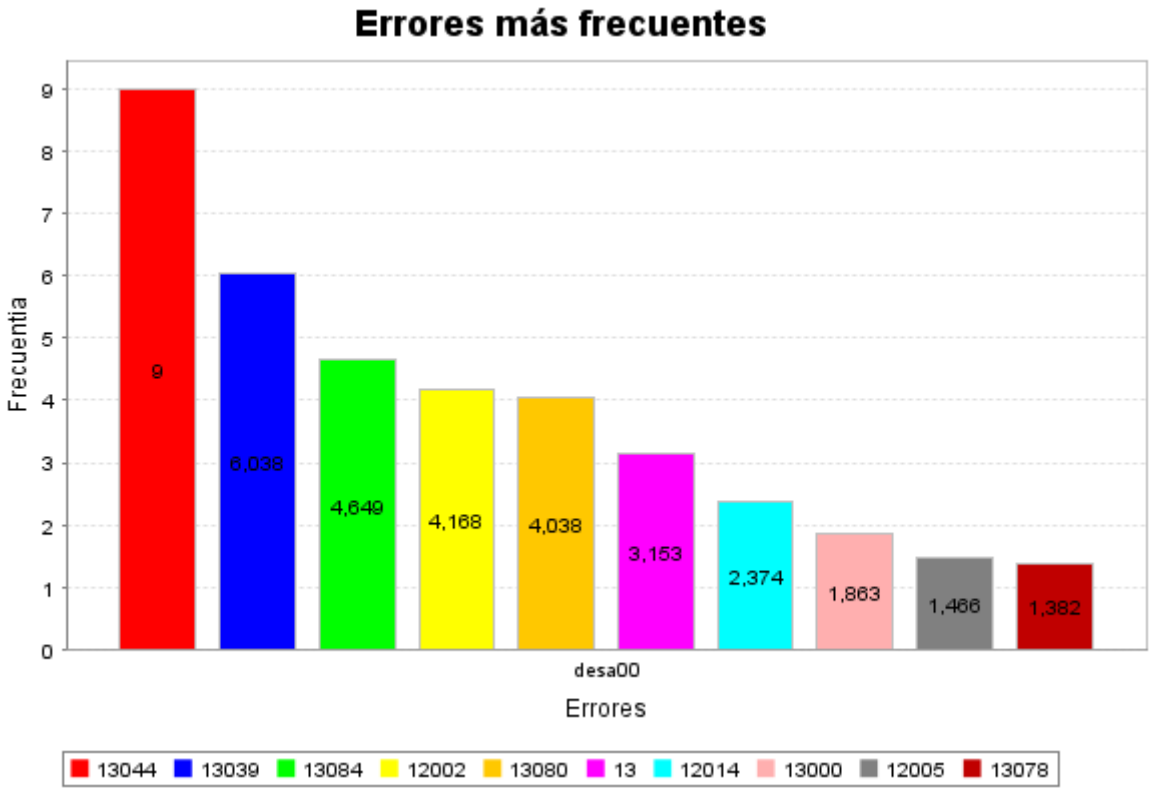
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24200 y acaba en la compilación 24330

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su código sea 13019 y que no cumplen que su código sea 13020 y que no cumplen que su código sea 13017 y que no cumplen que su código sea 13071 y que no cumplen que su código sea 13089 y que no cumplen que su código sea 13088

Resultados para el usuario desa00



Código de Error	13044	13039	13084	12002	13080	13	12014	13000	12005	13078
Frecuencia	9,00	6,038	4,649	4,168	4,038	3,153	2,374	1,863	1,466	1,382

Número de compilaciones : 131

3 - Estadística de errores frecuentes

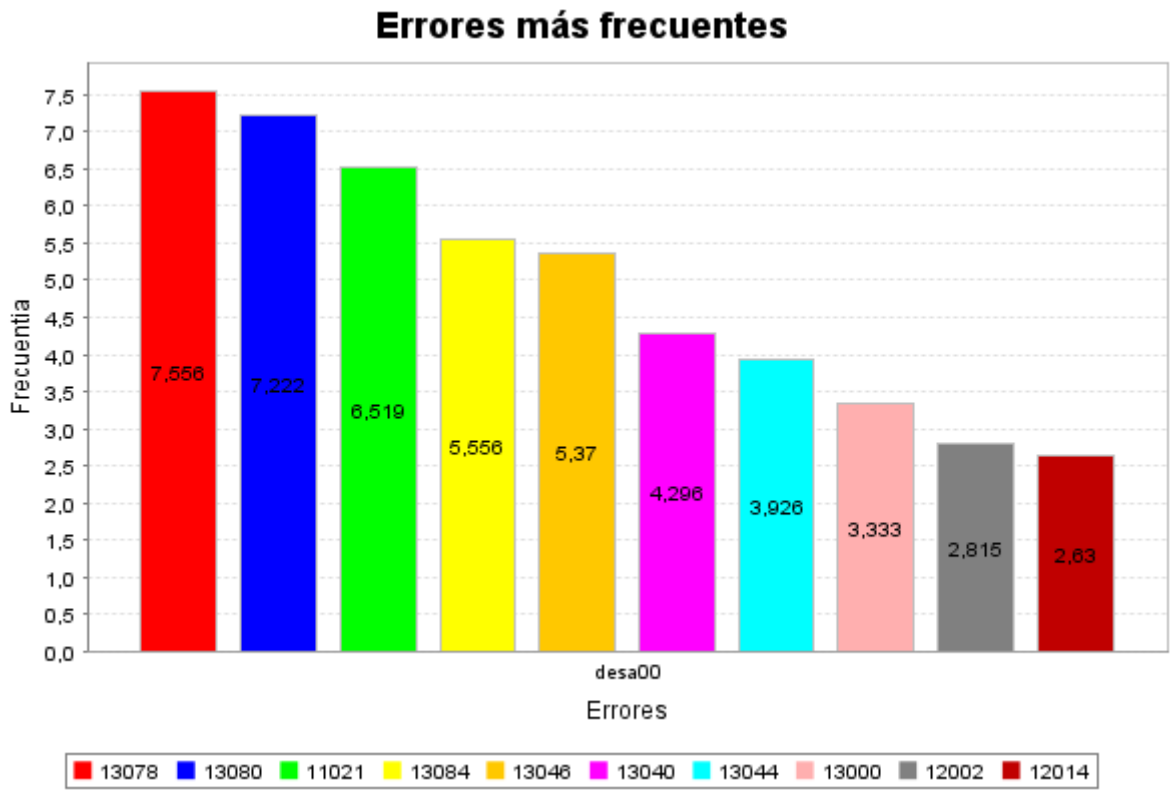
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24400 y acaba en la compilación 24426

