Universidad Autónoma de Madrid

Escuela Politécnica Superior

Análisis y Diseño de Software 2019-2020

Práctica 1: Introducción a Java

Inicio: A partir del 3 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 que los programas *javac* y *java* estén en el PATH.

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:

```
/**
 * 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 muestra cómo se reciben los parámetros por la línea de comandos y cómo se utiliza la clase SortedSet para mantener un conjunto ordenado de números.

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

/**
 * Esta clase mantiene 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 numeros
     */
    public SortedSet<Integer> getNumeros(){
        return numeros;
    }

    /**
     *
     * @return Cadena que representa este objeto
     */
    public String toString(){
        return "Hay "+ numeros.size()+ " números"
                + ": " + numeros; //o +numeros.toString()
    }
    /**
     * 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(c); // Imprimimos el conjunto ordenado, por salida estándar
            // En java la destrucción de objetos es automática
        }
    }
}
```

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

Ejecuta el programa con distintos parámetros (numéricos o no, negativos, etc.), o sin parámetros. ¿Qué sucede?

Apartado 4: Tu primer programa Java (10 puntos)

Crearemos una nueva clase Primos que nos permita saber si un número es primo. Esta clase usará un conjunto ordenado para mantener todos los números primos que ya ha encontrado. También tendrá una variable "max", con el número mayor que se ha comprobado. Añadiremos inicialmente dos métodos sencillos, el que obtiene los primos calculados hasta el momento, y el método toString(), que devuelve una cadena representando el objeto.

```
public class Primos {
   // usamos un conjunto ordenado, que implementa TreeSet
   private SortedSet<Integer> primos= new TreeSet<>();
   private int max=1;

   /**
    *
    * @return cache con los primos calculados
    */
   public SortedSet<Integer> getPrimos(){
       return primos;
   }

   public String toString(){
       return "Primos hasta "+ max+ " = "+primos;
   }
```

El método más importante de esta clase será "boolean esPrimo(int n)", que devolverá si n es primo. Puedes usar la siguiente implementación como punto de partida:

```
public boolean esPrimo(int n){
   if (n<2) return false;
   if (n>max) actualizaPrimos(n);
   return primos.contains(n);
}
```

Solo falta crear los métodos de más bajo nivel. El primero, actualizaPrimos(int n), ha de añadir todos los primos que faltan, entre max+1 y n, al conjunto primos, y ha de actualizar max, para que pase a ser n.

Por último, el método básico, que comprueba si un número es primo, boolean compruebaPrimo(int n), asume que ya están en el conjunto primos todos los números primos menores a n, y devuelve si n es primo.

Puedes recorrer el conjunto primos con un bucle for similar a este:

```
for (int p:primos) { /* p toma en cada iteración un elemento del conjunto primos */ }
```

Además Java permite una estructura "clásica" de bucle *for* (con gestión explícita de índices) como ésta:

```
for ( <var inicialización>; <condición>; <actualización> ) {
   <cuerpo del bucle>
}
```

El bucle clásico se puede utilizar con arrays, o listas, pero no con conjuntos, ya que carecen de un método para obtener un elemento dada su posición.

Para la concatenación de cadenas hemos utilizado el operador "+". Se puede utilizar con cualquier tipo de dato, y éste se convertirá a cadena automáticamente 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.

Como veremos en próximos temas, la liberación de memoria en Java es automática, así que no necesitas liberar memoria ni destruir los objetos al final del programa.

El código de la clase Primos estará en un archivo llamado Primos.java.

Para probar el código se creará un main en la clase Primos, similar al del apartado 3, donde se lea más de un número como argumento. Se ha de mostrar si cada número es primo, y el conjunto de números primos encontrados hasta ese momento.

Notas:

- Para indicar si algo es público o privado, ponemos public o private delante del atributo o método. En este caso, los métodos que has de completar, al ser internos, han de declararse como private, en lugar de public.
- Para que podamos usar las clase de la librería (TreeSet, SortedSet, etc.) sin poner delante el nombre del paquete (java.util), hemos de importarlas. Puedes añadir la siguiente línea al comienzo del archivo para importar todas las clases del paquete java.util, en lugar de indicarlas de forma individual:

  ```
  import java.util.*; //importamos todas las clases del paquete java.util
  ```

Apartado 5: Ejercicio opcional (1 punto)

Añade a la clase Primos un método divisoresPrimos, que devuelva el conjunto de números primos que son divisores del número indicado por argumento.

```
public SortedSet<Integer> divisoresPrimos(int n)
```

Modifica también el método main de la clase Primos para imprimir los divisores primos de los números pasados por argumento, en caso de que no sean primos.

Notas:

- En las siguientes prácticas, en lugar de modificar clases existentes, en este caso Primos, será preferible crear subclases mediante herencia.
- Para simplificar, no estamos realizando una factorización del número, ya que no se pide saber cuantas veces aparece cada número primo.

Normas de entrega:

- Se deberá entregar el apartado 4 (y opcionalmente el 5)
- 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