METRO STATE UNIVERSITY

ICS 141 - 02: Problem solving with programming
Spring 2023

Assignment 4: Inheritance and Polymorphism

Out: Wednesday, March 1st, 2023
Due: Wednesday, March 29th, 2023

Total points: 50

In this assignment, you will implement a Java application, called **MyMusicApp,** which can be used to manage a playlist. The application is to be implemented using three classes, a superclass **Music,** which will be provided. Base on the class lecture you'll create two subclasses. The description of these classes is below.

To receive full credits, create a video recording less than 10 minutes with your **face on camera** describing your code step by step. Upload the java package as well.

Problem Description

`class Music`

- Download the **MyMusicApp** project from D2L week#7 folder and import it into your Eclipse IDE.
- Review the **Music** class details to familiarize yourself with the implementation.
- Base on your knowledge of Objects, add a **toString()** method to the class.

Note:

Make sure that the completed assignment passes the "is-a" relationship test of superclass and subclasses*:* Note that all of these are valid superclass and subclasses relationship.

- Relative (superclass); Sister, Brother, Aunt, Uncle (subclasses)
- Parent (superclass); Mom, Dad (subclasses)
- Appliance (superclass); Stove, Refrigerator, Oven, Dishwasher (subclasses)
- Animal (superclass); Dog, Cat, Hamster, Tiger (subclasses)
- Publication (superclass); Book, Magazine, Newspaper (subclasses)

`class SubClass1`

Choose any genre of music (i.e. Pop, Jazz, Rock etc.); however, it MUST **extend** the parent class.

- Your subclass should contain 2 – 3 additional variables relevant to the genre.
- The subclass must have a parameterize constructor that calls the Music class constructor and pass the necessary variables to initialize the parents instance variables. The constructor should also initialize the variables within Suclass1.
- Override the **toString()** method of the parent class and print the details of both the parent class and Subclass1.
- Override the **increaseVolume()** method in the Music class and change the implementation so that it is specific for this class.
- Implement a method call **nameThatTune(),** that returns the title of the Subcass1.

`class SubClass2`

Choose any genre of music (i.e. Pop, Jazz, Rock etc.); however, it MUST **extend** the parent class.

- Your subclass should contain 2 – 3 additional variables relevant to the genre. This must be different from the previous variable established for SubClass1.
- The subclass must have a parameterize constructor that calls the Music class constructor and pass the necessary variables to initialize the parents instance variables. The constructor should also initialize the variables within Suclass2.
- Override the **toString()** method of the parent class and print the details of both the parent class and subclass1.
- Implement a method call **nameThatTune(),** that returns the title of the Subcass2.

`class PlaylistDriver`

Your driver class should do the following actions:
Instantiate three objects:

- One of data type **Music**.
- One of data type **Subclass1**. (i.e. Pop, Jazz, Rock etc.)
- One of data type **Subclass2**, (i.e. Pop, Jazz, Rock etc.)
- Call/invoke all the method in the ***Music class***.
- Call/invoke all the method in **Subclass1**.
- Call/invoke all the method in **Subclass2**.
- Call/invoke the **Music** methods on **Subclass1** and vice versa. Describe what happens?
- Instantiate a **Music** object and assign it to the **Subclass1** reference variable.
    - Call all the methods in the **Subclass1** again.
    - What happens? Add comments in your code to describe any errors the code encounters. Comment out the line of code with the error.
- Instantiate a **Subclass2** object and assign it to the **Music** reference variable.
    - Call the methods in the **Music** class again.
    - What happens? Add comments in your code to describe any errors the code encounters. Comment out the line of code with the error.

Note: Hover your mouse over any code underline with a red line to display the error.

`Important Requirements`

- The output of our program must be nicely formatted.
- The programs should display your name.
- At the top of the program include your name, a brief description of the program and what it does and the due date.
- **Add appropriate comments to your code.**
- All code blocks must be indented consistently and correctly. Blocks are delimited by opening and closing curly braces. Opening and closing curly braces must be aligned consistently.
- Variable names should convey meaning.
- The program must be written in Java and submitted via D2L.

Submission Instructions

Follow the following steps to upload your code to D2L:

- Create a java project and call it <yourLastName>Assignment4 (e.g., mine will be called DillonAssignment4)
- Create one.java files to solve the problem described above. Export your .java file into **a zip** file using Eclipse using the following steps:
  - In Eclipse Project Explorer, right click on the src folder of the project and click on Export.
  - Choose General then Archive File and click Next.
  - Use the Browse key to choose a folder to store the archive file on your hard drive and give the file the same name as your project (e.g., DillonAssignment4.zip), then click Save, then click Finish.
  - Upload the **.zip** file you created to the D2L folder called Assignment4.
  - It is important that you upload only **one** zip file. Your assignment will not be graded if you upload individual .java files to the drop box.