Samsung
TECH INSTITUTE

MÓDULO 2. PROGRAMACIÓN ORIENTADA A OBJETOS CON JAVA

Relación de Problemas Nº 6

Proyecto prNotas2. (Colecciones (List y Set), i/o)

Se va a modificar el proyecto prNotas para que pueda leer la información de ficheros y guardar la información a fichero.

Eliminar de la clase `Asignatura` el constructor

```
public Asignatura(String n, List<String> entradas)
```

Incluir en la clase Asignatura los siguientes constructores y métodos.

- Un constructor `public Asignatura(String nombre)` que cree una asignatura con el nombre dado.
- Un método `public void leeDatos(String ficheroEntrada) throws FileNotFoundException` que lea los datos de los alumnos del fichero del nombre dado. Igual que el prNotas, la información de cada alumno aparece en una línea del fichero. Para ello, crear también el método `public void leeDatos(Scanner sc) throws FileNotFoundException` que lee los datos de un escáner.
- Un método `public void guardaDatos(Set<Alumno> datos, String nombrefichero) throws FileNotFoundException` que guarde los datos de los alumnos (incluida la nota) en el fichero de nombre dado. Para ello, crear también el método `public void guardaDatos(Set<Alumno> datos, PrintWriter pw)` que guarda los datos de los alumnos en un flujo.

Usar la siguiente aplicación para probar las clases:

```
import prNotas.Alumno;
import prNotas.ArbitroAlumnos;
import prNotas.Asignatura;

import java.io.PrintWriter;
import java.util.*;

import java.io.FileNotFoundException;

public class Main {


   public static void main(String[] args) throws FileNotFoundException {
      Asignatura algebra = new Asignatura("Algebra");
      algebra.leeDatos("prNotas2/datosAlumnos.txt");
      Alumno al1 = new Alumno("Lopez Turo, Manuel", "23322443K");
      Alumno al2 = new Alumno("Fernandez Vara, Pedro", "34242442J");

      try {
         System.out.println("Calificacion de " + al1 + ": "
               + algebra.getCalificacion(al1));
         System.out.println("Calificacion de " + al2 + ": "
               + algebra.getCalificacion(al2));
      } catch (RuntimeException e) {
         System.err.println(e.getMessage());
      }

      System.out.println("Media: " + algebra.getMedia());

      System.out.println("Alumnos...");
      for (Alumno alumno : algebra.getAlumnos()) {
         System.out.println(alumno + ": " + alumno.getCalificacion());
      }
      System.out.println("Malos...");
      for (String malo : algebra.getErroneas()) {
         System.out.println(malo);
      }
      System.out.println("Alumnos aprobados...");
      for (Alumno alumno : algebra.getAlumnosAprobados()) {
         System.out.println(alumno + ": " + alumno.getCalificacion());
      }

      System.out.println("Asignatura...");
      System.out.println(algebra);

      TreeSet<Alumno> setAlON = new TreeSet<>(algebra.getAlumnosAprobados());
      algebra.guardaDatos(setAlON, "prNotas2/aprobados.txt");
      algebra.guardaDatos(setAlON, new PrintWriter(System.out, true));

      TreeSet<Alumno> setAlOA =
                new TreeSet<>(new OrdenNota().theComparing(new OrdenDNI());
      setAlOA.addAll(algebra.getAlumnosAprobados());

      algebra.guardaDatos(setAlOA, "prNotas2/aprobadosOC.txt");
      algebra.guardaDatos(setAlOA, new PrintWriter(System.out, true));

   }
}
```

Proyecto prPartidos2. (Colecciones (List, Set y Map), i/o)

Se pretende modificar el proyecto prPartidos ya realizado para que los datos puedan ser tomados de un fichero de entrada y volcados en un fichero de salida o en la consola. Para ellos se van a realizar modificaciones al código ya existente.