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su código sea 13019 y que no cumplen que su código sea 13020 y que no cumplen que su código sea 13017 y que no cumplen que su código sea 13071 y que no cumplen que su código sea 13089 y que no cumplen que su código sea 13088

Resultados para el usuario desa00



Código de Error	13078	13080	11021	13084	13046	13040	13044	13000	12002	12014
Frecuencia	7,556	7,222	6,519	5,556	5,37	4,296	3,926	3,333	2,815	2,63

Número de compilaciones : 27

4 - Estadística de errores frecuentes

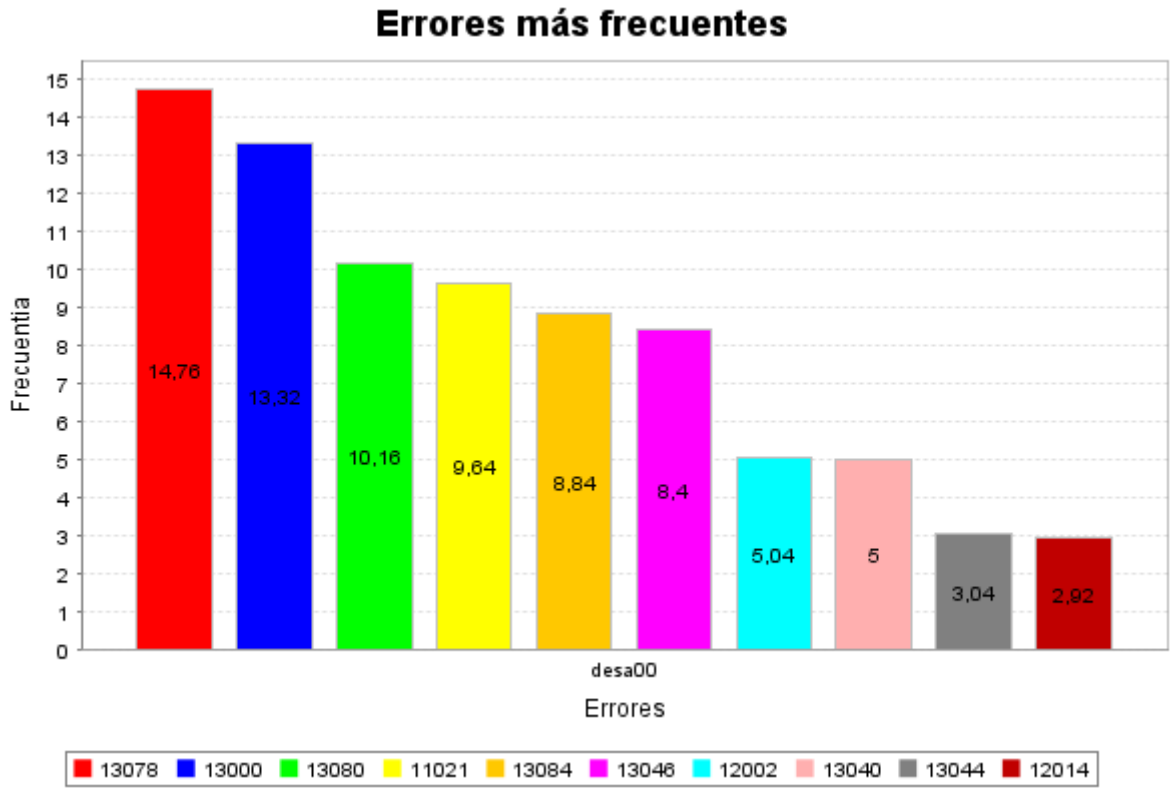
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24500 y acaba en la compilación 24524

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su código sea 13019 y que no cumplen que su código sea 13020 y que no cumplen que su código sea 13017 y que no cumplen que su código sea 13071 y que no cumplen que su código sea 13089 y que no cumplen que su código sea 13088

