Curso de Java orientado a Android – Parte 1

Curso de Java orientado a Android

Introducción

Este es el primero de cuatro posts de introducción al desarrollo en Java. Cabe decir que este tutorial de Java va estar encarado al desarrollo par Android. Es decir, se van a omitir partes que no sean interesantes para el posterior desarrollo de Aplicaciones Android.

- Lenguaje de programación Java
- Java Virtual Machine
- JDK y JRE
- Configuración del equipo para programar en Java
- Hello World en Java
    - Utilizando un editor de texto
    - Utilizando un IDE
- Tipos de datos primitivos
- Utilización de nombres

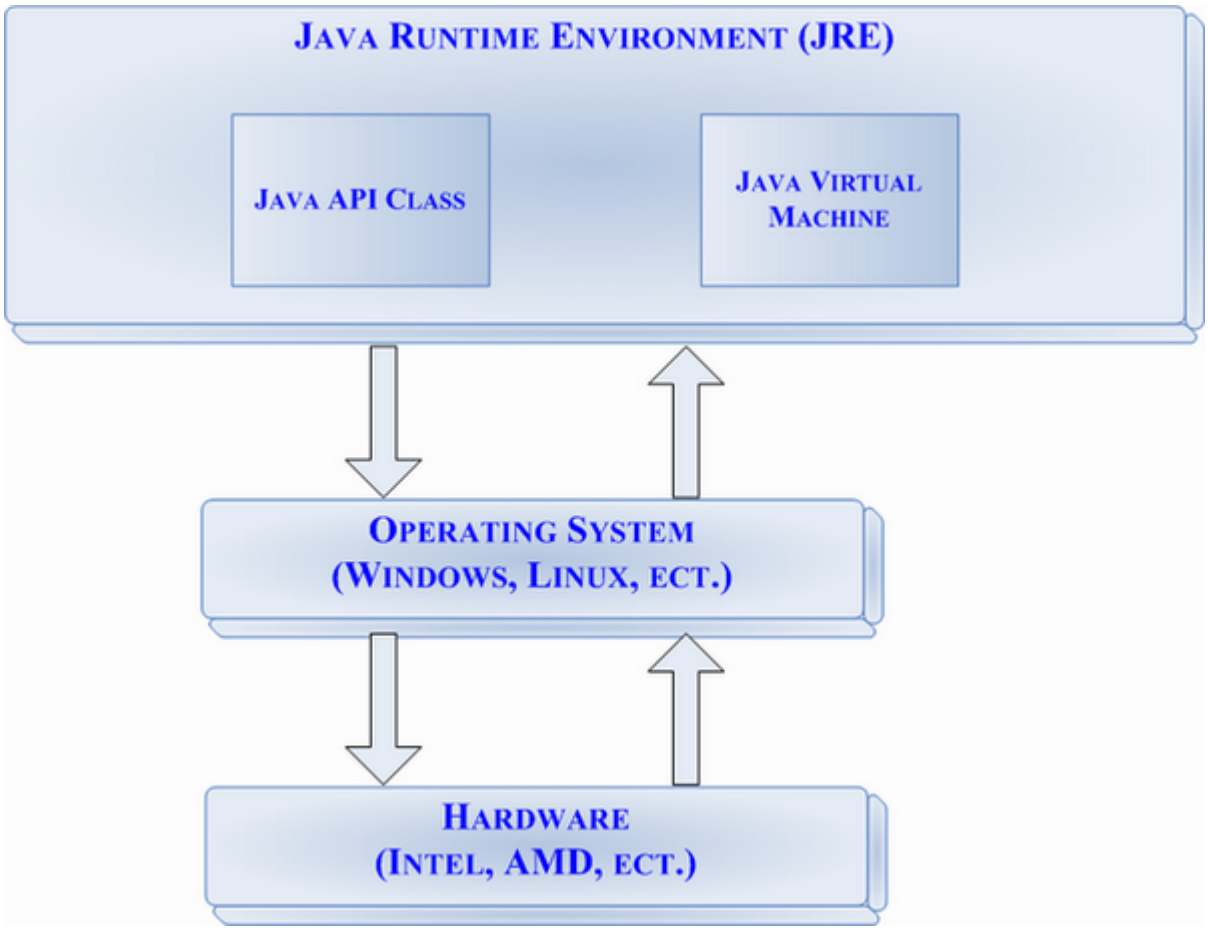
- Comentarios
    - Comentarios para documentación
- Arrays
- Control Flow
    - If/Else and If/Else If
    - Switch statement
    - While loop
    - For Loop

Lenguaje de programación Java

El lenguaje de programación Java fue desarrollado originalmente por Sun Microsystems (Actualmente de Orace) y liberado en 1995. Las aplicaciones Java están típicamente compiladas a byte codes (archivo de clase) que puede funcionar en cualquier máquina virtual de Java (Java Virtual Machine) independientemente del sistema operativo. Java es un lenguaje orientado a objetos. Fue diseñado para permitir a los desarrolladores «escribir una vez y ejecutar en cualquier lugar» (Write Once Run Anywhere), lo que significa que el código que se ejecuta en una plataforma no necesita ser recompilado para funcionar en otra.