1. En la clase `Elecciones`,

a) Definir el método `public void leeDatos(String fEntrada) throws FileNotFoundException` que lee los lados de los partidos políticos desde el fichero cuyo nombre se pasa como argumento.

b) Definir el método

```
public void
        presentaResultados(String fSalida,
                           Map<Partidos, Integer> map)
                           throws FileNotFoundException
```

que presenta los resultados de las elecciones en el fichero de salida. Para ello, definir también un método con el mismo nombre, pero donde el primer argumento se cambia por un `PrintWriter`. El formato de salida será parecido al del ejemplo:

```
P.P. : 123655, 19
P.S.O.E. : 57245, 8
IULV-CA : 25354, 3
UPyD : 8099, 1
LOS VERDES : 3197, Sin representación
...
```

NOTA: Con estos cambios, el método `public void presentaResultados(Map<Partido, Integer>)` es innecesario.

2. En la clase `EleccionesManager`,

a) Definir las variables de instancia siguientes:

`private String fEntrada` que mantiene el nombre del fichero de entrada de datos. Este método devolverá el receptor.

`private String fSalida` que mantiene el nombre del fichero de salida de datos. Este método devolverá el receptor.

`private boolean consola` que indica si se deben presentar los resultados por consola. Este método devolverá el receptor.

b) Incluir los siguientes métodos:

`public EleccionesManager setEntrada(String fEntrada)` que proporcione el nombre del fichero de entrada.

`public EleccionesManager setSalida(String fSalida)` que proporcione el nombre del fichero de salida.

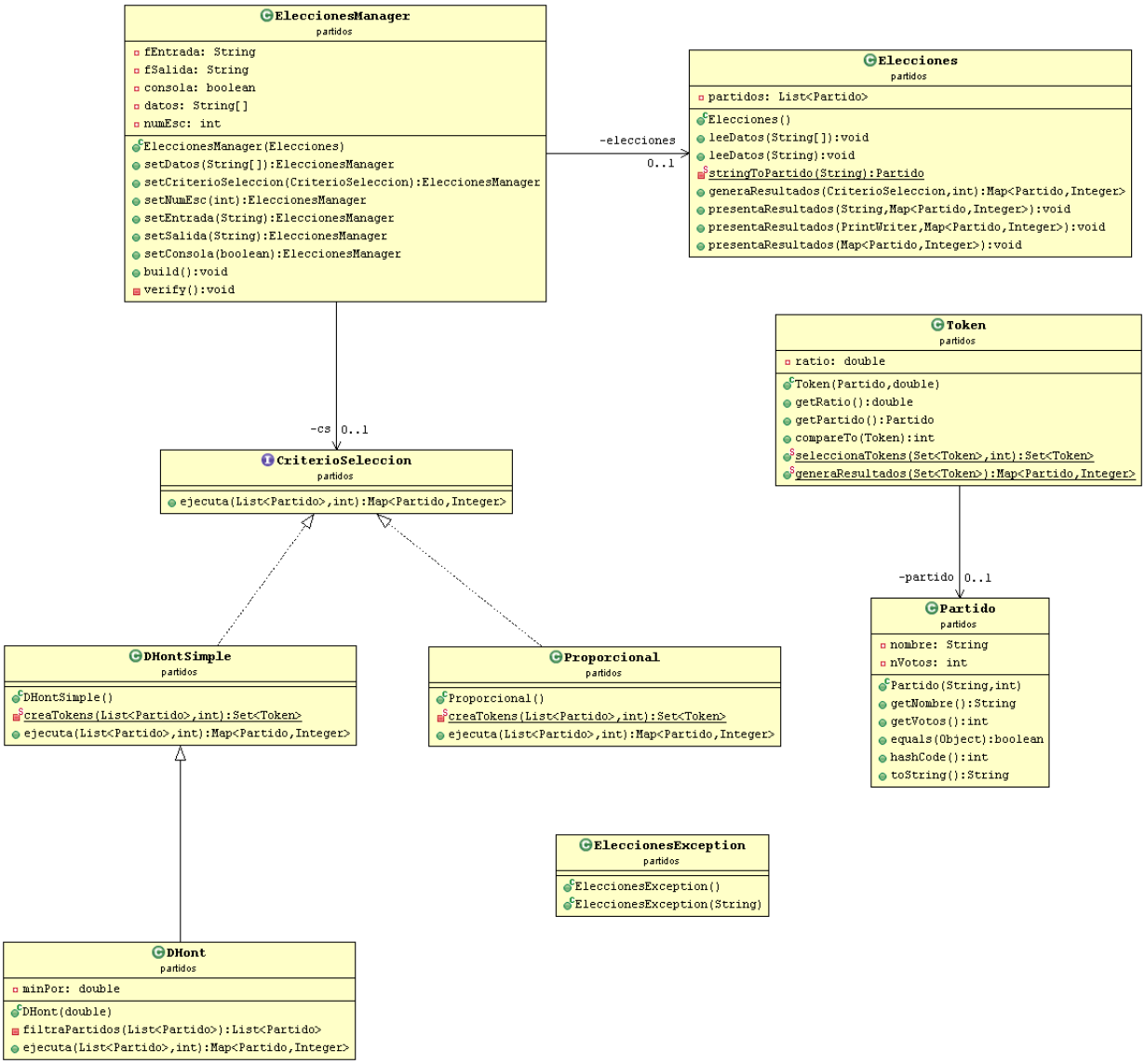
`public EleccionesManager setConsola(boolean consola)` que proporciona si un booleano que indica si se deben mostrar los resultados por la consola.

