UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR

DE

INGENIERIA Y SISTEMAS DE TELECOMUNICACIÓN

PROGRAMACIÓN II

BLOQUE I de prácticas

Práctica 1

Semestre de primavera curso 2013/14

INDICE

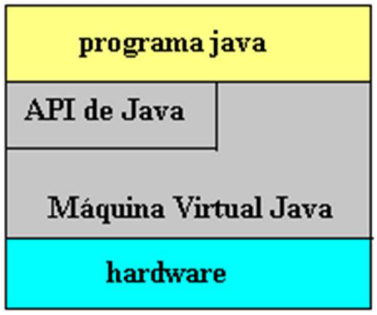
Introducción

Java se introdujo a finales de 1995 como lenguaje de programación para computadores. A finales de 1998 nació el JDK 1.2, más tarde rebautizado como Java 2, y desde entonces se han ido añadiendo sucesivas versiones con diversas mejoras. En el laboratorio de la asignatura se utilizará Java 7, versión JDK 1.7, para la realización de las prácticas.

Un aspecto importante de Java es que se proporciona de forma gratuita el JDK (*Java Development Kit*) y existen numerosos manuales, ejemplos, entornos de desarrollo, etc., que se pueden obtener gratuitamente en Internet (*http://www.oracle.com/technetwork/java/javase/downloads/index.html*)

Otro aspecto importante de Java es el uso de la JVM (Java Virtual Machine). Una máquina virtual Java es el entorno en el que se ejecutan los programas Java, ese entorno permite la portabilidad de las aplicaciones a cualquier arquitectura, de forma que un programa escrito, por ejemplo, en un entorno Windows pueda ser ejecutado en Linux o cualquier otro sistema operativo. Únicamente es necesario disponer de la máquina virtual para dicho entorno y la aplicación podrá ejecutarse en él ("*escríbelo una vez, ejecútalo en cualquier parte*").

El lenguaje nos sirve para crear aplicaciones y la Máquina Virtual Java sirve para ejecutarlas. En la siguiente figura se muestra la arquitectura comentada. La API de Java y la JVM forman un nivel intermedio (Java platform) que aísla el programa Java de las particularidades del hardware y del sistema operativo subyacente.



Entorno de desarrollo

Existen distintos programas comerciales que permiten desarrollar código Java. La compañía Oracle distribuye gratuitamente el JDK. Se trata de un conjunto de programas y bibliotecas que permiten desarrollar, compilar, ejecutar y depurar programas en Java.

El compilador de Java

Se trata de una de las herramientas de desarrollo incluidas en el JDK. Realiza un análisis del código escrito en los ficheros fuente de Java (con extensión *.java). Si no encuentra errores en el código genera los ficheros compilados (con extensión .class). En otro caso muestra la línea o líneas erróneas. También puede suceder

que el compilador genere advertencias sobre aspectos del código que no impiden la compilación, pero que podrían denotar deficiencias de codificación, malentendidos del código, etc. No debe dejarse sin revisar ninguna advertencia ni deben corregirse ciegamente sin comprender la causa por la que el compilador la generó. En el JDK para plataformas Microsoft Windows, dicho compilador se llama javac.exe. Tiene numerosas opciones, algunas de las cuales varían de una versión a otra. Se aconseja consultar la documentación de la versión del JDK utilizada para obtener una información detallada de las distintas posibilidades.

El intérprete Java

El intérprete de Java (java.exe.) se encarga de ejecutar los "*bytecodes*" (ficheros compilados con extensión *.class*) creados por el compilador de Java, sobre JVM.

API de Java (*Application Programming Interface*)

La API es una Interfaz de Programación de Aplicaciones desarrollada por *Sun Microsostem*, proporciona a los programadores un conjunto de clases que permiten simplificar el proceso de desarrollo de una aplicación Java. En la API java encontraremos un conjunto de clases organizadas en paquetes que podremos utilizar en cualquier desarrollo.

La API Java está disponible en: *http://docs.oracle.com/javase/7/docs/api/*

Generación de documentación

La herramienta de generación de documentación *(javadoc.exe*) permite la generación de forma automática de documentación en formato HTML a partir del código fuente Java, siempre que éste se haya marcado adecuadamente con las etiquetas utilizadas por Javadoc. Es muy importante acostumbrarse a documentar el código con las etiquetas que interpreta Javadoc, con el fin de facilitar las tareas de mantenimiento y actualización de las aplicaciones.

