Guía para la entrega de trabajos en las titulaciones de Informática / Telemática

Elaborado por: Comisión de calidad del Grado en Ingeniería Informática en Tecnologías de la Información e Ingeniería Telemática

Fecha:

Introducción

El objetivo del presente documento es proporcionar una guía de cómo se deben realizar las entregas de los trabajos de las asignaturas de las titulaciones de Informática y Telemática. En él se detallará desde cómo debe ser la estructura de directorio del fichero entregable hasta cómo realizar el código fuente o cómo realizar las memorias de los trabajos.

Se trata de una guía genérica donde cada profesor, particularmente y a la hora de solicitar la entrega, indicará cuál de cada uno de los puntos especificados en esta guía deben acompañarse a la entrega.

Qué se debe entregar

Toda entrega consistirá en la entrega de un fichero comprimido .zip con la siguiente denominación:

IDPráctica_UsuarioCorreo.zip

donde

- IDPráctica: es el identificador de la práctica (P1, Práctica1, etc) tal y como el profesor la haya indicador
- UsuarioCorreo: es el identificador del correo electrónico del estudiante

IMPORTANTE: Sólo se permitan la entrega de fichero .zip (no .rar, .arj, .7z, etc).

Este fichero comprimido será una carpeta con la siguiente estructura:

1. Fichero leeme.txt: donde figure claramente la fecha de entrega, el nombre del alumno, del profesor, titulación, asignatura así como una breve descripción de la entrega y particularidades, si fuera necesario, de la misma.
2. Carpeta Código fuente: Deberá contener todo el código fuente desarrollado en la práctica
3. Carpeta Documentación
    1. Manual/Documentación de la entrega: Debe contener:
        1. Portada donde se específica, al menos, el nombre del alumno, del profesor, de la asignatura y fecha.
        2. Índice del trabajo
        3. Pruebas realizadas (si fuera necesario)
        4. Funcionamiento del programa (si fuera necesario)
        5. Conclusiones y/o valoración personal
        6. Bibliografía
    2. API de autodocumentación (javadoc,...) (si fuera necesario)
4. Carpeta Ejecutable (si fuera necesario)
5. Carpeta de Prueba (si fuera necesario)
6. Carpeta de librerías (si fuera necesario)

Normas de codificación

El presente apartado quiere hacer hincapié en la importancia de escribir programas más legibles y documentados. Las ventajas de documentar el código de forma completa y precisa son:

- Fácil mantenimiento
- Comprensión del código escrito por terceros
- Amigabilidad, Etc

Formato de líneas

- No usar más de 80 caracteres por línea (imagen de tarjeta). De esta forma se pueden visualizar las líneas completas con un editor de texto o en una hoja impresa tamaño DIN A4.
- Cuando la línea sea mayor de 80 caracteres, debe dividirse en varias partes, cada una sobre una línea. Por ejemplo:

```
someMethod(longExpression1, longExpression2, longExpression3,
             longExpression4, longExpression5);
//USE THIS INDENTATION INSTEAD
if ((condition1 && condition2)
  || (condition3 && condition4)
        ||!(condition5 && condition6)) {
      doSomethingAboutIt();
}
```

- Use líneas en blanco como elemento de separación entre bloques de código conceptualmente diferentes.
- Indentar adecuadamente cada nuevo bloque de sentencias. De esta forma se aprecia visualmente la diferencia de nivel entre bloques de sentencias.

Estructura de los ficheros fuentes

Todos los ficheros fuente deben tener la la siguiente estructura:

1. Comentario de Copyright
2. Sentencias de importación: (import, include, etc).
3. Comentario de autodocumentación de clase (nombre de clase, versión, fecha) (Java y C++)
4. Código

Normas de codificación comunes en C/C++/Java

Declaraciones

1. Deben seleccionarse identificadores significativos y a ser posible breves. En cualquier caso prefiera la claridad a la brevedad. Es preferible el identificador,
   nombreCliente en lugar de, *nombre*

2. Para clases e interfaces debe seleccionarse identificador en sustantivo. Siempre debe ir en minúscula excepto la letra inicial. Si el identificador consta de varias palabras debe colocarse juntas, con la inicial de cada una en mayúsculas. Por ejemplo
   class Persona
   class PuntosCerrados

3. Las variables y objetos deben identificase en minúsculas. En el caso que el identificador conste de varias palabras, se separan usando mayúsculas o por el carácter de subrayado. Por ejemplo,

```
double precioVenta = 10.0;
int cantidadFinal = 0;
String nombre_alumno = "Pepe";
```

4. Las constantes debe ir en mayúsculas. En caso de varias palabras separar con el carácter de subrayado. Ejemplo correctos serían,
   Java
   ```
   final int MINIMO=100;
   final double PRECISION=0.001;
   final double ALTURA_MEDIA=29.5;
   ```