c) Modificar el privado `void verify()` para que además verifique:

Si hay una entrada y es única. Es decir, o se introducen los datos a través de un array o se introducen a través de un fichero (un debe haber).

Si hay fichero de salida, es decir `fSalida` no es `null` o hay salida por consola (pueden estar los dos).

d) Modificar el método público
`void build() throws FileNotFoundException` para que realice las siguientes acciones:
- Verifica que los datos almacenados en las variables de instancia son correctos.
- Crea una lista de partidos a partir de la información almacenada en el fichero de entrada o en el array de datos.
- Crea la distribución de escaños (`Map<Partido, Integer>`) a partir del criterio de selección, la lista de partidos y el número de escaños.
- Si se ha indicado un fichero de salida, vuelca los resultados en ese fichero.
- Si se desea mostrar en consola, se muestran los mismos resultados por la consola.



Proyecto prIndicePalabrasv2 (Colecciones (Set y Map), Scanner, io, herencia)

En este ejercicio vamos a modificar las cuatro clases del ejercicio prIndicePalabrasv1 para que tomen su información de ficheros y muestren sus resultados en un flujo dado.

Vamos a realizar las siguientes modificaciones en la clase `Indice`:

Crea el método `public void agregarDesdeFichero(String nombreFichero) throws FileNotFoundException` que lea las líneas de texto desde un fichero de nombre dado. Para ello crea también el método `public void agregarDesdeScanner(Scanner sc)` que lee las líneas desde un escáner.

Crea el método `protected Set<String> leeNoSig(String nombreFichero) throws FileNotFoundException` que lea las palabras no significativas desde un fichero de nombre dado. Para ello crea también el método `protected Set<String> leeNoSig(Scanner sc)` que lee las palabras no significativas desde un escáner.

Crea el método `public void resolver(String delim, String nombreFichero) throws FileNotFoundException` que lea las palabras no significativas desde un fichero de nombre dado y resuelva el índice teniendo en cuenta el delimitador dado.

Crea el método `public void presentarIndice(String nombreFichero) throws FileNotFoundException` que presente el índice en el fichero de nombre dado. Para ello crea el método `abstract public void presentarIndice(PrintWriter pw)` que presentará el índice en un flujo dado. Este método al ser abstracto, se redefinirá posteriormente en cada una de las subclases de `Indice`. Cambiar el método `public void presentarIndiceConsola()` para no sea abstracto y se implemente aquí usando estos nuevos métodos. Borrar las implementaciones de este método definidas en las subclases de `Indice` que ya no son necesarios.

Probar con el programa:

```
import java.io.FileNotFoundException;

import prIndicePalabrasv2.*;

public class EjIndice {
	public static void main(String args[]) throws FileNotFoundException {
		String delimitadores = "[ .,:;-[¡][!][¿][?]]+";

		Indice indice = new Indice1aLinea();
//		Indice indice = new IndiceLineas();
//		Indice indice = new IndicePosicionesEnLineas();

		indice.agregarDesdeFichero("prIndicePalabrasV2Nuevo/frases.txt");
		indice.resolver(delimitadores, "prIndicePalabrasV2Nuevo/noSig.txt");
		indice.presentarIndice("prIndicePalabrasV2Nuevo/salida.txt");
		indice.presentarIndiceConsola();
	}
}
```