Resultados para el usuario desa00



Código de Error	13078	13000	13080	11021	13084	13046	12002	13040	13044	12014
Frecuencia	14,76	13,32	10,16	9,64	8,84	8,40	5,04	5,00	3,04	2,92

Número de compilaciones : 25

5 - Estadística de errores frecuentes

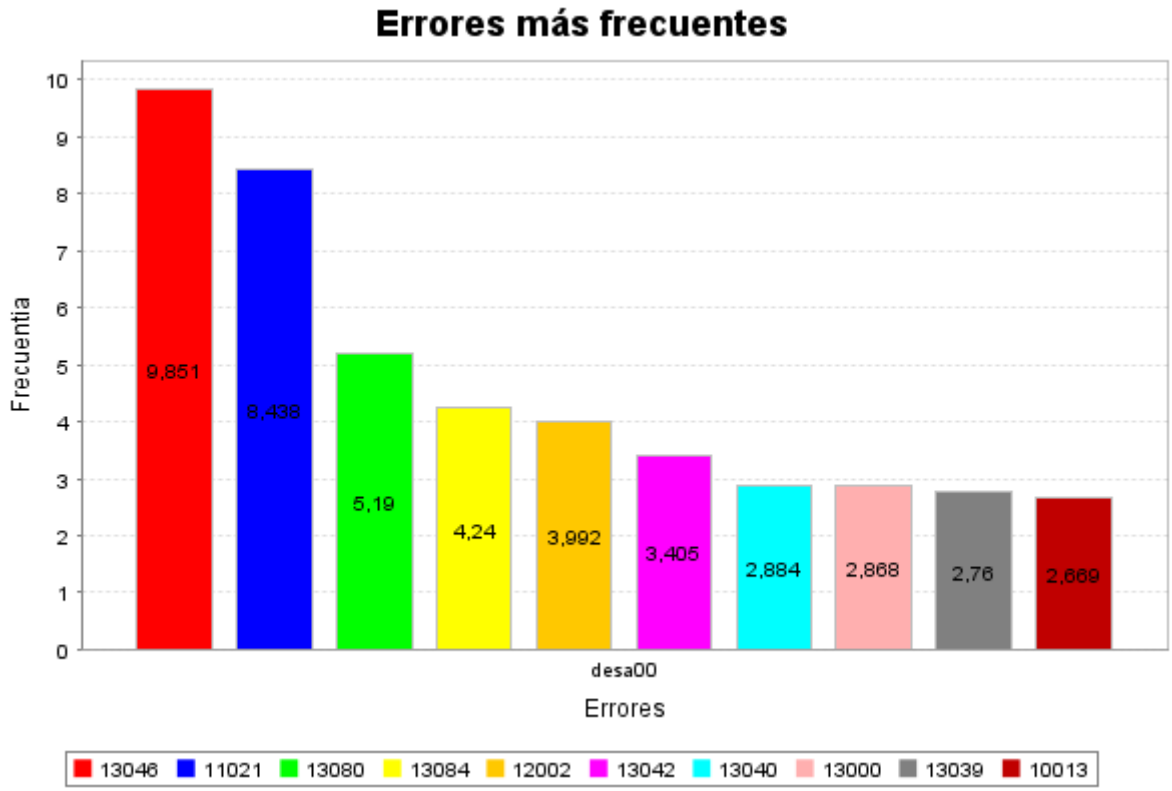
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24600 y acaba en la compilación 24720

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su código sea 13019 y que no cumplen que su código sea 13020 y que no cumplen que su código sea 13017 y que no cumplen que su código sea 13071 y que no cumplen que su código sea 13089 y que no cumplen que su código sea 13088

Resultados para el usuario desa00



Código de Error	13046	11021	13080	13084	12002	13042	13040	13000	13039	10013
Frecuencia	9,851	8,438	5,19	4,24	3,992	3,405	2,884	2,868	2,76	2,669

