Universidad Autónoma de Madrid

Escuela Politécnica Superior

Análisis y Diseño de Software 2020-2021

Práctica 1: Introducción a Java

Inicio: A partir del 15 de febrero.
Duración: 1 semana.
Entrega: Una hora antes del comienzo de la siguiente práctica.
Peso de la práctica: 5%

El objetivo de esta práctica es aprender el funcionamiento de algunas de las herramientas de la Java Development Toolkit (JDK), comprender el esquema de funcionamiento de la máquina virtual de Java y escribir tus primeros programas en Java.

Apartado 1: Hola Mundo

Con un editor de texto, teclea el siguiente programa, y guárdalo con el nombre *HolaMundo.java*

```
public class HolaMundo {

   public static void main(String[] args) {
      System.out.println("Hola mundo!");   // muestra el string por stdout
   }
}
```

En la línea de comandos, ejecuta la siguiente sentencia:

```
javac HolaMundo.java
```

para compilar la clase *HolaMundo* y generar el fichero *HolaMundo.class* correspondiente. Ejecuta el fichero *.class* mediante la sentencia:

```
java HolaMundo
```

Nota: el nombre del fichero *.java* tiene que ser el mismo que la clase que contiene respetando mayúsculas y minúsculas. Asegúrate de que los programas *javac* y *java* estén en el PATH[1].

Apartado 2: Generación de Documentación.

El programa *javadoc* permite generar documentación HTML de los distintos programas fuente leyendo los comentarios del código. Por ejemplo, modifica el programa anterior incluyendo los siguientes comentarios, añadiendo tu nombre como autor:

[1] https://javadesdecero.es/fundamentos/instalar-configurar-variables-entorno/

```java
/**
 * Esta aplicación muestra el mensaje "Hola mundo!" por pantalla
 *
 * @author Estudiante EPS estudiante.eps@uam.es
 *
*/
public class HolaMundo {

 /**
  * Punto de entrada a la aplicación.
  *
  * Este método imprime el mensaje "Hola mundo!"
  *
  * @param args Los argumentos de la línea de comando
  */
   public static void main(String[] args) {
      System.out.println("Hola mundo!");   // muestra el string por stdout
   }
}
```

Genera la documentación del programa mediante la sentencia:

```
javadoc -author HolaMundo.java
```

Nota: es conveniente almacenar los ficheros de documentación en un directorio separado. Por ejemplo, *javadoc -d doc HolaMundo.java* los almacena en el directorio *doc*, creando el directorio si no existe.

Abre la página *index.html* para ver la documentación generada. Tienes más información sobre el formato adecuado para comentarios *JavaDoc* en: http://en.wikipedia.org/wiki/Javadoc

---

Apartado 3: Uso de librerías básicas

El siguiente programa ordena de menor a mayor una serie de números enteros, que se le pasan por la línea de comandos.

```java
import java.util.SortedSet;
import java.util.TreeSet;

/**
 * Esta clase mantiene ordenado un conjunto de números enteros
 */
public class Ordena {
    // usamos un conjunto ordenado, que implementa TreeSet
    private SortedSet<Integer> numeros= new TreeSet<>();

    /**
     * Constructor, con el array de cadenas
     * @param cadenas a insertar, tras convertir a números
     */
    public Ordena(String ... cadenas){
        for (String s: cadenas){ //recorremos el array
            int n= Integer.parseInt(s); //convertimos a entero
            numeros.add(n); //añadimos al conjunto
        }
    }

    /**
     * @return la colección de números
     */
    public SortedSet<Integer> getNumeros(){
        return numeros;
    }

    /**
     * @return la cantidad de números en la colección
     */
    public int getTotalNumeros() {
        return this.numeros.size();
    }
```

```
    /**
     *
     * @return Cadena que representa este objeto
     */
    public String toString(){
        return this.numeros.toString();      // imprime el conjunto
    }
    /**
     * Punto de entrada a la aplicación.
     *
     * Este método ordena los números de la línea de comando
     * @param args Los argumentos de la línea de comando. Se esperan enteros, como cadenas
     */
    public static void main(String[] args) {
        if (args.length<2) {
            System.out.println("Se espera al menos dos números como parámetros");
            System.out.println("Devuelve el conjunto ordenado");
        }
        else {
            Ordena c = new Ordena(args);
            System.out.println("Tenemos "+c.getTotalNumeros()+" números");
            System.out.println(c); // Imprimimos el conjunto ordenado, por salida estándar
            // En java la destrucción de objetos es automática
        }
    }
}
```

El programa declara una clase *Ordena*. Las variables de esta clase (llamados *objetos*), se crean llamando al método "*public Ordena(String ... cadenas)*". En programación orientada a objetos, este método se llama "*constructor*", y en este caso recibe un número arbitrario de parámetros de tipo *String*. La clase también define tres métodos (*getNumeros, getTotalNumeros y toString*) que devuelven la colección de números, la cantidad de números almacenados y una representación del objeto en forma de cadena, para imprimirlo por pantalla. El punto de entrada es el método "main". Como puedes observar, la llamada al constructor se realiza con el operador "*new*", y la llamada a otros métodos del objeto se realiza con la notación "*objeto.metodo(params)*".