Java Virtual Machine

La máquina virtual de Java (JVM) es el componente del framework de Java que ejecuta el código compilado. Cuando un desarrollador compila un archivo Java, el compilador de genera una archivo de bytecode con extensión .class. El bytecode de Java es un lenguaje intermedio generado por el compilador y ejecutado únicamente en una JVM.



JDK y JRE

Para poder iniciar la programación en Java un desarrollador necesita dos componentes principales:

- Java Development Kit.

  El JDK proporciona un compilador de Java, además de otras herramientas. Estas herramientas permiten a un programar código Java y compilarlo a un archivo de bytecodes ejecutable en una JVM. El programa que compila el código se llama javac.En resumen, para empezar a escribir programas Java y compilarlos necesitamos el JDK.

- Java Runtime Environment

  El JRE es el entorno de ejecución para programas Java. Estos programas se compilan en un formato binario portátil (archivos .class) por el compilador. En resumen,

para ejecutar programas compilados en Java necesitamos el JRE.

Configuración del equipo para programar en Java

Instalaremos la version 7 de OpenJdk (versión libre de la plataforma de desarrollo Java) y seguidamente comprobaremos que se ha instalado ejecutando un comando para obtener la versión instalada en nuestro equipo:

```
$ sudo apt-get install openjdk-7-jdk
$ java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (IcedTea 2.5.4) (7u75-2.5.4-1~utopic1)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
```

Hello World en Java

Vamos a crear el programa que por excelencia es el primer paso para cualquier programador en un nuevo lenguaje. Para ello utilizaremos *pluma* y *Eclipse.*

Utilizando un editor de texto (pluma)

Ejecutaremos el siguiente comando

```
$ pluma HelloWorldAdictos.java
```

Escribimos el siguiente código y guardamos el fichero:

```
public class HelloWorldAdictos {
    public static void main(String[] args) {
        System.out.println("Hello World Adictos!");
    }
}
```

Seguidamente ejecutamos el siguiente comando para compliar:

```
$ javac HelloWorldAdictos.java
```

Y finalmente ejecutamos nuestro programa:

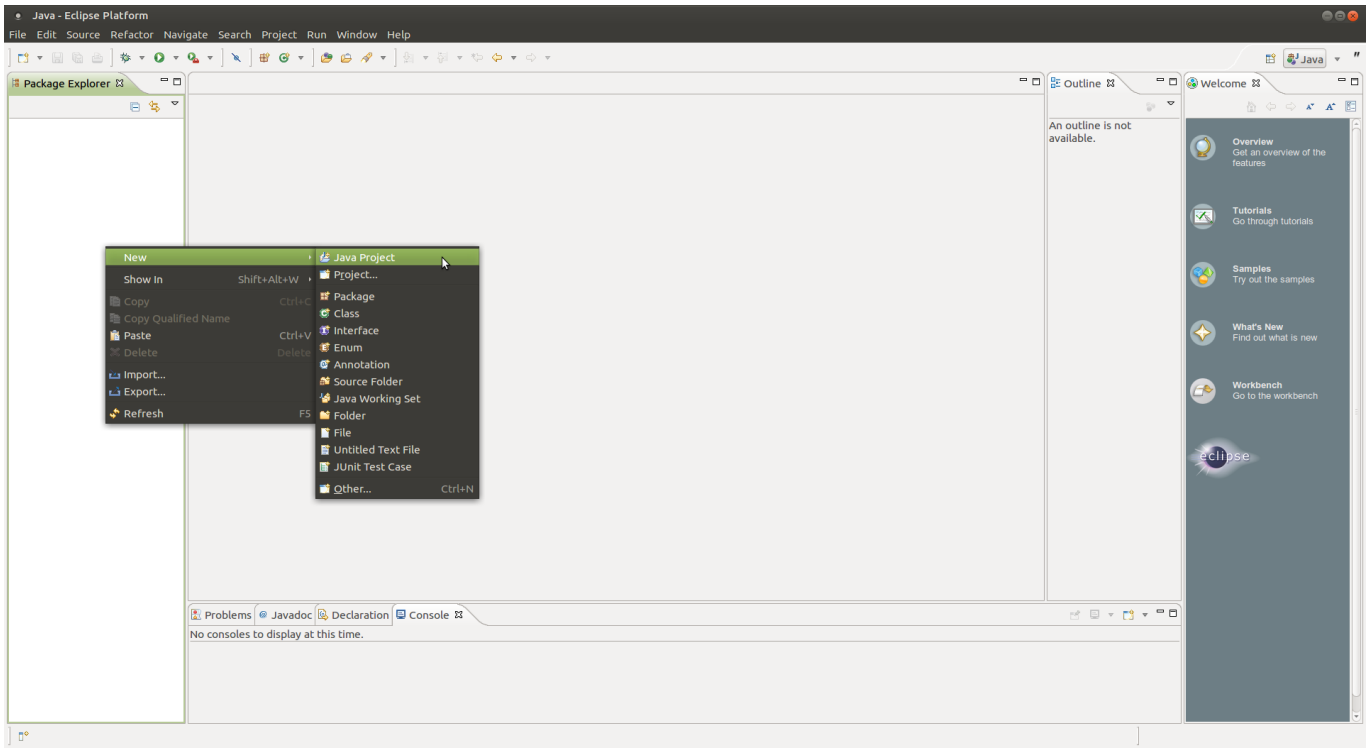
```
$ java HelloWorldAdictos
Hello World Adictos!
```