Número de compilaciones : 121

6 - Estadística de errores frecuentes

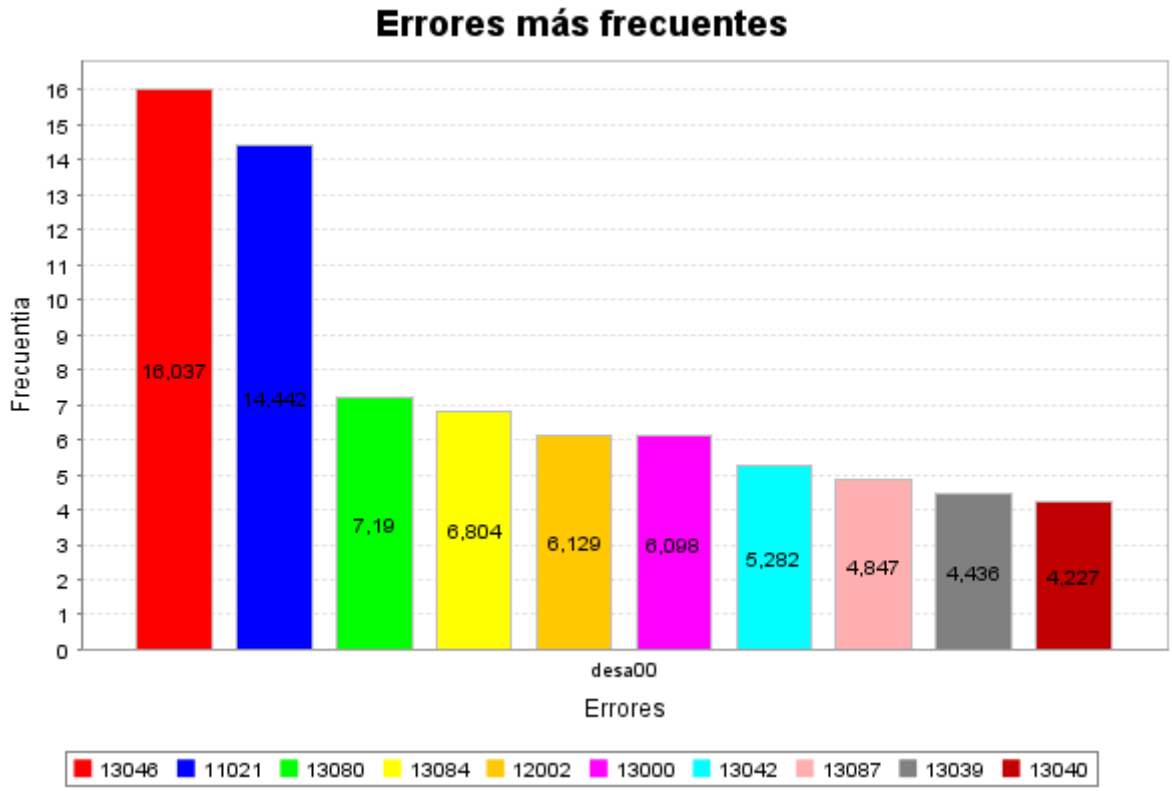
Aspectos generales de la estadística

Periodo : Empieza en la compilación 24800 y acaba en la compilación 24963

Conjunto de Errores 1

Contenido : Todos los errores que no cumplen que su código sea 13019 y que no cumplen que su código sea 13020 y que no cumplen que su código sea 13017 y que no cumplen que su código sea 13071 y que no cumplen que su código sea 13089 y que no cumplen que su código sea 13088

Resultados para el usuario desa00



Código de Error	13046	11021	13080	13084	12002	13000	13042	13087	13039	13040
Frecuencia	16,037	14,442	7,19	6,804	6,129	6,098	5,282	4,847	4,436	4,227

Número de compilaciones : 163

Apéndice D. Referencias

[Allen 2002] Allen, Eric; (2002) *Bug Patterns in Java*. Apress.

[ANECA 2005] ANECA; (2005) *Libro blanco del Título de grado en Ingeniería Informática*. (http://www.aneca.es/modal_eval/docs/libroblanco_informatica.pdf)

[Ant 2003] Apache Ant. The Apache Software Foundation. 2003. (http://ant.apache.org/)

[Arnold 1996] Arnold, Ken; Gosling, James (1996) The Java™ Programming Language. Addison Wesley.

[Artho 2001] ARTHO, Cyrille; (2001) *Finding faults in multi-threaded programs*. Master's thesis report.

[Barros 1999] Barros Blanco, Beatriz (1999) Aprendizaje Colaborativo En Enseñanza A Distancia: Entorno Genérico Para Configurar, Realizar Y Analizar Actividades En Grupo. Tesis doctoral Departamento de Inteligencia Artificial. Universidad Politécnica De Madrid.

[Barros 2000] B.Barros, M.F.Verdejo. "DEGREE: un sistema para la realización y evaluación de experiencias de aprendizaje colaborativo en enseñanza a distancia". (2000) Revista Iberoamericana de Inteligencia Artificial Vol 9 ISSN 1137-3601. Pages 27-37.

[Barros 2001] BARROS, Beatriz; Verdejo, M. F.; (2001) *Entornos para la realización de actividades de aprendizaje colaborativo a distancia*. Revista Iberoamericana de Inteligencia Artificial. ISSN 1137-3601.

[BCEL 2004] The Byte Code Engineering Library, Apache Software Foundation, 2004. (http://jakarta.apache.org/bcel/)

[Beck 2000] Beck, Ken. Extreme Programming Explained. Embrace Change. Pearson Education, Inc. Publishing as Addison Wesley Logran, 2000.

[Benford 1994] Benford, S. D., Burke, E. K., Foxley, E., Gutteridge, N., & Zin, A. M. (1994). Ceilidh as a Course Management Support System. Journal of Educational Technology Systems, 22, 235-250.

[Bergin 1997] Bergin, Joseph, Stehlik, M., Roberts, J., and Pattis, R., 1997: *Karel++ - A Gentle Introduction to the Art of Object-Oriented Programming*, John Wiley & Sons. (http://www.csis.pace.edu/~bergin/karel.html)

[Bergin 2000] Bergin, Joseph, 2000: "Introducing Objects with Karel J. Robot", Procs. of the Workshop "Tools and Environments for Understanding Object-Oriented Concepts", European Conference on Object-Oriented Programming.

[Bergin 2003] Bergin, Joseph, Stehlik, M., Roberts, J., and Pattis, R., 2003: *Karel J. Robot – A Gentle Introduction to the Art of Object-Oriented Programming in Java.* http://csis.pace.edu/%7Ebergin/KarelJava2ed/Karel++JavaEdition.html

[Boder 1992] Boder, A. "The process of knowledge reification in human-human interaction" Journal of Computer Assisted Learning, Vol. 8, No. 3, September, pp. 177-185. (1992)

[Boroni 2001] Boroni, C.M., Goosey, F.W., Grinder, M.T. and Ross, R.J., 2001: "Engaging Students with Active Learning Resources: Hypertextbooks for the Web", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[Boulay 1989] du Boulay, B., Some Difficulties Of Learning To Program, In Studying The Novice Programmer, Soloway, E., Sprohrer, J. (Eds.), Lawrence Erlbaum Associates, 1989, pp. 283-300.

