**CSE 220 Homework Assignment 5 (Due 4/24/23)**

**1) (50 pts)** *(Polymorphism)* This problem deals with polymorphism in Java.   
a. Explain what the term polymorphism means in object-oriented programming.  
b. Explain why polymorphism is useful when implementing algorithms that sort elements within a hierarchy or search for a specific element within a collection.  
c. Assume the following classes and interface are defined:  
- *Abstract* Class **Musician**, which has methods **void practice()** and **void getPaid(double)**  
- Class **Instrumentalist**, which extends *Musician*, and has the additional method **void play()**.  
- Class **Singer**, which extends *Musician*, and has the additional method **void sing()**.

Assuming all methods have been implemented correctly, and the class constructors for Employee and Agent have no arguments, indicate which of the following snippets of code (i. – v.) will work correctly, and which will not. Explain your answers.  
**i. Musician m1 = new Musician();  
 m1.practice();  
ii. Instrumentalist i1 = new Instrumentalist();  
 i1.getPaid(500.00);  
iii. Singer s1 = new Singer();  
 s1.play();  
iv. Musician m2 = new Instrumentalist();  
 m2.practice();  
v. Musician m3 = new Singer();  
 m3.sing();**

**2) (50 pts)** *(JavaFX and Events)* For this problem you are going to be assembling a collection of shapes in a JavaFX scene and adding a simple interactive control.  
a. You are first to draw a 2d house in a JavaFX scene with the following properties:  
- The house needs a body/trunk, which should be rectangular.  
- The house needs a roof, which should be triangular.  
- The house needs a door, which should be rectangular.  
- The house needs a minimum of two windows, which should be rectangular or circular.  
- All of the house components should have distinct fill colors.

You can include other elements in your scene if you like, such as landscape, sky, etc.

b. You are to add *at least one* control to your scene that allows you adjust a property or element of the house in some fashion. Examples include, but are not limited to, altering the size of the entire house in response to a slider, adding a color control for one or more house components (see attached example .png), or adjusting the size/shape of the house windows, etc. You must employ *at least* one of the following in your code: i. An event-handler, ii. Property-binding, or iii. A ChangeListener.  
  
**Responses to Problems #1 should be in .doc(x) or .pdf format. Upload this file and a .zip file containing your .java files for problem 2 to Blackboard *separately*.**

**Both files should have the filename “LN\_FN\_5” where LN is your last name and FN is your first name.**

1. Poly in Greek means many, and morph means form. Polymorphism allows objects of different types to be treated similarly by inheritance. Polymorphism uses inheritance perform different tasks. This allows us to perform a single task in many ways.
2. It will help programmers to reuse the codes once written you can implement polymorphism and save a lot of time. When we are sorting elements within a hierarchy polymorphism allow us to use a single sorting algorithm that can be applied to many types of objects. The sorting algorithm could be designed to work with many types of classes in this case ‘class (car model)’ allowing us more efficient and flexible implementation. When we are searching for a specific element within a collection, polymorphism allows us to use a single search algorithm that can be used to many types of objects. Using polymorphism, the search algorithm could be designed to work more efficiently to find a single element.
3. The musician is an abstract class that’s mean it cannot be instantiated. It can only be use as a base class for other inherit classes.

ii. The instrumentalist is a concrete class that inherence(extends) the musicians class, it inherits the method from its superclass, including **getPaid(500.00).**

iii. The number three looks incorrect to me because singer is missing the play () method that instrumentalist has. If we run this code, there is going to be an error message because the singer class have only the sing method does not play. We cannot call method that we didn’t declare it yet.

iv. Number four looks good to me because the instrumentalist is a musician, we can declare musician m2 as a second variable inherited from the instrumentalist class. When m2.practice() is called we are overriding the practice() method in the instrumentalist class.  
v. This one is very similar to the third one, singer does not have the sing() method. Every time we call this method, it is going to give us an error message because there is not a sing method for musician 3.