Seneca College

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

Workshop - 1

Due Date: 24th January 2024

INSTRUCTIONS

- *This workshop must be completed individually without any outside collaboration. All work must be your own. Copying or reproducing the work done by others (in part or in full) or letting others to copy or reproduce your own work is subject to significant grade reduction or getting no grade at all and/or being treated as academic dishonesty under the College's Academic Dishonesty Policy.*
- *This is an out of class workshop and you are required to complete this workshop on your own time, unless otherwise specified by your instructor.*
- *Your application must compile and run upon download to receive any mark. If the corresponding program fails to compile and run it will receive ZERO grade.*
- *To submit the workshop, please follow the Submission Guideline provided at the end of this document.*
- *You must submit the workshop by the due date mentioned above. Late submissions policy is specified in the Academic Procedures for Evaluations document available through the class plan on blackboard.*
- *Total mark = 7.5% of the final grade.*

- *Make sure you go through/ read carefully all the requirements given below and understand them very clearly before starting coding you solutions.*

Musical Instruments

Design and implement the following interface and class hierarchy (shown in Figure 1) in java to demonstrate the relationships and functions among various musical instruments in a typical musical instrument shop.

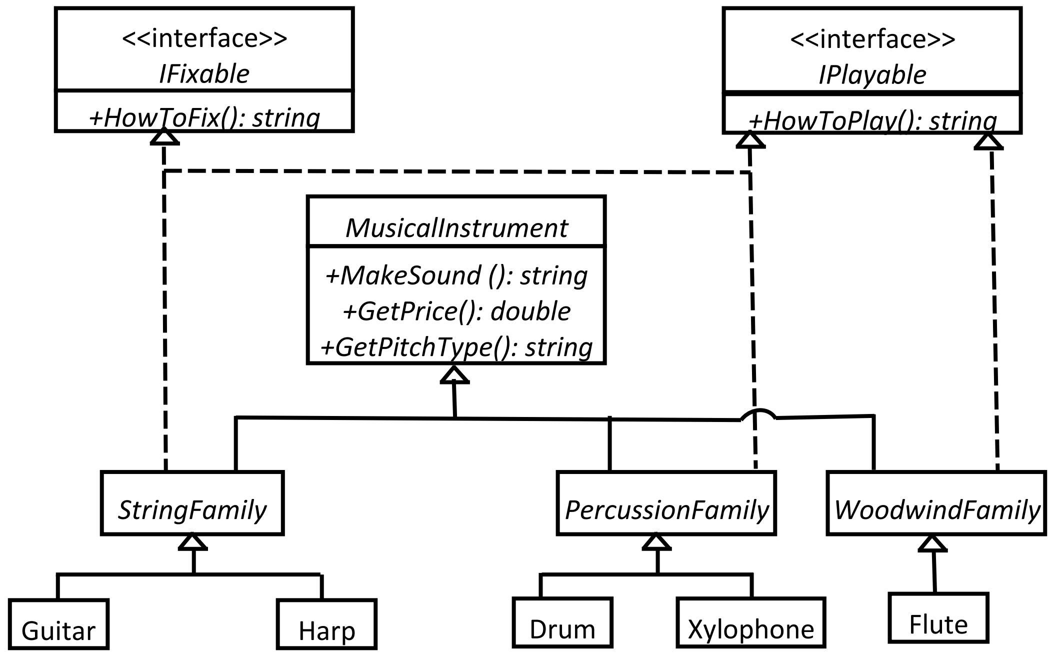

Figure 1: `IFixable` and `IPlayable` are two interfaces. `MusicalInstrument`, `StringFamily`, `PercussionFamily`, and `WoodwindFamily` are all abstract classes. Guitar, Harp, Drum, Xylophone, and Flute are concrete classes.

Most information handled and/or returned by interface and abstract class methods are presented in Table 1. Use this table for information to be returned by your method implementations within respective class.