Proyecto prKWIC (equals, compare, colecciones, io)

El objetivo de esta práctica es realizar un glosario de palabras atendiendo a su aparición en un conjunto de frases (*KeyWord In Context*, KWIC), desechando aquéllas que no se consideren significativas. Para ello, contaremos con dos ficheros de entrada. El primero contendrá la relación de palabras no significativas (y que, por lo tanto, no aparecerán en el listado KWIC). El segundo contendrá una relación de frases, a partir de las cuales deberemos obtener el correspondiente índice. Un ejemplo de fichero con palabras no significativas podría contener las siguientes líneas:

```
el la los las un una unos unas
y o
a ante bajo cabe con contra de desde en entre hacia hasta
para por según sin sobre tras
si no
al del
corre toma llama
```

El siguiente listado de títulos de películas podría servir como ejemplo de contenido de un fichero con las frases (una frase por línea) a partir de las cuales hay que construir un índice KWIC.

```
El color del dinero
Color púrpura
Misión: imposible
La misión
La rosa púrpura del Cairo
El dinero llama al dinero
La rosa del azafrán
El nombre de la rosa
Toma el dinero y corre
```

El índice que se desea generar debe tener el siguiente aspecto:

```
AZAFRÁN
      La rosa del azafrán
CAIRO
      La rosa púrpura del Cairo
COLOR
      Color púrpura
      El color del dinero
DINERO
      El color del dinero
      El dinero llama al dinero
      Toma el dinero y corre
IMPOSIBLE
      Misión: imposible
MISIÓN
      La misión
      Misión: imposible
NOMBRE
      El nombre de la rosa
PÚRPURA
      Color púrpura
      La rosa púrpura del Cairo
```

ROSA
El nombre de la rosa
La rosa del azafrán
La rosa púrpura del Cairo

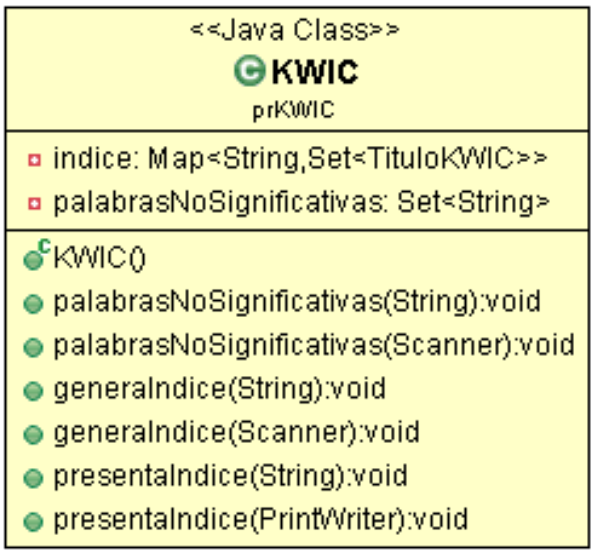
Como vemos, el índice está ordenado por palabras significativas y, por cada una de ellas, aparecen todas las frases que la contienen (ordenadas alfabéticamente.
Visto lo anterior, se pide:

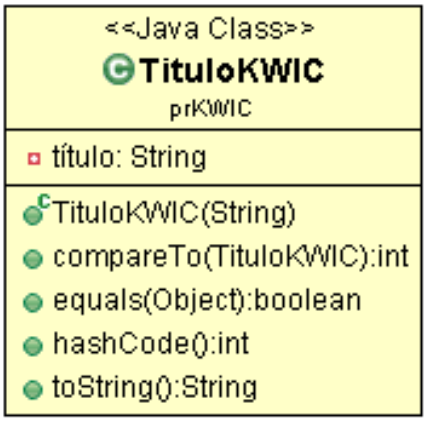
a) Definir una clase `TituloKWIC` que dé envoltura a una frase o título (de tipo `String`), y que permita ordenar y comparar frases independientemente de si éstas contienen caracteres en minúsculas o mayúsculas.