Instalación y uso del JDK

Aunque existen entornos de desarrollo integrados IDE (*Integrated Development Environment*) para programar en Java, inicialmente sólo es imprescindible la versión que incorpora el JDK basada en la utilización de una consola (denominada habitualmente *"ventana de comandos"* o el *"símbolo del sistema"* en Windows). Posteriormente se utilizará posteriormente un IDE por su sencillez de uso (en los ordenadores del laboratorio está instalado el IDE Eclipse).

En Moodle, en la documentación adicional para la realización de la práctica, se encontrará un sencillo manual para instalar el JDK7 sobre plataformas Windows. En los PC del laboratorio se encuentra ya preinstalado.

Para poder ver correctamente las tildes en el intérprete de comandos de Windows consulte el anexo al final de este documento.

El desarrollo y ejecución de aplicaciones en Java exige que las herramientas para compilar (***javac.exe***) y ejecutar (**java.exe**) se encuentren accesibles. El ordenador, desde una ventana de comandos de Windows, sólo es capaz de ejecutar los programas que se encuentran en los directorios indicados en la variable de

entorno PATH del sistema operativo o en el directorio de trabajo. Si se desea compilar o ejecutar código en Java, el directorio donde se encuentran el compilador y el intérprete de Java deberá encontrarse en el PATH. Tecleando PATH en una ventana de comandos se muestran los nombres de directorios incluidos en dicha variable de entorno.

En general toda la configuración se realizará de forma automática al instalar el JDK.

Práctica 1

Los objetivos que se pretenden cubrir con el desarrollo de esta práctica son los siguientes:

1. Familiarizarse con uno de los entornos de desarrollo de programación en Java de uso común.
2. Desarrollar programas en lenguaje Java de dificultad baja aplicando los conceptos de programación aprendidos en Programación I.
3. Trabajar con la API de Java.

La duración de esta práctica es de 1 semana.

Prerrequisitos

Se debe estudiar la documentación de la clase ***Scanner*** y de la clase ***String*** (ver documentación de la Unidad 2 y la API de Java), ya que se utilizarán para el desarrollo de esta práctica.

Trabajo a realizar

Durante la sesión, el alumno deberá realizar los ejercicios "HolaMundo" y CambioDeBase. Deberá iniciar "GestionPalabras" y opcionalmente CambioDeBaseC.

Desarrollo de la sesión de trabajo

1. **Actividad 1: Familiarización con el entorno de desarrollo Java.**
   - Objetivos. Manejar el compilador de Java desde la línea de mandatos, ejecutar el programa Java desde la línea de comando y generar la documentación del programa.
   - Descripción. El alumno tiene que editar, compilar y ejecutar el ejercicio denominado "HolaMundo". Deberá además generar la documentación del programa con la herramienta javadoc. Se recomienda llevar ya escrita una primera versión en formato electrónico al laboratorio.
   - Evaluación. Este ejercicio no puntuará, pero su no realización implicará 3 puntos menos en la nota final de la práctica.

2. **Actividad 2: Implementación de un programa que realice el cambio de base de un número entero.**
   - Objetivos. Manejar la entrada de datos usando la clase Scanner y métodos de la clase Integer.
   - Descripción. El alumno deberá traer el programa CambioDeBase.java diseñado y codificado, deberá editarlo compilarlo y probarlo consiguiendo que funcione adecuadamente. Antes del final de la clase de laboratorio deberá subir el programa codificado a Moodle en una tarea de entrega específica.

- Este ejercicio supondrá el 40% del peso de la práctica.

3. **Actividad 3 (Opcional): Implementación de una modificación del programa de la actividad 2.**
   - Objetivos. Manejar la entrada de datos desde la línea de comando.
   - Descripción. El alumno deberá traer el programa *CambioDeBaseC.java* a la clase de laboratorio diseñado y codificado, deberá editarlo compilarlo y probarlo consiguiendo que funcione adecuadamente. El ejercicio se subirá a Moodle junto con el de la actividad 4.
   - Este ejercicio supondrá un 10% adicional sobre el peso de la práctica.