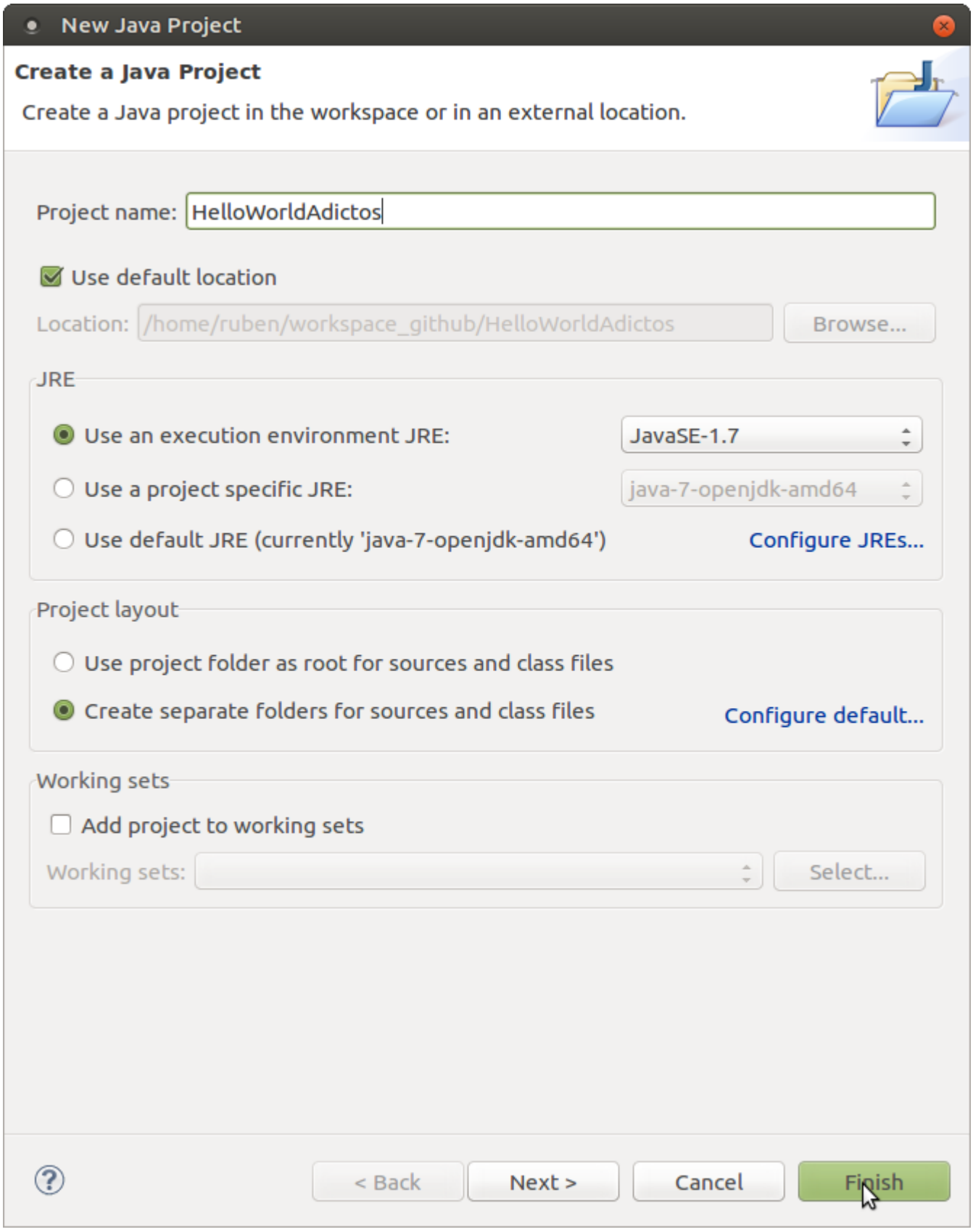
Utilizando un IDE (Eclipse)