Recuerda grabar el programa en un fichero llamado *Ordena.java*.

Ejecuta el programa con distintos parámetros (numéricos o no, negativos, etc.), o con parámetros. ¿Qué sucede? ¿Ves algún problema o mejora a implementar en dicho código?

Algunas clarificaciones adicionales:

- Como has visto en el constructor, en Java para recorrer objetos suele usarse un "*for*" mejorado de la forma `for (<Tipo> s: <coleccion>)` donde *s* es una variable de tipo <*Tipo*>, que va tomando cada uno de los valores de la colección <*coleccion*> en las iteraciones del bucle.

- Para la concatenación de cadenas se puede utilizar el operador "+". Dicho operador puede usarse con cualquier tipo de dato, y este se convertirá automáticamente a cadena si cualquiera de los operandos es una cadena. Esta conversión automática también ocurre cuando ejecutamos *System.out.println(objeto)*. En este caso se llama automáticamente al método *toString()* del objeto.

Apartado 4: Tu primer programa Java (10 puntos)

Basándote en el programa anterior, crea un programa Java que mantenga ordenadas por separado una colección de números pares, y otra de números impares, y nos permita obtener su suma. Puedes completar el siguiente fragmento:

```
public class OrdenaYSuma {
   //… completar
   public static void main(String... args) {
     if (args.length<2) {
        System.out.println("Se espera al menos dos números como parámetros");
        System.out.println("Devuelve los números pares e impares ordenados, sin repetidos, y su suma");
     }
     else {
        OrdenaYSuma c = new OrdenaYSuma(args);
        System.out.println(c); // Imprimimos los conjuntos ordenados y su suma, por salida estándar
        c.anyade(30); // método para añadir un numero
        c.anyade(33); // Añade el 33
        System.out.println(c); // Imprimimos los conjuntos ordenados y su suma
        System.out.println("Suma pares: "+c.getSumaPares()); // Imprimimos la suma de los pares
        System.out.println("Suma impares: "+c.getSumaImpares()); // Imprimimos la suma de los impares
        System.out.println("Suma total: "+(c.getSumaPares()+c.getSumaImpares())); // suma total
     }
   }
}
```

Si ejecutamos el programa con la entrada 4 -3 5 2 obtendríamos la siguiente salida:

```
Números pares [2, 4] (suma: 6)
Números impares [-3, 5] (suma: 2)

Números pares [2, 4, 30] (suma: 36)
Números impares [-3, 5, 33] (suma: 35)

Suma pares: 36
Suma impares: 35
Suma total: 71
```

Algunas indicaciones (aunque trataremos todas estas cuestiones en mucho mayor detalle durante el curso):

- Para la concatenación de cadenas se puede utilizar el operador "+". No obstante, ¿qué ocurre si quitamos los paréntesis de la línea `System.out.println("Suma total: "+(c.getSumaPares()+c.getSumaImpares()));`? ¿Puedes explicar lo que ocurre?
- Para comprobar si un número es par o impar, puede usarse el operador módulo (%) y el de igualdad (==).
- La liberación de memoria en Java es automática, como veremos en los próximos temas, por lo que no necesitas liberar memoria ni destruir objetos al final del programa.
- Las variables internas de una clase (llamadas atributos) pueden declararse con un control de acceso (*private*, *public* y otros). Normalmente serán de tipo *private* para evitar que sean modificadas desde el exterior. También puede declararse un control de acceso para los métodos: los privados será las funciones auxiliares (sólo necesarias internamente, desde dentro de la clase), y los públicos serán los que se pueden llamar desde fuera de la clase.
- El código del programa estará en un archivo llamado *OrdenaYSuma.java*.

Normas de entrega:

- El nombre de los alumnos debe ir en la cabecera *JavaDoc* de todas las clases entregadas
- La entrega la realizará uno de los alumnos de la pareja a través de Moodle
- Se debe entregar un único fichero ZIP / RAR con todo lo solicitado, que deberá llamarse de la siguiente manera: GR<numero_grupo>_<nombre_estudiantes>.zip. Por ejemplo Marisa y Pedro, del grupo 2261, entregarían el fichero: GR2261_MarisaPedro.zip
- La estructura de los ficheros entregados deberá ser la siguiente:
  - **src**. Ficheros fuente
  - **doc**. Documentación *JavaDoc* generada
  - **cuestiones.txt**. Respuestas a las diferentes preguntas planteadas en los apartados 3 y 4.