[Boyle 1994] Boyle, T., Gray, J., Wendl, B., & Davies, M. (1994). Taking the plunge with CLEM: the design and evaluation of a large scale CAL system. Computers and Education, 22, 19-26.

[Bravo 2004] Bravo, Crescencio, Redondo, M.A., Ortega, M.; (2004) Aprendizaje en grupo de la programación mediante técnicas de colaboración distribuida en tiempo real. Actas del V Congreso Interacción Persona Ordenador, Lleida.

[Brown 1983] Brown, J. S. "Process versus product: a perspective on tools for communal and informal electronic learning", Report from the Learning: Education in the Electronic Age. (1983)

[Brusilovsky 1996] Brusilovsky, Peter, and Weber, G., 1996: "Collaborative example selection in an intelligent example-based programming environment", Procs. of the International Conference on Learning Sciences.

[Brusilovsky 1998] Brusilovsky, Peter, Calabrese, E., Hvorecky, J., Kouchnirenko, A. & Miller P., (1998) Mini-languages: a way to learn programming principles, Education and Information Technologies, 2, pp. 65 -83.

[Brusilovsky 2001] Brusilovsky, Peter, 2001: "WebEx: Learning from Examples in a Programming Course", Procs. of the World Conference of the WWW and Internet.

[Buck 2001] Buck, D., and Stucki, D.J., 2001: "JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum", *Procs. of th 32nd SIGCSE Technical Symposium on Computer Science Education.*

[Cavaness 2004] Chuck Cavaness 2004. Programming Jakarta Struts, 2nd Edition. O'Reilly Media, Inc. ISBN: 0596006519

[Cederqvist 1993] Per Cederqvist et al (1993) Version Management with CVS for cvs 1.11.2. Signum Support AB. (https://www.cvshome.org/docs/manual/)

[Champbell 2001] David A. Champbell y Tyler Jewwll (2001) Java Web Services. O'Reilly.

[Checkstyle] Página principal proyecto Checkstyle: http://checkstyle.sourceforge.net/

[Collis 1997] Collis, B. & Smith, C. "Desktop multimedia environments to support collaborative distance learning", Instructional Science, Vol. 25, nº 6, November, pp. 433-462. (1997)

[Cooper 2003] Cooper, S., Dann, W., and Pausch, R., 2003: "Teaching Objects-First in Introductory Computer Science", aceptado para 34th SIGCSE Technical Symposium on Computer Science Education (http://db.grinnel.edu/sigcse/sigcse2003/programAtaGlance.asp)

[Coward 2001] Coward, Danny (2001) Java™ Servlet Specification Version 2.3. Sun Microsystems, Inc. (http://jcp.org/aboutJava/communityprocess/first/jsr053/index.html)

[Crescenzi 2000] Crescenzi, P., Demetrescu, C., Finocchi, I. and Petrschi, R., 2000: "Reversible execution and visualization of programs with Leonardo", *Journal of Visual Languages and Computing*, 11(2).

[Dann 2000] Dann, W., Cooper, S., and Pausch, R., 2000: "Making the Connection: Programming with Animated Small World", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education.*

[Dawson-Howe 1996] Dawson-Howe, Kenneth M.; (1996) *Automatic submission and administration of programming assignments.* SIGCSE Bull. 28, 2 (Jun. 1996) 40-42.

[Demetrescu 2000] Demetrescu, C. and Finocchi, I., 2000: "Smooth animation of algorithms in a declarative framework", *Journal of Visual Languages and Computing*, 12(3).

[Diaz 1996] Díaz, P., Catenazzi, N. and Aedo, I. (1996) De la multimedia a la hipermedia. Ra-Ma.

[DoctorJ] DoctorJ sitio Web: http://www.incava.org/projects/doctorj/

[DOM 1998] Document Object Model (DOM) Level 1 Specification. Version 1.0. W3C Recommendation 1 October, 1998

[Faries 1988] Faries, J.M., and Reiser, B.J., 1988: "Access an d use of previous solutions in a problem solving situation", *Procs. Of the 10th Annual Conference of th Coginitive Science Society*.

[Fernández 2003] Fernández González, Raúl; Paule Ruíz, María del Puerto; Pérez Pérez, Juan Ramón; González Rodríguez, Martín; González Gallego, Marcos (2003): Adaptable Contents Visulization (VIC). ICWE 2003: 167-170

[Flanagan 2002] C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata. Extended static checking for Java. In Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 234–245, Berlin, Germany, June 2002.

[Foster 2002] J. S. Foster, T. Terauchi, and A. Aiken. Flow-sensitive type qualifiers. In Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 1–12, Berlin, Germany, June 2

[Fowler 1993] Fowler, W. A. L., & Fowler, R. H. (1993). A hypertext-based approach to computer science education unifying programming principles. In T. Ottmann & I. Tomek (Eds.), Proceedings of ED-MEDIA'93 - World conference on educational multimedia and hypermedia. Charlottesville, VA: AACE. 203-209.

