Assignment 4

*Start on Mar 18, complete by Friday, 25 Mar end of day Montreal time zone. *

Problem Statement

Using the principles, mechanisms, and techniques seen in Chapter 5 (Unit Testing) and 6 (Composition) of the book, design and write the code necessary to fulfill the requirement. This assignment is within the same domain of A4 but you do not need to consider the constraint in previous assignment. Please make changes on the base code included in the attachment. *Your solution should continue to respect principles of good design seen in prior chapters*.

1. Modify the code base to allow a client to dynamically specify any Playable object for the library to use as a "default playable". [5 points]
2. Consider the following methods that change the state of PlayList: addPlayable, removePlayable, shuffle and setName. Add two more methods: undo and redo. The undo() method reverts the state to the earlier one before the last state-modifying method was executed. Calling undo() multiple times consecutively undoes the methods in the reverse order of how they were executed. The method undo() does nothing if there are no methods to be undone. Similarly, redo() will execute the last method that was undone and calling multiple redo() methods executes the undone methods in the reverse order of how they were undone. The method redo() does nothing if there are no undone methods to be redone. If redo() is called right after a state-modifying method, it repeats the execution of that method. For instance, redo() after calling removePlayable will remove the playable at the same index if the index is not out of the bounds of the playlist. If another state-modifying method is called after undo(), redo() cannot redo any undone actions. Instead, it repeats the last method. Note that the state-modifying methods refer to the four methods mentioned at the beginning, and undo() and redo() do not consider methods other than those four. [7 points]
3. Use JUnit to create unit tests and test all functionality of requirements 1 and 2, including invalid cases. Demonstrate the line and branch coverage of your test in the supporting pdf file. Make sure your design of the tests follows the principles seen in class. [3 points]

Deliverables

1. The submission must include a zip file of the following structure:

```
260900000.zip
    └──src
        |       └──*.java                          all your source code
        |       |
    └──test
        |       └──*.java                          all your tests
        └──*.pdf                                   pdf of your design and diagram
```

2. A pdf file of a short description (max. 100 words) of the design of your solution (major decisions, techniques used, trade-offs, etc.), supported by at least one UML **sequence** diagram created with JetUML or other tools.
3. Submit your work on MyCourses -> Assignment -> Assignment4.

Rules of Carrying Out Assignment

- The focus of the assignment is to evaluate your application of design techniques while respecting all the requirements in the problem statement. To this end, you are expected to come up with an original design when the requirements don't specify a particular design.
- The goal of this assignment is to arrive at a solution having explored the design space and to understand the trade-offs.
- Remember that there's no single optimal solution. However, you should be able to justify the design choices you made during

the assignment both by your code and your description file.
- Try to think of different use cases for your code, and how your solution could accommodate them. Writing your own client code can help identify design issues. For example, you can use real world scenarios to approach the problem statements. However, you don't need to implement features other than those in the problem statement, and you are not expected to implement a perfect solution for all real world scenarios.

Assessment

The TA evaluation of your solution will focus on three aspects:

1. How well does the delivered code satisfy the requirements in the problem statement?
2. How well are the design concepts, principles, and techniques are applied and explained in the description file (including the trade-offs of different options)? Although the focus will be on techniques from Chapter 3, you can be penalized for violating design principles from earlier chapters.
3. How readable is the source code (through its style and documentation)?

Policy on Code Reuse

1. All design activities are prepared so as to be completed without any need to reuse external code; **"External code" means any code not part of the course samples or Java/JavaFX class libraries;** If you are pining for the a perfect score, please note that using external code will not give you any kind of competitive advantage to help you reach this level.
2. You can reuse external code only for implementation of functionalities that clearly go beyond the requirements; All major design decisions required to fulfill the requirements should be yours and yours only. In case of doubt either refrain from using external code or clarify the situation with your teaching assistant (TA). If you reuse external code it is your responsibility to ensure it does not take away from your required contributions.
3. If you reuse external code it is your responsibility to understand how it works; External code included with a solution is considered part and parcel of the solution: any bug or design flaw caused by external code will be considered a problem with the solution.
4. If you reuse external code you must clearly identify it and locate it in a separate package named `external`, and code files must clearly bear the origin of the external code.
5. If you reuse external code you are responsible for looking up its license and respecting it (in particular any attribution clause).
6. If you reuse external code you are responsible for documenting how it works to the extent where it integrates with your solution. The TAs can be assumed to have knowledge of the Java class libraries, but not external code.
7. Use of external code must be credited anywhere it appears. For example, if you paste a code fragment from Stack Overflow into a comment, this needs to be credited even if it might not appear in the final solution.
8. Any unaccredited contribution of external code will be considered a breach of academic integrity. In brief, reusing code is a big responsibility, not to be taken lightly. Please read the article Surviving Software Dependencies for more insights.