Universidad Autónoma de Madrid

Escuela Politécnica Superior

Análisis y Diseño de Software 2024-2025

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 del 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* están 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:

```
/**
 * 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:

---

[1] https://www.java.com/es/download/help/path_es.html

```
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 (1.5 puntos)

El siguiente programa implementa de manera muy simplificada una cartelera de cine. La clase `CarteleraCine` contiene una lista de objetos de tipo `Pelicula`, cada uno con su título, género y año. Cada clase debe ir en un archivo distinto, llamado como el nombre de la clase.

Archivo Pelicula.java

```java
public class Pelicula {
	private String titulo;
	private String genero;
	private int anyo;

	public Pelicula (String titulo, String genero, int anyo) {
		this.titulo = titulo;
		this.genero = genero;
		this.anyo = anyo;
	}

	@Override
	public String toString() {
		return this.titulo+" ("+this.genero+"): "+this.anyo;
	}
	public String getTitulo() {
		return titulo;
	}
	public String getGenero() {
		return genero;
	}
	public int getAnyo() {
		return anyo;
	}
}
```

Archivo CarteleraCine.java

```java
import java.util.*;

public class CarteleraCine {
	private String nombreCine;
	private List<Pelicula> peliculas = new ArrayList<>();

	public CarteleraCine(String cine, Pelicula[] peliculas) {
		this.nombreCine = cine;
		for (Pelicula p: peliculas)
			this.peliculas.add(p);
	}
	@Override
	public String toString() {
		return "Cine: "+this.nombreCine+"\nPeliculas en cartelera: "+this.peliculas;
	}

	public static void main(String ...args) {
		Pelicula peliculas[] = {   new Pelicula("Perfect days", "Drama", 2023),
		                           new Pelicula("Inception", "Accion", 2010),
		                           new Pelicula("Jumanji", "Aventura", 1995) };
		CarteleraCine cinesTelmo = new CarteleraCine("Telmo", peliculas);
		System.out.println(cinesTelmo);
	}
}
```

El programa declara una clase llamada `Pelicula`. Las variables de esta clase (llamadas *objetos*), se crean llamando al método `public Pelicula(String titulo, String genero, int anyo).` En programación orientada a objetos, este método se llama *"constructor",* y en este caso recibe tres parámetros. Las llamadas al constructor se realizan mediante el operador new. La clase define un método adicional sobrescrito, `toString`, que proporciona una representación del objeto en forma de cadena, así como tres métodos "getters", que devuelven el valor de cada uno de los atributos de la clase.

La clase `CarteleraCine` tiene como atributos el nombre del cine y la lista de películas en cartelera. El constructor recibe ambos como parámetro, y añade cada película del array a la lista usando el método `add`. Las llamadas a los métodos del objeto se realizan con la notación *"objeto.metodo(params)"*. El método `toString` de `CarteleraCine` simplemente muestra el nombre del cine y las películas.

El punto de entrada del programa es el método `main`, como en el lenguaje *"C"*. En este método, se crean 3 películas – usando el operador new – y se almacenan en un array. También se crea un objeto `CarteleraCine` y se imprime.

Algunas aclaraciones adicionales:

- La referencia "`this`" es una constante que representa el objeto que ejecuta el método.
- En Java, existen varias primitivas de iteración, como `while` o `for`. Para recorrer colecciones puede usarse el for tradicional al estilo "C", pero en Java es mucho más frecuente y conveniente usar el for mejorado que tiene la forma:

  ```
  for (<Tipo> p: <coleccion>)
  ```

  donde `p` es una variable de tipo <Tipo>, que va tomando cada uno de los valores de la colección <coleccion> en las iteraciones del bucle.
- El programa utiliza dos tipos del paquete `java.util`: `List` y `ArrayList`, que permiten usar listas. Los arrays, las listas, y otras colecciones de `java.util`, son iterables mediante el for mejorado.
- La clase `Pelicula` sobrescribe (*override*) el método `toString` para definir cómo se debe convertir un objeto `Pelicula` a una cadena de texto. Esto permite que, al imprimir el objeto directamente con `System.out.println(objeto)`, se llame automáticamente al método `toString`.
- La concatenación de cadenas se realiza con el operador +, que convierte automáticamente los datos al tipo `String` si alguno de los operandos es una cadena.
- La liberación de memoria en Java es automática, como veremos en los próximos temas, por lo que no es necesario liberar memoria ni destruir objetos al final del programa.
- Las variables internas de una clase (llamadas atributos) pueden declarar 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 serán 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. Los métodos como `toString` y los `getters` son públicos porque necesitan ser utilizados fuera de la clase.

Contesta a las siguientes preguntas:

Modifica el `main` para añadir más películas en el array `peliculas`.

- ¿Qué sucede si creas dos objetos `Pelicula` con los mismos parámetros?
- ¿Puedes crear una `Pelicula` sin género? ¿Y con varios?
- ¿Qué sucede si le pasas al objeto `CarteleraCine` un array vacío? ¿Y `null`? ¿Cómo puedes mejorar el código para que ese problema no suceda?

Apartado 4: Tu primer programa Java (8.5 puntos)

Modifica el programa anterior, añadiendo un atributo `director` a la clase `Pelicula`, y haz que se muestre el director al imprimir una `Pelicula`. Extiende la clase `CarteleraCine` con dos métodos para obtener listas de películas de acuerdo a ciertos criterios. Para ello, crea un método `peliculasPorGenero`, que reciba un género como parámetro, y devuelva la lista de películas de dicho género. Añade otro método, `peliculasPosterioresA`, que reciba un año como parámetro y devuelva la lista de películas posteriores a dicho año.

Modifica el `main` del apartado anterior para obtener todas las películas que sean dramas, y todas las posteriores a 2020. El resultado debe ser el siguiente:

```
Cine: Telmo
Peliculas en cartelera: [Perfect days - dirigida por: Wim Wenders (Drama): 2023, Inception - dirigida por: Christopher Nolan (Accion): 2010, Jumanji - dirigida por: Joe Johnston (Aventura): 1995]
Dramas: [Perfect days - dirigida por: Wim Wenders (Drama): 2023]
Recientes: [Perfect days - dirigida por: Wim Wenders (Drama): 2023]
```

Añade más películas de género drama y posteriores a 2020 para comprobar que tu programa funciona bien.

Contesta a la siguiente pregunta:
Si se quisieran obtener las películas por otros criterios, habría que añadir más métodos a la clase `CarteleraCine`. ¿Se te ocurre alguna manera de generalizar el diseño para que no haya que añadir un método por cada posible criterio de interés?

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 preguntas planteadas en los apartados 3 y 4.