[Geis 1999] Jennifer Geis; (1999) JavaWizard User Guide. Collaborative Software Development Laboratory. Department of Information and Computer Sciences. University of Hawai. (http://csdl.ics.hawaii.edu/techreports/98-15/98-15.html)

[Gomez 2003] Gómez Albarrán, Mercedes; (2003) Una revisión de métodos pedagógicos innovadores para la enseñanza de la programación. Proc. Jornadas de Enseñanza Universitaria de la Informática (JENUI'03).

[González 2002] González Rodríguez, Martín; López Pérez, Benjamin; Paule Ruíz, María del Puerto; Pérez Pérez, Juan Ramón (2002): Dynamic Generation of Interactive Dialogs Based on Intelligent Agents. AH 2002: 564-567

[González 2004] González Rodríguez, Martín; Pérez Pérez, Juan Ramón; Paule Ruíz, María del Puerto (2004): Designing User Interfaces Tailored to the Current User's Requirements in Real Time. ICCHP 2004: 69-75

[Grindstaff 2004] Grindstaff, Chris; (2004) FindBugs, Part 2: Writing custom detectors. DeveloperWorks. IBM's resource for developers. (http://www-128.ibm.com/developerworks/java/library/j-findbug2/)

[Guzdial 1995] Guzdial, M., 1995: "Software-realized scaffolding to facilitate programming for science learning", *Interactive Learning Environments*, 4(1).

[Hitz 1997] Hitz, M. and Kögeler, S. Teaching C++ on the WWW. In Proc. ITiCSE'97, ACM Press, 11-13.

[Hovemeyer 2004a] Hovemeyer, David H.; (2004) FindBugs *Manual*. University of Maryland (http://findbugs.sourceforge.net/index.html).

[Hovemeyer, 2004b] Hovemeyer, David H.; Pugh, William; (2004). *Finding Bugs is Easy*. 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004).Vancouver, British Columbia, Canada.

[Huizinga 2001] Huizinga, Dorota M.; (2001) Identifying Topics for Instructional Improvement Through On-line Tracking of Programming Assignments. SIGCSE Bulletin 33, 3 (Sep. 2001) 129-132.

[Humphrey 1989] Humphrey, Watts S.; (1989) *Managing the Software Process*. Reading, MA: Addison-Wesley.

[Humphrey 1997] Humphrey, Watts S.; (1997) Introduction to the Personal Software Process, First Edition. 1ª edición. Pearson Education.

[Hundhausen 2002] Hundhausen, C., Douglas, S., and Stasko, J., 2002: "A Meta-Study of Algorithm Visualization Effectiveness", *Journal of Visual Languages and Computing*, 13(3)

[Jacobson 2002] Jacobson, Ivar. Call for expert systems. Application Development Trends Magazine. Número de junio de 2002. (http://www.adtmag.com)

[JCSC] Página principal del proyecto JCSC. http://jcsc.sourceforge.net

[JHA] JHAVÉ. http://csf11.acs.uwosh.edu/

[Johnson 1998] JOHNSON, P.M. (1998) Reengineering Inspection: The Future of Formal Technical Review. Communications of the ACM, vol. 41, pp. 49-52.

[Jones 1998] Jones, Capers; (1997) *The Impact of Poor Quality and Canceled Project son the Software Labor Shortage*. Informe técnico, Software Productivity Research, Inc.

[JUnit 2002] Kent Beck, Erich Gamma. (2002) JUnit Cookbook. JUnit 3.8. (http://junit.sourceforge.net/doc/cookbook/cookbook.htm)

[Knizhnik 2003] KNIZHNIK, Konstantin y Artho, Cyrille; (2003) Jlint manual. Java Program Checker (http://artho.com/jlint/).

[Kölling 1999a] Kölling, M. (1999a). The Problem of Teaching Object-Oriented Programming, Part 2: Environments. Journal of Object-Oriented Programming, 11(9), 6-12.

[Kölling 1999b] Kölling, M. (1999b). Teaching Object Orientation with the Blue Environment. Journal of Object-Oriented Programming, 12(2), 14-23.

[Kölling 1999c] Kölling, Michael (1999). The design of an object-oriented environment and language for teaching. Dissertation of DPh. Department of Computer Science, University of Sydney

[Kölling, 2003] Kölling, M., Quig, B., Patterson, A. and Rosenberg, J., (2003) The BlueJ system and its pedagogy, Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, Vol 13, No 4, Dec 2003.

[Koschmann 1996] Koschmann, T. (Editor) "CSCL: Theory and Practice of an emerging paradigm". Lawrence Erlbaum (1996).

[Labra 2003] Labra Gayo, José Emilio; Morales Gil, J. M.; Fernández Álvarez, A. M.; Sagastegui Chigne, H. (2003): A generic e-learning multiparadigm programming language system: IDEFIX project. SIGCSE 2003: 391-395

[LEO] Leonardo. http://www.cc.gatech.edu/gvu/softviz/algoanim/

[Leuf 2001] Bo Leuf, Ward Cunningham. (2001) The Wiki Way: Collaboration and Sharing on the Internet. Addison-Wesley.

[Luján-Mora 2003] Luján-Mora, Sergio; (2003) Un enfoque para la enseñanza de la depuración de errores en las asignaturas de programación. Actas de IX Jornadas de

Enseñanza Universitaria de la Informática. Jenui 2003, pp 473-480. Thomson Paraninfo, S.A., Madrid, 2003.

