

Examen de Prácticas de Programación Ingeniería Informática

Junio 2006

- 1) (2 puntos) Escribir en Java la implementación de los métodos

```
public void escribeMatrizEnFichero(double[][] m, String nomFich)
...
public double[][] leeMatrizDeFichero(String nomFich) ...
```

Los métodos deben servir para escribir y leer matrices de cualquier dimensión, se deja a elección del alumno la forma en que se almacenan las dimensiones de la matriz en el fichero.

El método de escritura deberá escribir los números con cuatro decimales y ocupando un número fijo de caracteres de forma que queden ordenados por columnas. El método de lectura debe ser capaz de leer ficheros generados por `escribeMatrizEnFichero` y también ficheros escritos "a mano" en los que no se respete esta ordenación por columnas.

Los métodos deberán detectar todos los errores que sea posible y lanzar las excepciones correspondientes.

- 2) (1.5 puntos) Se dispone de la clase `Alumno` ya realizada cuya interfaz es la mostrada a continuación:

```
public class Alumno {
    public Alumno(String nombre, int curso, Estudios estudios)
    {...}

    otros métodos no relevantes para el problema planteado
}
```

Además existe la clase enumerada `Estudios`:

```
public enum Estudios {Físicas, Matemáticas, Informática}
```

Se pide implementar en Java un método que pida al usuario los datos de un alumno (utilizando una ventana de lectura del paquete `fundamentos`) y retorne un objeto de la clase `Alumno` con dichos datos. El método deberá permitir al usuario reintentar la entrada de datos hasta que sean correctos, es decir, hasta que el curso sea un número entre 1 y 5 y los estudios correspondan a uno de los tres valores válidos.

- 3) (1.5 puntos) Escribir la salida por consola que se obtendría con la ejecución del método main de la clase ValorObjs. Indicar de forma clara las líneas en blanco que formen parte de la salida (por ejemplo sustituyéndolas por una línea horizontal).

```
public class MiClase {
    public double d;
    public Integer i;

    public MiClase(double d, Integer i) {
        this.d=d;
        this.i=i;
    }

    public MiClase() {
    }

    public void muestra(String msj) {
        System.out.println(msj+": (d:"+d+" i:"+i+"");
    }
}

public class ValorObjs {
    public static void main() {
        MiClase obj1=new MiClase();
        MiClase obj2=new MiClase(1.0, new Integer(2));

        obj1.muestra("obj1");
        obj2.muestra("obj2");

        MiClase[] arrayObjs=new MiClase[2];
        int[] arrayInts=new int[2];

        System.out.println("\narrayObjs[0]:"+arrayObjs[0]+
            " arrayObjs[1]:"+arrayObjs[1]);
        System.out.println("arrayInts[0]:"+arrayInts[0]+
            " arrayInts[1]:"+arrayInts[1]);

        arrayObjs[0]=new MiClase(2.0, new Integer(3));
        arrayObjs[1]=obj2;

        obj1.muestra("\nobj1");
        obj2.muestra("obj2");
        arrayObjs[0].muestra("arrayObjs[0]");
        arrayObjs[1].muestra("arrayObjs[1]");

        obj1.d=11.0;
        obj2.d=12.0;

        obj1.muestra("\nobj1");
        obj2.muestra("obj2");
        arrayObjs[0].muestra("arrayObjs[0]");
        arrayObjs[1].muestra("arrayObjs[1]");

        for(int i: arrayInts)
            i+=1;
        System.out.println("\narrayInts[0]:"+arrayInts[0]+
            " arrayInts[1]:"+arrayInts[1]);
    }
}
```

- 4) (5 puntos) Se pretende desarrollar un programa que permita gestionar el préstamo de las películas de un vídeo club. Para ello se dispone de las clases `Película` y `Cliente`, ya realizadas y cuya interfaz es:

```
public class Película {
    /** crea una película en estado no prestada */
    public Película(String título) {...}
    /** retorna el título de la película */
    public String título() {...}
    /** presta la película al cliente indicado, si la película ya
     * se encontraba prestada lanza YaPrestada */
    public void presta(Cliente cliente) throws YaPrestada {...}
    /** marca una película prestada como devuelta. Si no se
     * encontraba prestada lanza NoPrestada, y se encontraba
     * prestada, pero a un cliente distinto de "cliente" lanza
     * ClienteIncorrecto */
    public void devuelve(Cliente cliente) throws NoPrestada,
        ClienteIncorrecto {...}
}
```

No va a ser necesario utilizar ningún método de la clase `Cliente`, por lo que no se muestra su interfaz.

Se pide implementar en Java la clase `VídeoClub` con la siguiente funcionalidad:

- Atributos: cuatro listas enlazadas (`LinkedList`), cada una para almacenar las películas correspondientes a un género (ver clase enumerada `Género`). Dentro de las listas las películas deberán estar ordenadas por orden alfabético en base a su título (de dicha ordenación se encarga el método `añadePelícula`).
- método público `añadePelícula`: recibe como argumentos la película (objeto de clase `Película`) y su género (valor de la clase `Género`). Añade la película a la lista correspondiente a su género en la posición que le corresponde por orden alfabético (utilizar el método `compareTo` de los strings para comparar por orden alfabético los títulos de las películas).
- método privado `buscaPelícula`: recibe como argumentos el título de la película (`String`) y su género (valor de la clase `Género`) y busca la película en la lista correspondiente. Retorna la película buscada (objeto de tipo `Película`) o `null` en el caso de que no la encuentre. El método deberá sacar partido del hecho de que las listas están ordenadas alfabéticamente para detectar cuando una película no existe sin necesidad de recorrer toda la lista.
- método público `prestaPelícula`: recibe como argumentos el título de la película (`String`), su género (valor de la clase `Género`) y el cliente al que se va a prestar la película (objeto de la clase `Cliente`). Utiliza el método `buscaPelícula` para buscar la película y, en el caso de que exista, llama al método `presta` para la película encontrada. Si la película no existe lanza la excepción `NoExiste`. No deberá tratar la excepción `YaPrestada` (que puede ser lanzada por el método `presta` de la clase `Película`), sino que deberá dejar que se propague hacia fuera.
- En una implementación real la clase `VídeoClub` tendría también métodos para eliminar una película del video club y para anotar que una película prestada ha sido devuelta, no se pide la realización de dichos métodos con el fin de limitar la duración del examen.

Las excepciones y la clase enumerada Género se encuentran declaradas en clases aparte:

```
public class ClienteIncorrecto extends Exception {}  
public class NoPrestada extends Exception {}  
public class YaPrestada extends Exception {}  
public class NoExiste extends Exception {}  
public enum Género {Drama, Acción, Terror, Comedia}
```

Nota: en la valoración del código se tendrá en cuenta:

- que implemente la funcionalidad solicitada
- su corrección (que compile sin errores)
- el uso de las estructuras de datos y de las instrucciones de control más apropiadas para resolver cada situación particular
- legibilidad: organización, tabulación, comentarios, ...