4. **Actividad 4: Implementación de un programa para la búsqueda de palabras en un texto**
   - Este ejercicio deberá comenzarse a trabajar en la sesión del laboratorio. Su objetivo es el uso de los paquetes java. Se realizará una aplicación –programa principal- con clases ya implementadas. En concreto, se utilizará la API de las clases *String* y *Scanner* y algunos de sus métodos
   - La entrega del programa se realizará en una tarea de moodle con fecha posterior a la finalización de la sesión de laboratorio.
   - Este ejercicio supondrá un 60% de la nota final de la práctica.

El alumno deberá leer con atención el apartado *Plazos de entrega y documentación a entregar* de este enunciado en el que se detalla el formato que han de tener los ficheros y documentación electrónica que se suban a la plataforma Moodle para su evaluación.

Actividades a realizar: "HolaMundo", "CambioDeBase", "CambiodeBaseC" (opcional) y "GestionPalabras"

Para la realización de los siguientes ejercicios, se va a utilizar la ventana de comandos del sistema. El JDK ya se encuentra instalado en los ordenadores del laboratorio, por lo que no es necesario realizar la instalación del mismo.

Es necesario que cada ejercicio se escriba en un fichero *.java* que tenga el mismo nombre de la clase, con las mayúsculas/minúsculas exactamente igual. Comience por crear un nuevo directorio en la unidad **I:**, donde guardar todos los ficheros de esta práctica; se sugiere que sea I:\PII\P1.

Actividad 1 "HolaMundo"

Escriba el siguiente programa en el fichero ***HolaMundo.java*** utilizando un editor de texto, por ejemplo *Crimson*:

```
/**
Ejemplo de la estructura general de un programa JAVA
@author Programacion II
@version 1.0
*/

public class HolaMundo
{
```

```
    /**
     Programa principal HolaMundo
     @param args argumentos recibidos en línea de comandos
    */
    public static void main(String args[])
    {
      System.out.println("¡Hola mundo!");
    }
}
```

Una vez escrito el programa, compilarlo y ejecutarlo desde la ventana de comandos de la siguiente forma:

```
D:\clase\programacion II\prácticas>
D:\clase\programacion II\prácticas>javac HolaMundo.java

D:\clase\programacion II\prácticas>java HolaMundo
¡Hola mundo!

D:\clase\programacion II\prácticas>
```

Si ha compilado y no ha mostrado errores el compilador, éste habrá sido el resultado. En caso contrario, corrija los errores y vuelva a compilar. Para utilizar la herramienta de generación de documentación automática (basada en los comentarios que comienzan por /** en el programa) ejecute el siguiente comando:

```
D:\clase\programacion II\prácticas>javadoc -version -author -d carpetaDoc HolaMundo.java
Creating destination directory: "carpetaDoc\"
Loading source file HolaMundo.java...
Constructing Javadoc information...
Standard Doclet version 1.6.0_30
Building tree for all the packages and classes...
Generating carpetaDoc\HolaMundo.html...
Generating carpetaDoc\package-frame.html...
Generating carpetaDoc\package-summary.html...
Generating carpetaDoc\package-tree.html...
Generating carpetaDoc\constant-values.html...
Building index for all the packages and classes...
Generating carpetaDoc\overview-tree.html...
Generating carpetaDoc\index-all.html...
Generating carpetaDoc\deprecated-list.html...
Building index for all classes...
Generating carpetaDoc\allclasses-frame.html...
Generating carpetaDoc\allclasses-noframe.html...
Generating carpetaDoc\index.html...
Generating carpetaDoc\help-doc.html...
Generating carpetaDoc\stylesheet.css...

D:\clase\programacion II\prácticas>
```

Se crearán en la *carpeta* ***doc*** varios ficheros .html, entre ellos uno llamado index.html de entrada a la documentación generada que se puede visualizar con cualquier navegador. Observe el contenido y como se ha generado a partir de los comentarios del programa (consulte la documentación de la herramienta javadoc). La documentación que se genera está en el mismo formato que la documentación de la API de Java.

Actividad 2 "CambioDeBase"

Codificar un programa que lea un número entero desde el teclado y lo imprima en la pantalla. Además debe imprimirlo en la misma línea en binario y en hexadecimal. El programa permanecerá leyendo números hasta que se teclee una línea vacía (un Intro).