b) Definir una clase `KWIC` que incluya los métodos necesarios para:
   i. leer (y almacenar) la información de las palabras no significativas,
      `public void palabrasNoSignificativas(String)`
      `public void palabrasNosSignificativas(Scanner)`
      La información se almacenará en un `Map<String,Set<TituloKWIC>>`
   ii. generar la estructura del índice a partir de un fichero de texto (p.ej. títulos de películas) teniendo en cuenta las palabras no significativas leídas previamente,
      `public void generarIndice(String)`
      `public void generarIndice(Scanner)`
   iii. representar el índice sobre un dispositivo.
      `public void presentaIndice(String)`
      `public void presentaIndice(PrintWriter)`

   La clase KWIC deberá responder a la siguiente prueba:

```
import java.io.IOException;
import prKWIC.KWIC;

public class EjemploKWIC {
	public static void main(String[] args) {
		try {
			KWIC k = new KWIC();
			k.palabrasNoSignificativas("noClaves.txt");
			k.generaIndice("frases.txt");
			// Para presentar por pantalla
			// Quitar comentarios para que salga en consola
			/*
			PrintWriter pw = new PrintWriter(System.out, true);
			k.presentaIndice(pw);
			*/
			k.presentaIndice("salida.txt");

		} catch (IOException e) {
			System.out.println("Error: " + e.getMessage());
		}
	}
}
```


<<Java Class>>
KWIC
prKWIC
indice: Map<String,Set<TituloKWIC>>
palabrasNoSignificativas: Set<String>
KWIC()
palabrasNoSignificativas(String):void
palabrasNoSignificativas(Scanner):void
generaIndice(String):void
generaIndice(Scanner):void
presentaIndice(String):void
presentaIndice(PrintWriter):void


<<Java Class>>
TituloKWIC
prKWIC
título: String
TituloKWIC(String)
compareTo(TituloKWIC):int
equals(Object):boolean
hashCode():int
toString():String

Proyecto prCanciones (Colecciones (List y map), equals, compare, herencia, io)

Se desea construir una aplicación que permita emular un el almacenamiento de listas de reproducción de canciones, con información sobre duración, título, intérprete y género. Para ello, se definirán un tipo enumerado Genero con los valores ROCK, POP, HIPHOP, DANCE y las clases que se indicarán a continuación.

1. Créese la clase `Duracion` para representar la duración en minutos y segundos, con las siguientes operaciones:

1.1. Proporciónese un constructor sin argumentos que cree objetos de duración cero y otro constructor con dos argumentos, que proporcionen los minutos y segundos. En este caso, al menos el valor que indica los segundos debe ser válido (entre 0 y 59); en caso contrario, debe lanzarse la excepción `IllegalArgumentException`, con un mensaje adecuado.

1.2. Proporciónese un método para incrementar la duración con otra que se pasa como argumento:

```
public void incrementa(Duracion tiempo)
```

El efecto debe ser el incremento de la duración que recibe el mensaje con la que se pasa como argumento. Debe tenerse en cuenta que el número de segundos del receptor no debe superar 59. Por ejemplo,

```
Duracion d1 = new Duracion(3,40);
Duracion d2 = new Duracion(2,50);
d1.incrementa(d2);      // d1 será ahora 6 minutos y 30 segundos
```

1.3. Dos objetos de tipo `Duracion` se considerarán iguales cuando sus minutos y segundos coinciden.

1.4. Defínase un orden natural que determine que un objeto `Duracion` es menor que otro cuando representa un periodo temporal menor.

1.5. Impleméntese el método `toString()` de forma que la representación textual de una duración sea "`[mm:ss]`", donde `mm` son los minutos y `ss` los segundos.

2. Constrúyase la clase `Cancion` que mantenga información sobre la duración (de tipo `Duracion`), el título (`String`), el intérprete (`String`) y el género (del tipo enumerado `Genero`), e incluya los siguientes elementos:

2.1. Un constructor para crear instancias de `Cancion` a partir de su título, intérprete, los minutos y segundos de reproducción y el género.

2.2. Operaciones que permitan obtener la información relevante de una canció

```
public Duracion getDuracion()
public String getTitulo()
public String getInterprete()
public Genero getGenero()
```

2.3. Una noción de igualdad que establezca que dos canciones son iguales cuando coinciden el título, el intérprete y la duración.