Un IDE (Integrated Development Enviroment) es una aplicación que ofrece un conjunto de herramientas para facilitar las tareas de desarrollo a los programadores. En los siguiente pasos instalaremos eclipse, crearemos un proyecto HelloWorldAdictos y lo ejecutaremos. Primeramente instalaremos Eclipse:

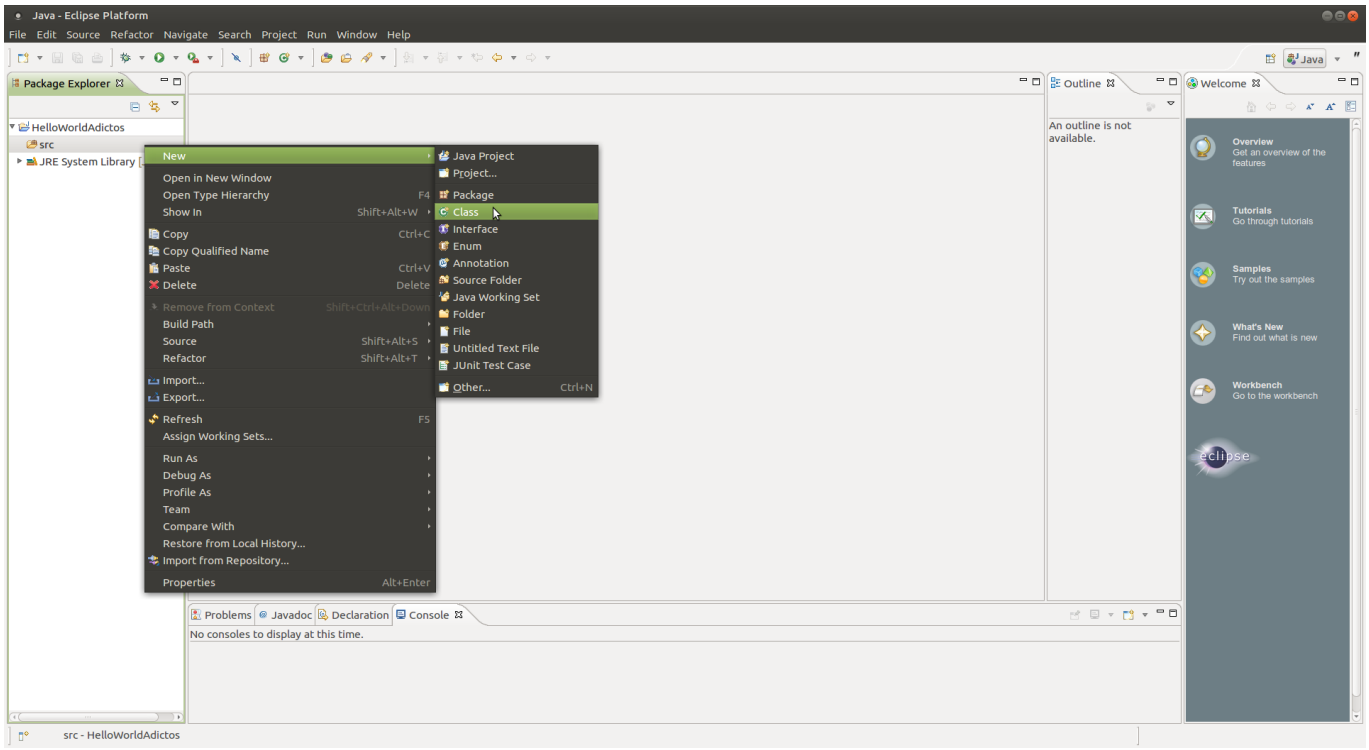
```
$ sudo apt-get install eclipse
```

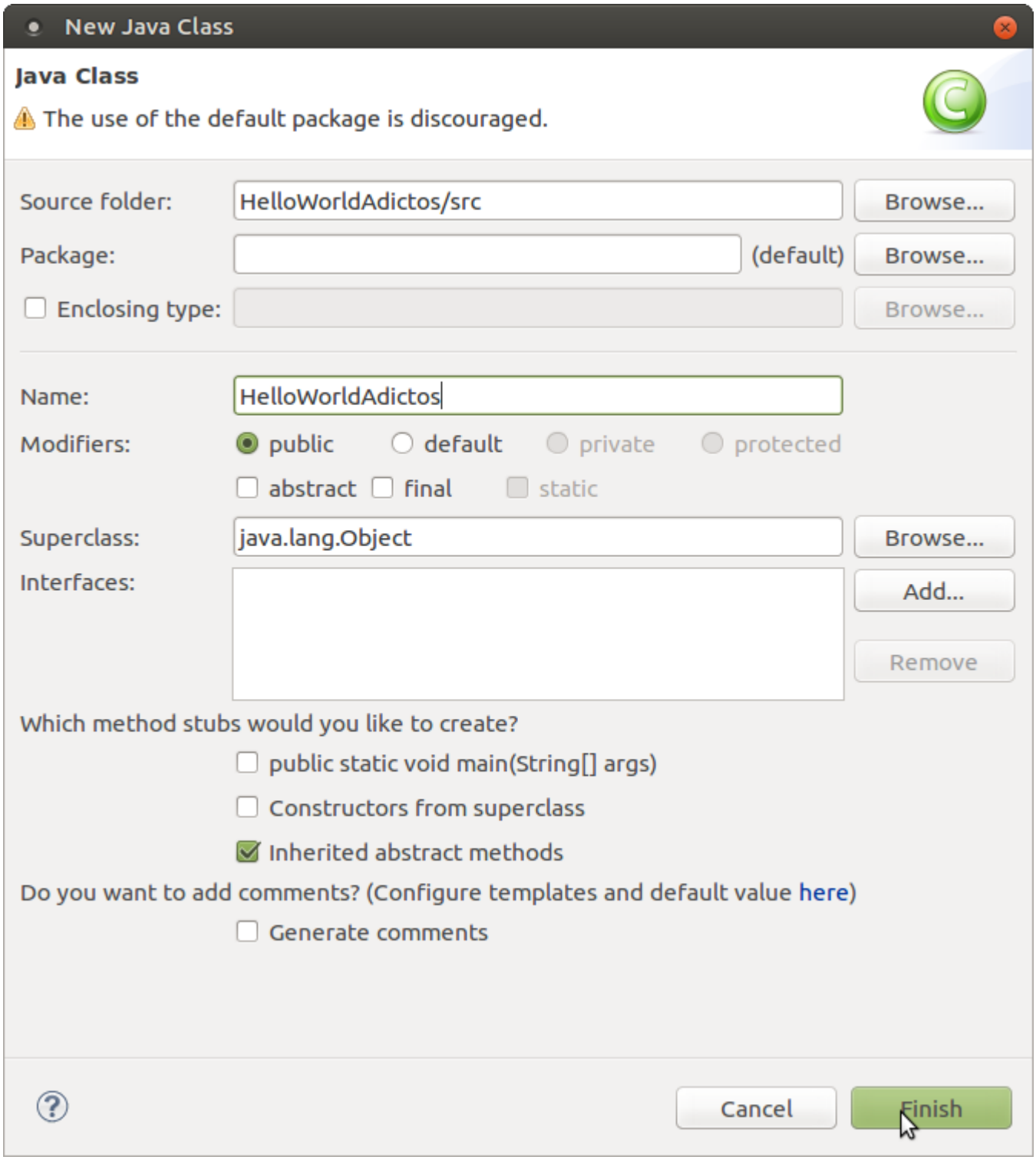
Una vez abierto eclipse creamos un nuevo proyecto java