Para simplificar esta tarea, se muestra un esqueleto de la aplicación que el alumno deberá completar:

```
/**
* CambioDeBase. Aplicación que permitirá manejar paquetes (Scanner), en este caso
* para leer número enteros desde el teclado. También se habrán de usar métodos
* estáticos de la clase Integer para realizar conversiones. En concreto se usarán:
*        static int parseInt(String s),
*        static String  toBinaryString(int i) y
*        static String  toHexString(int i).
* Descripción: El programa ha de leer un número entero desde el teclado e imprimirlo*
 en la pantalla. Además lo imprimirá en binario y hexadecimal.
* Permanecerá leyendo números hasta que se teclee una línea vacía (un Intro).
*/
import java.util.Scanner;

public class CambioDeBase{
 public static void main (String [] arguments){
// Resto de variables de la aplicación (codificar por el alumno)

// Creación del objeto para manejar la lectura de datos.
   teclado = new Scanner( System.in );

// Sacar mensaje invitando a teclear un número entero.
// Leer la cadena representando el número entero usando teclado.nextLine()
// Mientras la cadena no sea una cadena vacía (“”)
      // Convertir la cadena a entero: usando Integer.parseInt(String)
      // Imprimir en número en base 10, en hexadecimal y en binario
      // Leer otro numero
}
}
```

Una posible ejecución del programa podría ser la siguiente:

```
I:\PII\BlqI\P1>java CambioDeBase
Teclea un número (Intro para terminar)?: 12
12      0xc     1100B

Teclea un número (Intro para terminar)?: 16
16      0x10    10000B

Teclea un número (Intro para terminar)?: 780
780     0x30c   1100001100B

Teclea un número (Intro para terminar)?: 41
41      0x29    101001B

Teclea un número (Intro para terminar)?:
```

Actividad 3 (Opcional) “CambioDeBaseC”

Codificar un programa que lea una lista de números enteros (en base diez) desde la línea de comando e imprima cada uno de ellos en la misma línea en base diez, en binario y en hexadecimal.

El formato del comando que se ha de implementar es:

java CambioDeBaseC Numero1 Numero2… NumeroN

La siguiente figura representa un esqueleto del programa:

```
/**
* CambioDeBaseC. Aplicación que permitirá manejar la entrada de datos desde la linea
* de comando. También se usan métodos estáticos de la clase Integer para realizar
* conversiones. En concreto se usarán:
*        static int Integer.parseInt(String s),
*        static String  toBinaryString(int i) y
*        static String  toHexString(int i).
* Descripción: El programa ha de obtener una lista de números enteros desde la linea
* de comando e imprimirlos en la pantalla en base diez, en binario y en hexadecimal.
*/

public class CambioDeBaseC{
 public static void main (String [] arguments){
// Resto de variables de la aplicación (codificar por el alumno)
//Comprobación del formato de los parámetros
    if (arguments.length == 0){
            System.out.println ("Formato erroneo. El formato correcto es: \n"+
                                                "\t\tjava CambioDeBaseP Numero1
 Numero2... NumeroN");
            System.exit(0);
     }
// Obtención de los números a convertir desde la línea de comando y escritura
//… en pantalla. (Codificar por el alumno).

}
}
```

A continuación se muestra una posible ejecución del programa:

```
I:\PII\BlqI\P1>java CambioDeBaseC 1 2 7 8 100 3000 7034
1       0x1     1B
2       0x2     10B
7       0x7     111B
8       0x8     1000B
100     0x64    1100100B
3000    0xbb8   101110111000B
7034    0x1b7a  1101101111010B
```

Actividad 4. Ejercicio sobre GestionPalabras

Escriba un programa en Java que busque las palabras que empiecen por una determinada letra, en mayúscula o minúscula, en las distintas líneas de texto que se irán leyendo una a una a través de teclado. El programa pedirá que se teclee la letra que se usará para formar el patrón de búsqueda, y a continuación irá

pidiendo líneas una a una. Una vez leída una línea por teclado escribirá en la pantalla las palabras que en esa línea cumplan la condición. A continuación pedirá la siguiente línea y repetirá este proceso hasta que se introduzca una línea vacía. Al finalizar el programa indicará el número total de palabras que han cumplido la condición.

NOTA: no se podrá utilizar un algoritmo que vaya buscando recorriendo la cadena carácter a carácter buscando espacios en blanco para acotar las palabras. Se aconseja utilizar una solución que use el método *Split(String)* de la clase String. Para hacer este ejercicio es imprescindible que el alumno consulte la API de Java sobre la clase *String* que se puede encontrar en:

http://download.oracle.com/javase/7/docs/api/java/lang/String.html.