2.4. Un método `toString()` que permita representar textualmente canciones como:

```
[mm:ss] – TITULO EN MAYÚSCULAS (Intérprete)
```

3. Defínase la clase `Reproductor`, que mantenga información sobre diversas listas de reproducción de canciones, de forma que utilice una correspondencia (`Map`) que asocie a nombres de listas (`String`), listas de canciones (`List<Cancion>`). Para esta clase, debe incluirse:

3.1. Un constructor sin argumentos que inicialice la estructura de forma adecuada.

3.2. Un método para añadir listas de reproducción a partir del nombre de la lista y el nombre de un fichero, que añada a la correspondencia que almacena las listas, una nueva entrada que asocie al nombre de la lista (primer argumento) la información almacenada en fichero indicado como segundo argumento:

```
public void anyadirLista(String nombreLista, String fichero)
```

La información sobre las canciones en el fichero se organiza por líneas, donde cada línea tiene el siguiente formato (véanse los ficheros de prueba proporcionados):

```
Título,Intérprete,minutos,segundos,GENERO
```

3.3. Un método para obtener el tiempo de reproducción de una lista de reproducción (indicada en el argumento), consistente en la suma de las duraciones de las canciones asociadas a esa lista, y otro método que devuelva el tiempo total de reproducción de todas las litas:

```
public Duracion tiempoReproduccion(String nombreLista)
public Duracion tiempoTotal()
```

3.4. Un método que vuelque sobre un fichero la información textual de todas las canciones que componen la lista indicada como argumento:

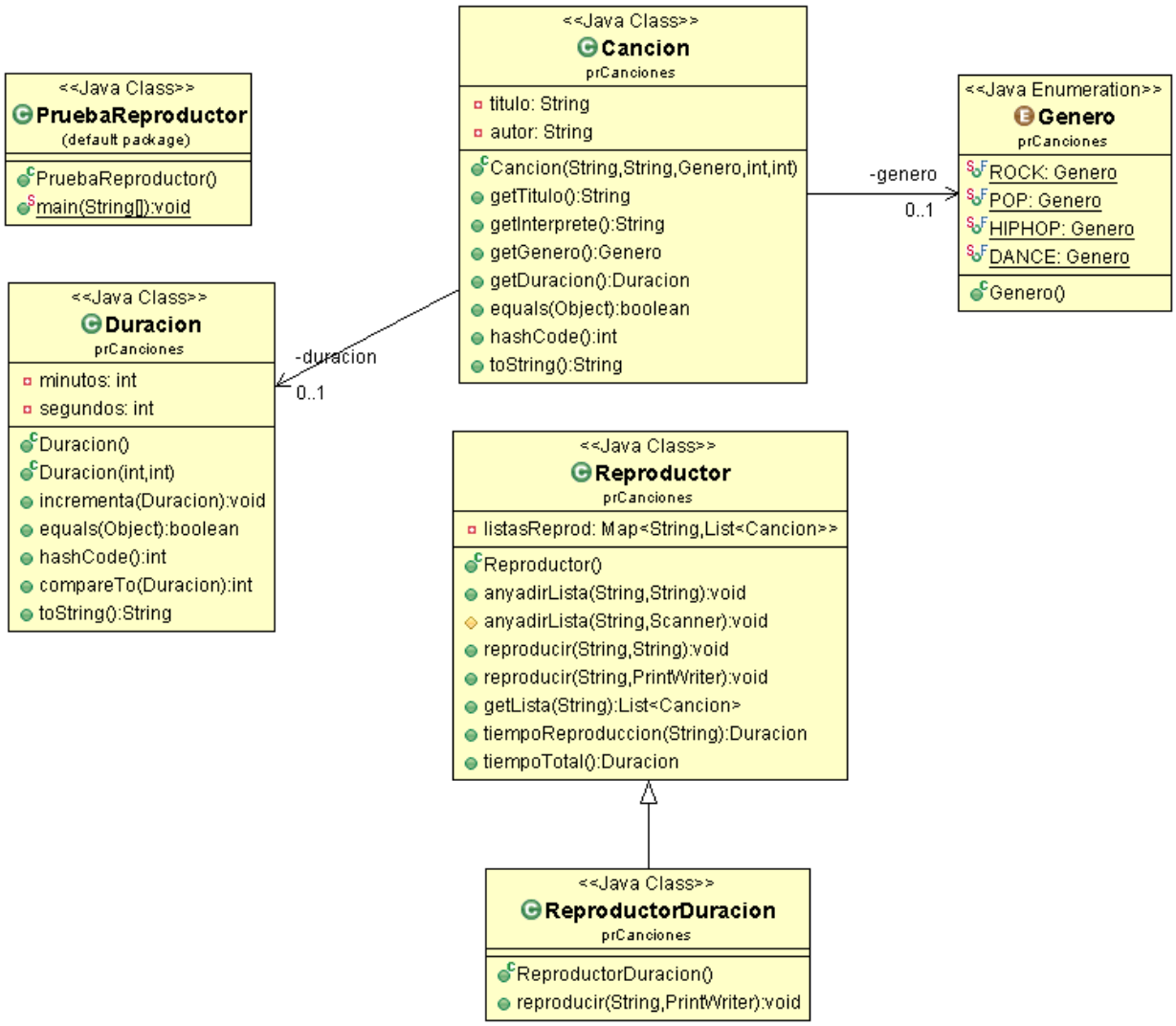
```
public void reproducir(String nombreLista, String ficheroSalida)
```