[Martínez-Unanue 2002] R. Martínez-Unanue, M. Paredes-Velasco, C. Pareja-Flores, J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide. (2002) Electronic books for programming education: A review and future prospects. 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2002)

[MEC 2003] Ministerio de Educación y Ciencia. (2003) Aplicación de Nuevas Tecnologías en el ámbito universitario. (http://wwwn.mec.es/univ/jsp/plantilla.jsp?id=2149)

[Meyerowitz 1995] Meyerowitz, J. (1995). PasTIL: a multiple-environment interactive Pascal tutor. In H. Maurer (Ed.), Proceedings of ED-MEDIA'95 - World conference on educational multimedia and hypermedia. Charlottesville, VA: AACE. 786.

[Naps 2000] Naps, T., Eagan, J., and Norton, L., 2000: "JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualization", *Procs. of th 31st SIGCSE Technical Symposium on Computer Science Education.*

[Neal 1989] Neal, L.R., 1989: "A system for example-based programming", *Procs. of Human Factors in Computing Systems.*

[Nielsen 1999] Nielsen, Jakob (1999) Designing Web Usability : The Practice of Simplicity. New Riders Press.

[Nosek 1998] NOSEK, J.T. (1998) The Case for Collaborative Programming. Communications of the ACM, Vol. 41, No. 3 (March), pp. 105-108.

[Pattis 1981] Pattis, R., 1981: *Karel the Robot*, John Wiley & Sons.

[Paule 2003] Paule Ruíz, María del Puerto; Ocio Barriales, Sergio; Pérez Pérez, Juan Ramón; González Rodríguez, Martín (2003): Feijoo.net: An Approach to Personalized E-learning Using Learning Styles. ICWE 2003: 112-115

[Pelegrí-Llopart 2001] Pelegrí-Llopart, Eduardo (editor) (2001) JavaServer Pages™ Specification Version 1.2. Sun Microsystems, Inc. (http://www.jcp.org/aboutJava/communityprocess/final/jsr053/)

[Pérez 2003a] Pérez Pérez, Juan Ramón; Paule Ruíz, María del Puerto; González Rodríguez, Martín. "Development Web Environment" for Learing Programming Languages. ICWE 2003, LNCS 2722, pp. 128-129, 2003. Springer, Berlin 2003.

[Pérez 2003b] Pérez Pérez, Juan Ramón; Paule Ruiz, Mª del Puerto; González Rodríguez, Martín. Entorno web de desarrollo para el aprendizaje de paradigmas de programación. Actas de IX Jornadas de Enseñanza Universitaria de la Informática. Jenui 2003, pp 465-471. Thomson Paraninfo, S.A., Madrid, 2003.

[Pérez 2004a] Pérez Pérez, Juan Ramón; Gonzalez Rodríguez, Martín; Paule Ruiz, María del Puerto. A development environment for cooperative programming. En Actas del V Congreso Interacción Persona Ordenador. Interacción 2004. Seminario de Doctorado. Mayo de 2004.

[Pérez 2004b] Pérez Pérez, Juan Ramón; Iglesias Suárez, Mª Cristina; Paule Ruiz, Mª del Puerto; (2004) SACODE: Sistema de Aprendizaje Colaborativo de la Programación. VI Simposio Internacional de Informática Educativa (SIIE 2004).

[Pérez 2004c] Pérez Pérez, Juan Ramón; Rodríguez Fernández, Daniel; González Rodríguez, Martín; (2004) ¿Es posible la eliminación de los errores de los programas? VI Simposio Internacional de Informática Educativa (SIIE 2004).

[phpwiki] Página de PhpWiki: http://phpwiki.sourceforge.net/

[PMD] Página del proyecto PMD: http://pmd.sourceforge.net/.

[Pressman 1997] Pressman, Roger S.; (1997) *Software Engineering: A Practitioner's approach*. Cuarta edición. McGraw-Hill.

[Redondo 2002] Redondo, M.A., Bravo, C., Ortega, M. & Verdejo, M.F.: PlanEdit: An adaptive tool for design learning by problem solving. En: De Bra, P., Brusilovsky, P. & Conejo, R. (eds.) Adaptive Hypermedia and Adaptive Web-Based Systems. Springer Verlag. Lecture Notes in Computer Science. Berlin (2002) 29-31

[ROB] Página official de Robocode. http://robocode.sourceforge.net/?Open&ca=daw-prod-robocode

[Rößling 2000a] Rößling, Guido, and Freisleben, B., 2000: "Experiences in Using animations in Introductory Computer Science Lectures", *Procs. of the 31st SIGCSE Technical Symposium on Computer Science Education*.

[Rößling 2000b] Rößling, Guido, Schüler, M. and Freisleben, B., 2000: "The ANIMAL Algorithm Animation Tool", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[Rößling 2000c] Rößling, Guido. ANIMAL User Guide. University of Siegen. 2000. http://www.animal.ahrgr.de/index.php3

[Rößling 2001] Rößling, Guido, and Freisleben, B., 2001: "The Extensible Visualization Framework ANIMAL", *Procs. Of the 32nd SIGCSE Technical Symposium on Computer Science Education*.