Se penalizará en la calificación de este ejercicio cualquier solución algorítmica que se pudiera haber realizado no usando principalmente los métodos de la clase *String*.

Se valorará que se identifiquen y traten posibles condiciones límite, por ejemplo que la letra tecleada esté vacía. Una ejecución de la aplicación deberá producir el siguiente resultado:

```
I:\PII\BlqI\P1>java GestionPalabras
Teclee la letra de patron de busqueda: a

Teclee una linea del texto: Esta es la linea de pruebas de Amalia
Amalia
Teclee una linea del texto: Ava Gadner era amable y guapa
Ava amable
Teclee una linea del texto:
El número de palabras que empezaban por “a” es: 3
```

Plazos de realización, entrega y documentación a entregar

La práctica 1 se realizará en la semana lectiva 4 que comienza el 28 de febrero y finaliza el 6 de marzo

Los archivos que son solución a la práctica 1 de este bloque de prácticas deberán entregarse a través de la plataforma Moodle de la siguiente manera:

- El resultado con el programa de la actividad 2 (clase `CambioDeBase.java`) al final de la clase de laboratorio. Cualquier entrega retrasada no será evaluada. **Tenga en cuenta que la plataforma Moodle registra la hora de entrega**.
- El resultado con los programas de las actividades 3 y 4 (clases `CambioDeBaseC.java` y `GestionPalabras.java`), deberán entregarse con plazo máximo de 48 después de finalizar la sesión de laboratorio (Por ejemplo el grupo V4 -viernes de 15:30 a 17:30- podría entregar como máximo hasta las 17:30 del domingo). En esta entrega se deberá incluir la documentación generada por javadoc de la actividad 4.

El alumno subirá cada entrega en un fichero ZIP a la plataforma Moodle con las siguientes características:

a) El archivo ZIP contendrá los ficheros .java de las aplicaciones y una carpeta llamada **doc** con la documentación generada con el javadoc, cuando se pida.
b) El nombre del fichero ZIP debe tener el formato:

grupolaboratorio_B1_P1_A<numero de la actividad>_apellidos_nombre.zip

Por ejemplo, un alumno que se llamara Rafa Nadal y que estuviera en el grupo V4 subiría a la plataforma Moodle los ficheros:

V4_B1_P1_A2_Nadal_Rafa.zip para la segunda actividad.

V4_B1_P1_A3-4_Nadal_Rafa.zip para la tercera y cuarta actividad.

Estas reglas deben respetarse **obligatoriamente** para que las prácticas sean evaluadas.

Evaluación

La práctica 1 será evaluada conjuntamente con el resto de prácticas del bloque I al finalizar el mismo.

- El peso total de esta práctica respecto al bloque I será de un 10% de la nota final del bloque.
- La siguiente tabla muestra los porcentajes de evaluación de cada parte de ésta práctica:

Hito de evaluación	Porcentaje en el peso de la práctica 1
Actividad 2	**40 %**
Actividad 4	**60 %**
Actividad 3	**10 % adicional**

Anexo. Como configurar el intérprete de comandos de Windows para ver correctamente las tildes

En la configuración por defecto del intérprete de comandos de Windows no se ven correctamente las tildes al ejecutar una aplicación java que utiliza el juego de caracteres UTF-8. A continuación se describe como configurar esta aplicación.

En primer lugar hay que configurar la fuente de letra:

1. Abrir el intérprete de comandos.

   Menú de inicio→Todos los programas→Accesorios→Símbolo del sistema

2. Pulsar con el botón derecho de tareas en la barra azul superior de la aplicación. Se desplegará un menú. Seleccionar la opción Propiedades.
3. En la pestaña Fuente elegir Lucida Console
4. Pulsar el botón Aceptar

Si este cambio se quiere hacer permanente, habría que hacer esta operación en la opción Predeterminados en vez de en Propiedades, dentro del menú del símbolo del sistema.

En segundo lugar hay que ejecutar el comando chcp 1252.

Al cerrar el intérprete de comandos se perderá esta configuración, por lo que es necesario ejecutar chcp cada vez que se habrá.

En los ordenadores del laboratorio de DIATEL está configurado el intérprete de comandos de Windows de forma que se ven correctamente las tildes, por lo que no es necesario realizar estas operaciones.