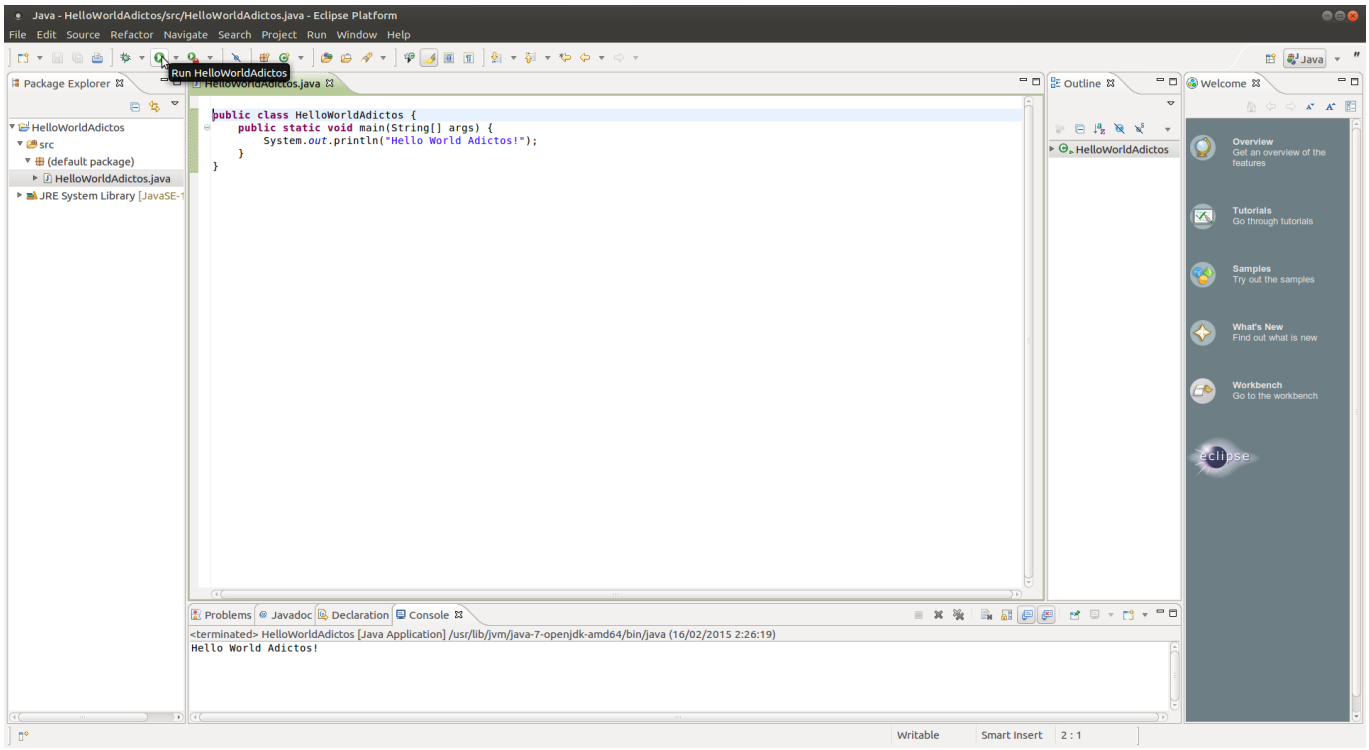


Llegados a este punto deberemos crear una clase en «src»


Java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explorer
HelloWorldAdictos
src
JRE System Library
New
Open in New Window
Open Type Hierarchy F4
Show In Shift+Alt+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Shift+Ctrl+Alt+Down
Build Path
Source Shift+Alt+S
Refactor Shift+Alt+T
Import...
Export...
Refresh F5
Assign Working Sets...
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
Import from Repository...
Properties Alt+Enter
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Other... Ctrl+N
Outline
An outline is not available.
Welcome
Overview
Get an overview of the features
Tutorials
Go through tutorials
Samples
Try out the samples
What's New
Find out what is new
Workbench
Go to the workbench
eclipse
Problems Javadoc Declaration Console
No consoles to display at this time.
src - HelloWorldAdictos



Copiamos nuestro código y ejecutamos nuestro «Hola mundo» clicando en «Run». Finalmente en la consola podemos ver el resultado de la ejecución de nuestro programa



Tipos de datos primitivos

Todas las variables en Java deben ser declaradas antes de que puedan ser utilizadas. Esto se realiza especificando el tipo de de la variable y el nombre de la variable:

float testVar = 1;

Java soporta ocho tipos de datos primitivos diferentes:

- byte: El tipo de datos byte es un entero de 8 bits. El intervalo de valores va desde -2^7 hasta 2^7 -1 (-128 a 127)
- short: El tipo de datos a corto es un entero de 16 bits. El intervalo de valores va desde -2^{15} hasta 2^{15}-1 (-32768 a 32767)
- int: El tipo de datos int es un entero de 32 bits. Tiene un valor máximo de 2147483647. El intervalo de valores va desde -2^{31} hasta 2^{31}-1 (-2147483648 a 2147483647)
- long: El tipo de datos de largo es un entero de 64 bits. El intervalo de valores va desde -2^{63} hasta 2^{63}-1 (-9223372036854775808 a 9223372036854775807)

- float: El tipo de datos float es un punto flotante de 32 bits de precisión simple (por ejemplo: **float a=12.5f;**).
- doble: El tipo de datos doble es un punto flotante de 64 bits de doble precisión (por ejemplo **double PI=3.14159**).
- boolean: El tipo de datos booleano tiene sólo dos posibles valores: true y false (por ejemplo: **boolean isEnabled=true;**).
- char: El tipo de datos char es un solo carácter Unicode de 16 bits. El intervalo de valores va desde 0 hasta 65535 (valores de 16-bits sin signo).

Tipos especiales de caracteres:

- retorno de carro: **\r**
- tabulador horizontal **\t**
- nueva línea **\n**
- barra invertida ****

Utilización de nombres

Java tiene las siguientes reglas y convenciones para utilizar nombre en el código:

- Los nombres de las clases se deben capitalizar (**UserContract**)
- Los nombres de variables distinguen entre mayúsculas y minúsculas (**newUser**)
- Por convención, se deben asignar los nombres de variables utilizando «camel case». Es decir, si el nombre consta de una sola palabra todas la letras serán minúsculas (**phone**). Si consiste en más de una palabra,la primera letra de cada palabra subsiguiente se escribe con mayúscula. (**phoneNumber**)
- También por convención, las constantes son capitalizados y contienen subrayado. (**PI**)

Hay nombres reservados para el lenguaje que no se pueden utilizar:

- Tipos de datos: **boolean, float, double, int, char**
- Sentencias condicionales: **if, else, switch**
- Sentencias iterativas: **for, do, while, continue**
- Tratamiento de las excepciones: **try, catch, finally, throw**
- Estructura de datos: **class, interface, implements, extends**
- Modificadores y control de acceso: **public, private, protected, transient**
- Otras: **super**, **null, this**.

