Week7Lab – 10 pts

Pre-lab questions

1. Create an inheritance hierarchy (class names only) for Currency that consists of several values of bills and several values of coins.

class Currency {

String description = "Unknown Currency";

public String getCurrencyDescription() {

return description;

}

public void bills() {

double[] value = { 1.00, 2.00, 5.00, 10.00, 50.00, 100.00 };

for (int i = 0; i < value.length; i++) {

System.***out***.println("US Currency bills: " + value[i]);

}

}

public void coins() {

double[] coins = { 0.1, 0.5, 0.10, 0.25 };

for (int i = 0; i < coins.length; i++) {

System.***out***.println("US Currency coins: " + coins[i]);

}

}

public class UKCurrency extends Currency {

public UKCurrency() {

description = "UK Currency";

}

*@Override*

public void bills() {

double[] value = { 5.00, 10.00, 20.00, 50.00 };

for (int i = 0; i < value.length; i++) {

System.***out***.println("UK Currency bills: " + value[i]);

}

}

*@Override*

public void coins() {

double[] coins = { 0.1, 0.2, 0.5, 0.10, 0.20, 0.50, 1.0, 2.0 };

for (int i = 0; i < coins.length; i++) {

System.***out***.println("UK Currency coins: " + coins[i]);

}

}

}

public class IndianCurrency extends Currency {

public IndianCurrency() {

description = "Indian Currency";

}

*@Override*

public void bills() {

double[] value = { 10.00, 20.00, 50.00, 100.00, 200.0, 500.0, 2000.0 };

for (int i = 0; i < value.length; i++) {

System.***out***.println("Indian Currency bills: " + value[i]);

}

}

*@Override*

public void coins() {

double[] coins = { 1.0, 2.0, 5.0, 10.0 };

for (int i = 0; i < coins.length; i++) {

System.***out***.println("Indian Currency coins: " + coins[i]);

}

}

}

public static void main(String args[]) {

Currency c = new Currency();

Currency u = c.new UKCurrency();

Currency r = c.new IndianCurrency();

c.getCurrencyDescription();

c.bills();

c.coins();

u.getCurrencyDescription();

u.bills();

u.coins();

r.getCurrencyDescription();

r.bills();

r.coins();

}

}

Choose one of the following to develop into a program that will use polymorphism. For the sake of time, you must have a minimum of four classes in your coded solution (1 has main), however, your inheritance hierarchy may be as large as you like. Once chosen, do the following:

Understand the problem (restate in your own words, make any assumptions clear):

I understand the problem where I have to make musical instruments class in that I have to use inheritance which will extends with some other class like drum, guitar, Piano, Violin. If the main class want to hear piano sound it will print only piano string as output. Same for others if the musical instrument object calls guitar it will print only guitar as string.  
  
Inheritance hierarchy (class names only) with arrows showing the relationships:

Music (Parent Class)

Children Classes

Violin

MusicalInstruments

Drum

Guitar

Piano

SoundTesting (Main class)

UML diagrams of the classes you will code, including the one with main:

Graphical user interface, application, Teams

Description automatically generated  
  
Pseudocode of any non-trivial methods in each class (no pseudocode needed for basic setters and getters or no args constructors):

In MusicalInstruments class

I created musicInstruments method which takes Music class and object as parameter. Using that parameter it will call the method of that class which it is called.

I have created abstract class Music which has sound method as abstract.

Then classes Drum, Guitar, Piano, and Violin are extends from class Music and overrides the method called sound.

In SoundTesting class, I have created main method and create class instances to call the methods.   
  
Name of files (.java) submitted:

MusicalInstruments.java SoundTesting.java  
  
Explanation of how you made use of polymorphism in your solution:

I create the musicalInstrument method to accept the Music object as a parameter. Inside the method we call the sound method of that object. As long as the object type is Music or superclass of Music, the compiler is happy.

If the object has a type of Drum, it will print Boom. Same for others.

This type of polymorphism is referred as overriding. The sound method is common in other classes. But print the different outputs like Boom for Drum.

Overriding is when you create a different implementation of the exact same instance method (identical method signature) in a related class.

At runtime, the method of the object type is chosen. This is why overriding is also referred to as runtime polymorphism.

Overriding is achieved by providing a different implementation of a method in a child class (subclass), which is defined in its parent class (superclass).

Musical Instruments

Represent musical instruments in a class hierarchy. Think about what various instruments have in common when planning the hierarchy. For this lab it is OK to just use Strings to represent the sounds an instrument makes. For example, a Drum could print “boom” and a guitar could print “strum”. Create a collection of various instruments and use polymorphism to print the appropriate sound for each.

Marine Life

Lots of different things live under the sea. Represent a selection of them in a class hierarchy. Think about what they have in common when planning the hierarchy. For this lab it is OK to give each of them a “Catch Phrase” they say when called. For simplicity, you can write this in the toString() method. Create a collection of various marine life and use polymorphism to print the catch phrase for each thing is the list.

Games Again

Choose a game genre such as fantasy or space and brainstorm several enemies or bad guys for a game in that genre. Represent these in a class hierarchy. Think about what they have in common and how they differ. For this lab it is Ok to give each of them an attack type and damage value that is randomly generated within a given range. The attack type should be unique to each class of enemy. Create a collection of enemies and use polymorphism to print out the attack and damage for each.