Proporciónese también un método similar que vuelque esa información sobre un `PrintWriter`:

```
public void reproducir(String nombreLista, PrintWriter salida)
```

4. Indíquese qué habría que hacer para definir una clase `ReproductorDuracion`, que se comporte como la clase `Reproductor`, pero en la que el efecto del método `reproducir` vuelque la información (sobre el fichero o sobre el `PrintWriter`), pero ordenando las canciones de la lista según la duración de las mismas de menor a mayor.

```java
import java.io.FileNotFoundException;
import java.io.PrintWriter;

import prCanciones.Reproductor;

public class PruebaReproductor {
	public static void main(String[] args) {
		Reproductor reproductor = new Reproductor();
		try {
			// A–adimos al reproductor varias listas de reproducci—n
			reproductor.anyadirLista("footing", "footing.txt");
			reproductor.anyadirLista("sobremesa", "sobremesa.txt");
			reproductor.anyadirLista("fiesta", "fiesta.txt");

			// Obtenemos por pantalla la salida de las tres listas
			PrintWriter pw = new PrintWriter(System.out, true);
			reproductor.reproducir("footing", pw);
			reproductor.reproducir("sobremesa", pw);
			reproductor.reproducir("fiesta", pw);

			// Obtenemos en un fichero la salida de la lista "footing"
			reproductor.reproducir("footing", "footingSalida.txt");

		} catch (FileNotFoundException e) { // Capturamos excepcion de fichero
			// no encontrado
			System.out.println("Error de E/S " + e.getMessage());
		} catch (RuntimeException e) { // Cualquier otra excepcion debe
			// corresponder con un error de formato
			System.out.println("Formato de fichero erroneo: " + e.getMessage());
		}
	}
}
```


<<Java Class>>
PruebaReproductor
(default package)
PruebaReproductor()
main(String[]):void
<<Java Class>>
Cancion
prCanciones
titulo: String
autor: String
Cancion(String,String,Genero,int,int)
getTitulo():String
getInterprete():String
getGenero():Genero
getDuracion():Duracion
equals(Object):boolean
hashCode():int
toString():String
-genero
0..1
<<Java Enumeration>>
Genero
prCanciones
ROCK: Genero
POP: Genero
HIPHOP: Genero
DANCE: Genero
Genero()
<<Java Class>>
Duracion
prCanciones
minutos: int
segundos: int
Duracion()
Duracion(int,int)
incrementa(Duracion):void
equals(Object):boolean
hashCode():int
compareTo(Duracion):int
toString():String
-duracion
0..1
<<Java Class>>
Reproductor
prCanciones
listasReprod: Map<String,List<Cancion>>
Reproductor()
anyadirLista(String,String):void
anyadirLista(String,Scanner):void
reproducir(String,String):void
reproducir(String,PrintWriter):void
getLista(String):List<Cancion>
tiempoReproduccion(String):Duracion
tiempoTotal():Duracion
<<Java Class>>
ReproductorDuracion
prCanciones
ReproductorDuracion()
reproducir(String,PrintWriter):void