Comentarios

Existen dos tipos de comentarios comentarios:

- De línia //

```
//line comment
```

- Y bloques que comienzan con /* y terminan con */

```
/*first line comment
  second line comment
  third line comment*/
```

Comentarios para la documentación

JDK proporciona javadoc, una herramienta para generar páginas HTML de documentación a partir de los comentarios incluidos en el código fuente. Para utilizar correctamente javadoc debemos seguir unas normas de documentación:

- Los comentarios deben empezar con /** y terminar con */.
- Se pueden incorporar comentarios a nivel de clase, a nivel de variable y a nivel de método.
- Se genera la documentación para miembros public y protected.
- Se pueden usar tags para documentar ciertos aspectos concretos como listas de parámetros o valores devueltos.

Tipo de tag	Formato	Descripción
Todos	@see	Permite crear una referencia a la documentación de otra clase o método.
Clases	@version	Comentario con datos indicativos del número de versión.
Clases	@author	Nombre del autor.
Clases	@since	Fecha desde la que está presente la clase.
Métodos	@param	Parámetros que recibe el método.
Métodos	@return	Significado del dato devuelto por el método
Métodos	@throws	Comentario sobre las excepciones que lanza.
Métodos	@deprecated	Indicación de que el método es obsoleto.

Ejemplo de clase comentada

```
/** Un programa Java simple.
  * Muestra por pantalla el primer digito de version android y
sus nombres.
  * @author Ruben
  * @version 1
  */
public class TestWhile {
        /** Unico punto de entrada.
          * @param args Array de Strings.
          * @return No retorna ningun valor.
          * @throws No dispara ninguna excepcion.
          */
        public static void main(String[] args) {
                String[] androidVersionsNames = new String[5];
                int counter = 0;
                androidVersionsNames[0] = "Donut";
                 androidVersionsNames[1] = "Eclair - Froyo -
```

```
Gingerbread";
                androidVersionsNames[2] = "Honeycomb";
                 androidVersionsNames[3] = "Ice Cream Sandwich
- Jelly Bean - KitKat";
                androidVersionsNames[4] = "Lollipop";
                while (counter <= 4){
                        System.out.println("Version number " +
(counter+1)   +   "   --   Version   name   "   +
androidVersionsNames[counter]);
                        counter++;
                }
        }
}
```

Con el siguiente comando generaremos la documentación en html para la clase TestWhile en el directorio ~/Documents/TestWhileDoc

```
$ javadoc -d  ~/Documents/TestWhileDoc TestWhile.java
```

Arrays

Los arrays son recipientes que contienen un número fijo de valores de un tipo determinado. La longitud de un array es fija y se declara cuando se crea. Por ejemplo, así declaramos un array de integers de 6 elementos:

int [] customArray = new int [5];

Cada contenedor de un array se llama un elemento, y a cada elemento se accede por su índice numérico. El índice de numeración en arrays comienza por 0. Por ejemplo, para acceder al indice 5 utilizaremos el número 4. Se puede asignar un valor a un elemento del array con la siguiente sintaxis:

```
public class TestAarray {
        public static void main(String[] args) {
                int[] customArray;
                customArray = new int[5];
```

```
                // Initialize elements
                customArray[0] = 1;
                customArray[1] = 2;
                customArray[2] = 3;
                customArray[3] = 4;
                customArray[4] = 5;
                  System.out.println("Value at index 0: " +
customArray[0]);
                  System.out.println("Value at index 1: " +
customArray[1]);
                  System.out.println("Value at index 2: " +
customArray[2]);
                  System.out.println("Value at index 3: " +
customArray[3]);
                  System.out.println("Value at index 4: " +
customArray[4]);
        }
}
```

Control Flow

Los bloques de código en Java se ejecutan secuencialmente -de arriba a abajo- en el orden en que aparecen. Sin embargo, un programador puede controlar el flujo de ejecución utilizando condicionales, loops y branches. En esta sección se describe el uso de declaraciones de condicionales (**if /else**, **if /else if** y **switch**), bucles (**for**, **while**), y las expresiones de branch (**break**, **continue**, **return**).