Table 1: Information on musical instruments

Instrument	Make sound	Price	How to play	How to fix	Pitch Type
Drum	vibrating stretched membrane	$349.50	by hitting the membrane	replace the membrane	Sonic pitch
Flute	guiding a stream of air	$74.90	by blowing into the flute	*N/A: it cannot be fixed*	Fundamental pitch is middle C
Guitar	vibrating strings	$199.00	by strumming the strings	replace the strings	Low to high pitch
Harp	vibrating strings	$255.00	with the thumb and first three fingers	replace the strings	Has seven levels of pitch

Xylophone	through resonators	$49.00	with two mallets	replace bars	Each bar produces different pitch

Your implementation should:

i) [***Requirement 1***] receive the price value for each instrument from the user to create an object of that instrument (other information for that instrument object e.g., make sound, how to play, how to fix, and pitch type can be hard coded as per Table 1).

ii) [***Requirement 2***] display *How to Play, How to Fix, Pitch type* and *Price* information for the most expensive instrument through comparing objects of all instruments.

iii) [***Requirement 3***] display the names of all instruments (Drum, Flute, Guitar, Harp, Xylophone) in descending order of price. You must override the ToString() method to display the name of an instrument.

iv) [***Requirement 4***] receive an instrument family name (e.g., String family, Percussion family or Woodwind family) from the user, and display how each instrument of that given family makes sound.

Your implementation must also satisfy following requirements:

v) [***Requirement 5***] You must demonstrate the use of `Comparable` interface and implement its `compareTo()` method (or `Comparator` interface and implement its `compare()` method) to find the instrument object with the highest price and sort them.

vi) [***Requirement 6***] You must demonstrate the use of an array of objects (e.g., an array of type `MusicalInstrument` to hold objects of each concrete type of musical instrument). Use the `compareTo()` method that you may should overridden in `MusicalInstrument` class (or `compare()` method that you might have implemented in a separate helper class) to find the instrument object with the highest price and sort them in the collection. No operation on the instruments should be hard coded – meaning all operations must be done programmatically.

vii) [***Requirement 7***] Your program must demonstrate proper modularization of code, java industry standard with proper comments, indentations, and naming convention.

viii) [***Requirement 8***] You must include the following information as commented at the beginning of your main class file.

- Workshop: 1
- Name: <Your Full Name>
- Id: <Your Seneca Id>

ix) [***Requirement 9***] You must test your application for multiple runs with different price values for all instruments. Change the price value for each instrument to show that your application works even if the price of instruments is updated.

x) ***[Requirement 10]*** Your code must follow the MVC design pattern.

Sample Output: On the above information of musical instruments in Table 1, the output of a test run should be as follows:

```
--: Requirement 1 :--
Enter the price for Drum: <349.5>
Enter the price for Flute: <74.9>
Enter the price for Guitar: <199>
Enter the price for Harp: <254>
Enter the price for Xylophone: <49>

--: Requirement 2 :--
The most expensive instrument is: Drum
Drum’s cost is: $349.50
Drum is played: by hitting the membrane
Drum fixing: replace the membrane
Drum pitch type: Sonic pitch

--: Requirement 3 :--
Instruments in price descending order:
[Drum, Harp, Guitar, Flute, Xylophone]

--: Requirement 4 :--
Enter an instrument family: <Percussion>
Xylophone makes sound through resonators.
Drum makes sound vibrating stretched membrane.
```

Workshop Header

/***
Workshop #
Course:*<subject type> - Semester*
Last Name:*<student last name>*
First Name:*<student first name>*
ID:*<student ID>*
Section:*<section name>*
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:*<submission date>*
***/

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention.
- Do Not have any debug/ useless code and/ or files in the assignment.

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- Design should be discussed/ created during the lab. 10%.
- Complete the code behind for the requirements plus if any other helper classes or functions required . 60%
- Submit a **reflect.txt** file with the submission. 15%
- Video submission explaining core code pointers and showing the full working application (3 minutes max). 15%
- All submission goes to Black Board.
- Your submission should include
  - Video file with audio
  - Reflect.txt file
  - Complete zipped project.
- Late submissions would result in additional 10% penalties for each day or part of it.

Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the workshops, but the final solution may not be copied from any source.