[Rößling, 2002] Rößling, Guido, and Naps, T., 2002: "A Testbed for Pedagogical Requirements in Algorithm Visualization", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[Rubio 2003] Enrique Rubio Royo, Domingo J. Gallego, Catalina Alonso García. e-Learning en la formación a distancia y en los nuevos contextos corporativos. NOVATICA nº 165. Septiembre-octubre 2003.

[Rutar 2004] Rutar, Nick; Almazan, Christian; Foster, Jeff (2004) *A Comparison of Bug Finding Tools for Java*. The 15th IEEE International Symposium on Software Reliability Engineering (ISSRE'04). Saint-Malo, Bretagne, France.

[Sama 2003] SAMA VILLANUEVA, Sergio; PÉREZ PÉREZ, Juan Ramón; OCIO BARRIALES, Sergio; GONZÁLEZ RODRÍGUEZ, Martín; (2003) DSTool: A Reflection-based debugger for Data Structures Comprehension in Computing

Science Learning. p. 1026 - 1030. Human - Computer Interaction: Theory and Practice (Part I). Volume 1. JACKO, Julie; STEPHANIDIS, Constantine (Eds.). LEA: Lawrence Erlabaum Associates, Publishers. Mahwah, New Jersey. ISBN 0-8058-4930-0

[Sama 2005] Sama Villanueva, Sergio; (2005) DSTOOL: Herramienta para la Programación con Estructuras de Datos. Proyecto Fin de Carrera de la Escuela Politécnica Superior de Ingeniería de Gijón. Nº 1042036.

[Sapsomboon, 2000] Sapsomboon, Bordin (2000) *Shared Defect Detection : The Effects of Annotations in Asynchronous Software Inspection.* Doctoral Dissertation (http://www.sis.pitt.edu/~cascade/bordin/).

[Satratzemi 2001] Satratzemi, Maria; Dagdilelis, Vassilios; Evagelidis, Georgios (2001) A system for program visualization and problem-solving path assessment of novice programmers. ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education.

[Scotts 2003] Scotts, D., Williams, L., Nagappan, N., Baheti, P., Jen, D., Jackson, A., Virtual Teaming: Experiments and Experiences with Distributed Pair Programming, Extreme Programming/Agile Universe 2003

[Seffah 1999] Seffah, Ahmed; Bari, Moncef; Desmaris, Michel (1999) Assessing Object-Oriented Technology Skills Using an Internet-Based System. . ITiCSE '99: Proceedings of the 4th annual conference on Innovation and technology in computer science education.

[Shen 2000] Shen, H. & Sun, C.; (2000) RECIPE: a prototype for Internet-based real-time collaborative programming. Proceedings of the 2nd Annual International Workshop on Collaborative Editing Systems in conjunction with ACM CSCW Conference, December 2-6, Philadelphia, Pennsylvania, USA.

[Stage3 2005] Alice. Learn to Program Interactive 3D Graphics. Stage3 Research Group. Carnegie Mellon University. 2005. http://www.alice.org/

[Stasko 1990] Stasko, J., 1990: "TANGO: A Framework and System for Algotirm Animation", *Computer*, 23(9).

[Stasko 1998] Stasko, J., Domingue, J., Price, B.A., Brown, M.H., 1998: *Software Visualization Programming as a Multimedia Experience*, MIT Press.

[Schwarz 1996] Schwarz, E., Brusilovsky, P. and Weber, G. World-wide intelligent textbooks. In Proc. ED-TELECOM'96 (1996), AACE, 302-307.

[Verdejo 1998] Verdejo, M.F. & Barros, B. "Creating an organizational learning memory for collaborative experiences in distance education" en Teleteaching´98, pp.1035-1046. (1998) (http://sensei.ieec.uned.es/~steed/papers/verdejo-tt98.pdf)

[WebCT 2003] WebCT, Inc. (2003) *Whitepaper: Learning Without Limits*. www.webct.com.

[Wessner 1999] Wessner, M., Hans-Rüdiger, P. & Miao, Y. (1999) Using Learning Protocols to Structure Computer- Supported Cooperative Learning. Proceedings of

the ED-MEDIA'99. World Conference on Educational Multimedia, Hypermedia & Telecommunications, pp. 471-476, Seattle, Washington, June 19-24.

[Williams 2000] WILLIAMS, L.A. & Kessler, R.R. (2000) All I really need to know about pair programming learned in kindergarten. Communications of the ACM, BOl. 43, No. 5.

[Williams 2001] WILLIAMS, L. & Upchurch, R.L. (2001) In Support of Student Pair-Programming. ACM SIGCSE Conference for Computer Science Educators, February.

[Xerces 2004] Xerces. The Apache Software Foundation. 2004. (http://xml.apache.org/xerces2-j/)

[Xie 2002] Y. Xie and D. Engler. Using redundancies to find errors. In Proceedings of the ACM SIGSOFT International Symposium on the Foundations of Software Engineering, November 2002.

[XML 2004] Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004 (http://www.w3.org/TR/2004/REC-xml-20040204/)

[XML Schema 2004] XML Schema Part 0: Primer Second Edition. W3C Recommendation 28 October 2004 (http://www.w3.org/TR/xmlschema-0/)

[XTA] XTango. http://www.cc.gatech.edu/gvu/softviz/algoanim/

[Zeller 2000] Zeller, Andreas; (2000) *Making Students Read and Review Code*. 5th ACM SIGCSE/SIGCUE Annual Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland, July 2000