If/Else y If/Else If

If/else ejecuta un bloque de codigo si una condición se evalua como true, en caso contrario se ejecuta otro bloque. La estructura es la siguiente:

```
if (someExpression)
//some code
else
//some code
```

También se pueden encadenar evaluaciones

```
if (someExpression)
//some code
else if (someExpression)
//some code
```

En el siguiente código mediante condicionales if evaluamos el número de versión de Android introducida por el usuario y posteriormente se muestra su nombre

```
import java.util.Scanner;


public class TestIf {

        public static void main(String[] args) {
                int androidVersionNumber = 1;
                String androidVersion="";
                Scanner in;
                in = new Scanner(System.in);
                System.out.println("Enter Android first number
version");
                androidVersionNumber = in.nextInt();
                if (androidVersionNumber == 1) {
                        androidVersion = "Donut";
                }else if(androidVersionNumber == 2){
                          androidVersion = "Eclair - Froyo -
Gingerbread";
                }else if(androidVersionNumber == 3){
                        androidVersion = "Honeycomb";
                }else if(androidVersionNumber == 4){
                        androidVersion = "Ice Cream Sandwich -
Jelly Bean - KitKat";
                }else if(androidVersionNumber == 5){
                        androidVersion = "Lollipop";
                }else{
                          androidVersion = "Invalid number
version";
                }
                    System.out.println("Version -> " +
androidVersion);
```

```
        }
}
```

Switch

La sentencia switch puede tener un número de posibles rutas de ejecución. Switch trabaja con los tipos de datos primitivos byte, short, char e int. En el siguiente código, dependiendo del número introducido por el usuario, se ejecuta una ruta en concreto mediante la cual finalmente se muestra el nombre de una versión de Android

```
import java.util.Scanner;


public class TestSwitch {

        public static void main(String[] args) {
                int androidVersionNumber = 1;
                String androidVersion="";
                Scanner in;
                in = new Scanner(System.in);
                System.out.println("Enter Android first number
version");
                androidVersionNumber = in.nextInt();
                switch (androidVersionNumber) {
                        case 1:
                                androidVersion = "Donut";
                                break;
                        case 2:
                                 androidVersion = "Eclair -
Froyo - Gingerbread";
                                break;
                        case 3:
                                androidVersion = "Honeycomb";
                                break;
                        case 4:
                                 androidVersion = "Ice Cream
Sandwich - Jelly Bean - KitKat";
                                break;
```

```
                        case 5:
                                androidVersion = "Lollipop";
                                break;
                        default:
                                  androidVersion = "Invalid
number version";
                break;
                }
                    System.out.println("Version -> " +
androidVersion);
        }
}
```

While Loop

Una sentencia de bucle while ejecuta continuamente un bloque de código mientras una condición es evaluada como true. Su sintaxis se puede expresar como:

```
while (expresión) {
statement (s)
}
```

El siguiente código itera sobre todas las posiciones del array androidVersionsNames (este contiene el nombre de las versiones de Android). En cada iteración del bucle se comprueba e incrementa la variable counter

```
public class TestWhile {

        public static void main(String[] args) {
                String[] androidVersionsNames = new String[5];
                int counter = 0;
                androidVersionsNames[0] = "Donut";
                 androidVersionsNames[1] = "Eclair - Froyo -
Gingerbread";
                androidVersionsNames[2] = "Honeycomb";
                 androidVersionsNames[3] = "Ice Cream Sandwich
- Jelly Bean - KitKat";
                androidVersionsNames[4] = "Lollipop";
```

```
                while (counter <= 4){
                        System.out.println("Version name " +
androidVersionsNames[counter]);
                       counter++;
                }
        }
}
```

For Loop

La sentencia proporciona una forma compacta para iterar sobre un rango de valores. Se ejecuta en bucle varias veces hasta que una condición se cumple. La forma general de la sentencia for se puede expresar de la siguiente manera:

```
for (inicialización; terminación condición; incremento) {
statement (s)
}
```

El siguiente código ejecuta un bucle for que itera por todas los posiciones del array androidVersionsNames (este contiene el nombre de las versiones de Android)

```
public class TestFor {
        public static void main(String[] args) {

                String[] androidVersionsNames = new String[5];
                androidVersionsNames[0] = "Donut";
                 androidVersionsNames[1] = "Eclair - Froyo -
Gingerbread";
                androidVersionsNames[2] = "Honeycomb";
                 androidVersionsNames[3] = "Ice Cream Sandwich
- Jelly Bean - KitKat";
                androidVersionsNames[4] = "Lollipop";
                for (int counter= 0;counter <= 4; counter++){
                        System.out.println("Version name " +
androidVersionsNames[counter]);
                }
        }
}
```

Observaciones

Bien ya tenemos una primera aproximación al lenguaje Java y sus principales características. Volvemos a repetir este no es un tutorial genérico, es un tutorial especifico encarado a desarrollo de aplicaciones Android. Por eso se obviarán muchas partes como por ejemplo Threads, entornos gráficos (awt)…

Ruben.