5. Para los métodos, en la medida de lo posible, el identificador debe ser un verbo y debe usarse en minúsculas. Si esta formado por varias palabras, la inicial de todas las posteriores a la primera en mayúsculas. Ejemplos

   ```
   void calcular(double valor){
         - - - código- - -
   }
   float obtenerMedia(){
         - - - código - - -
   }
   ```

6. Variables:
   1. Se debe colocar cada variable en una línea, incluso siendo del mismo tipo.
   2. Deben inicializarse en su declaración
   3. Deben declararse todas al comienzo de la función o método

   ```
   //Evitar
   int altura=0, base=10, [] valores;
   //Preferido
   int altura=0;
   int base=10;
   int []valores;
   ```

Sentencias

Se utilizará el siguiente formato para las sentencias:

1. Sentencias return

   ```
   return;
   return persona.toString();
   return (area);
   ```

2. Sentecias if, if-else, if-else-if

```
if ( condicion) {
    sentencias;
}
```

```
if ( condicion) {
   sentencias;
} else {
   sentencias;
}
```

```
if ( condicion) {
    sentencias;
} else if ( condition) {
    sentencias;
} else {
   sentencias;
}
```

3. Setencia for

   ```
   for(initializacion; condicion; actualizar) {
         sentencias;
   }
   ```

4. Sentencia while

   ```
   while(condicion) {
         sentencias;
   }
   ```

5. Sentencia do-while

```
do {
       sentencias;
} while (condicion);
```

6. Sentencia switch

```
switch ( condicion) {
       case 1:
              sentencias;
              /* sigue la ejecución */
       case 2:
              sentencias;
              break;
       case 3:
              sentencias;
              break;
       default:
              sentencias;
              break;
}
```

7. Sentencia try-catch (Java)

```
try {
      sentencias;
} catch (ExceptionClass e) {
      sentencias;
} finally {
      sentencias;
}
```

Normas de codificación para Java

El presente documento es un resumen del Java Code Convention publicado por Oracle que se puede obtener de (http://www.oracle.com/technetwork/java/codeconv-138413.html)

Comentarios

Existen dos tipos de comentarios:

1. Comentarios de implementación. Tres tipos:

Comentario de Bloque	Comentario en línea	Comentario Corto

`/*`
`* Un comentario de bloque`
`*/`
	`/* comentario de línea */`	`/* comentario de línea */`

2. Comentarios de **auto-documentación**. Para generarlos se utilizará herramienta de autodocumentación como JavaDoc o doxygen. Tipos: Más información: http://java.sun.com/javadoc/writingdoccomments/index.html

Comienzo de Fichero (copyright → Opcional)

Cada nuevo fichero debe tener la siguiente estructura:

```
/*
 * @(#)Blah.java        1.82 99/03/18
 *
 * Copyright (c) 1994-1999 Sun Microsystems, Inc.
 * 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of Sun
 * Microsystems, Inc. ("Confidential Information").  You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered into
 * with Sun.
 */
```

Comienzo de Clase (en POO)

```
**
 * Class description goes here.
 *
 * @version     1.82 18 Mar 1999
 * @author      Firstname Lastname
 */
```

Declaración de métodos (en POO)

```
/**
* Explicación del método. Esta función calcula el área de un triángulo
* @param b Valor de la base del triángulo
* @param dest Valor de la altura del triángulo
* @return double Área del triangulo
* @see recomendación para el lector. Ver la función Calcular del fichero rectangulo.
*/
double Area(double b, double a);
```

Ejemplos de Código en Java

Obtenido de http://java.sun.com/docs/codeconv/html/CodeConventions.doc10.html#182

```
/*
 * @(#)Blah.java        1.82 99/03/18
 *
 * Copyright (c) 1994-1999 Sun Microsystems, Inc.
 * 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of Sun
 * Microsystems, Inc. ("Confidential Information").  You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered into
 * with Sun.
 */
package java.blah;

import java.blah.blahdy.BlahBlah;

/**
 * Class description goes here.
 *
 * @version     1.82 18 Mar 1999
 * @author      Firstname Lastname
 */
public class Blah extends SomeClass {
    /* A class implementation comment can go here. */

    /** classVar1 documentation comment */
    public static int classVar1;

    /**
     * classVar2 documentation comment that happens to be
     * more than one line long
     */
    private static Object classVar2;

    /** instanceVar1 documentation comment */
    public Object instanceVar1;

    /** instanceVar2 documentation comment */
    protected int instanceVar2;

    /** instanceVar3 documentation comment */
    private Object[] instanceVar3;

    /**
     * ...constructor Blah documentation comment...
     */
    public Blah() {
        // ...implementation goes here...
    }

    /**
     * ...method doSomething documentation comment...
     */
    public void doSomething() {
```

```
        // ...implementation goes here...
    }

    /**
     * ...method doSomethingElse documentation comment...
     * @param someParam description
     */
    public void doSomethingElse(Object someParam) {
        // ...implementation goes here...
    }
}
```