Examen de Programación Orientada a Objetos (Plan 2014)

10 de diciembre de 2018

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

Problema 1. (10 puntos)

Se quiere definir una clase `PlayList` que permita definir listas de reproducción, sin tamaño prefijado, de canciones y podcast indistintamente. Teniendo en cuenta que las clases `Cancion` y `Podcast` implementan la interfaz `IMedia`. Se pide:

1.1.- Definir la cabecera de la clase, implementar su constructor y los métodos que permiten añadir una canción o un podcast a la lista de reproducción (`addMedia`) y devolver la lista de reproducción contenida en la clase `PlayList` (`getPlaylist`). Definir las propiedades e implementar los métodos que adicionalmente se consideren necesarios.

```
public class PlayList<Media extends IMedia> {
  private ArrayList<Media> playlist; // también LinkedList o List

  public PlayList() {
      this.playlist = new ArrayList<>();
  }

  public void addMedia(Media m){
      this.playlist.add(m);
  }

  public ArrayList<Media> getPlaylist() {
      return playlist;
  }


}

Otra opción válida de cabecera: public class PlayList<Media>
```

1.2.- Teniendo en cuenta que las clases `Cancion` y `Podcast` implementan el método `reproduce`, que permite reproducir una canción o podcast y que genera la excepción `FinReproduccion` cuando finaliza la misma. Se pide implementar la clase `ReproductorAleatorio` que recibe un objeto de tipo `PlayList` en su constructor y permite la reproducción *aleatoria en paralelo a otra actividad* de la lista de reproducción contenida en dicho objeto. La reproducción finaliza cuando se hayan reproducido todas las canciones/podcast de la lista.

```
public class ReproductorAleatorio implements Runnable {
  private ArrayList<Media> lista;

  public ReproductorAleatorio(PlayList lista) {
      this.lista = (ArrayList<Media>)lista.getPlaylist().clone();
  }

  @Override
  public void run() {
      Media m = null;
      int posicion = 0;
      Random r = new Random();
      do {
          try {
              posicion = r.nextInt(lista.size());
              m = lista.get(posicion);
              m.reproduce();
          } catch (FinReproduccion fin) {
              lista.remove(posicion);
          }
      } while (!lista.isEmpty());
  }
}
```

1.3.- Teniendo en cuenta que las clases `Cancion` y `Podcast` implementan la interfaz `Comparable` permitiendo una comparación basada en autor y título. Se pide implementar el método getOrderedPlaylistByAuthor, de la clase `PlayList`, que devuelve una lista de reproducción ordenada por autor y en caso de que los autores sean iguales por título.

Opción a)

```
public ArrayList<Media> getOrderedPlaylistByAuthor() {
    SortedSet<Media> aux = new TreeSet();
    aux.addAll(playlist);
    return new ArrayList(aux);
}
```

Opción b)

```
public ArrayList<Media> getOrderedPlaylistByAuthor() {
    SortedSet<Media> aux = new TreeSet();
    Iterator<Media> item = playlist.iterator();
    while(item.hasNext()){
        aux.add(item.next());
    }
    return new ArrayList(aux);
}
```

Opción c)

```
public ArrayList<Media> getOrderedPlaylistByAuthorC() {
    SortedSet<Media> aux = new TreeSet();
    for(Media m: playlist){
        aux.add(m);
    }
    return new ArrayList(aux);
}
```

Opción d)

```
public ArrayList<Media> getOrderedPlaylistByAuthor() {
    SortedSet<Media> aux = new TreeSet();
    for(int i=0; i<playlist.size(); i++){
        aux.add(playlist.get(i));
    }
    return new ArrayList(aux);
}
```