**Centennial College**

**COMP 228: Java Programming**

**LAB #3 – Using Inheritance and Polymorphism**

**Studen**t: Aryan Patel(301226774)

Due Date: **Week 7**.

Purpose: The purpose of this Lab assignment is to:

1. Practice the use of Inheritance
2. Practice the use of Polymorphism.

References: Learning materials for weeks 5 and 6, textbook, and other references (if any)

Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students.

YOU NEED TO SUBMIT THE FOLLOWING 2 DOCUMENTS IN THE DROPBOX TITLED LAB3:

1. THE FIRST ONE IS A WORD DOCUMENT. USE THIS DOCUMENT AND ADD SCREEN SHOTS OF THE RUNNING STATE OF EACH EXERCISE (If there are more than 1 exercise). DO NOT DELETE THE QUESTIONS. THE SCREEN SHOTS SHOULD FOLLOW EACH QUESTION AND COVER ALL THE ASPECTS/FUNCTIONALITIES OF EACH EXERCISE. AFTER THE SCREEN SHOTS PLEASE COPY THE CODE FROM THE CODE WINDOW AND PASTE THE COMPLETE CODE. DO NOT GIVE ME SCREEN SHOTS OF THE CODE. DO NOT ZIP THIS FILE AND KEEP IT SEPARATE FROM YOUR ZIPPED PROGAM FILE.

2. SUBMIT ALSO ONE ZIPPED PROJECT FILE THAT CONTAINS ALL THE EXERISES SEPARATELY INTO THE SAME DROP BOX.

This material provides the necessary information you need to complete the exercises.

You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228Labnumber**

Example: **JohSmith\_COMP228Lab3**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228Labnumber.zip**

Example: **JohSmith\_COMP228Lab3.zip**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

## Exercise 1

Write a Java application that implements different types of insurance policies for employees of an organization.

Let **Insurance** be an abstract superclass and **Health** and **Life** two of its subclasses that describe respectively health insurance and life insurance.

The **Insurance** class defines an instance variable of type **String** to describe the **type of insurance** and an instance variable of type **double** to hold the **monthly cost** of that insurance.

Implement the **get** methods for both variables of class **Insurance**. Declare also two **abstract** methods named **setInsuranceCost()** and **displayInfo()** for this class.

The **Life** and **Health** class should implement **setInsuranceCost** and **display** methods by setting the appropriate monthly fee and display the information for each insurance type.

Write a driver class to test this hierarchy. This application should ask the user to enter the type of insurance and its monthly fee. Then, will create the appropriate object (Life or Health) and display the insurance information.

As you create each insurance object, place an **Insurance** reference to each new **Insurance** object into an array. Each class has its own **setInsuranceCost** method. Write **a polymorphic** screen manager that walks through the array sending **setInsuranceCost** messages to each object in the array and displaying this information on the screen.

(3 marks)

Driver

package exercise1;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JOptionPane;

public class Driver {

public static void main(String[] args) {

// TODO Auto-generated method stub

String type;

double cost;

String user = "";

List<Insurance> insuArray = new ArrayList<>();

while(!user.equalsIgnoreCase("exit"))

{

type = JOptionPane.showInputDialog("Insurance Type Life or Health");

cost = Double.parseDouble(JOptionPane.showInputDialog("What is Monthly cost of this insuarnce"));

if(type.equalsIgnoreCase("health"))

{

Insurance temp = new Health(type);

temp.setMonthlyCost(cost);

insuArray.add(temp);

temp.displayInfo();

}

else if(type.equalsIgnoreCase("life"))

{

Insurance temp = new Health(type);

temp.setMonthlyCost(cost);

insuArray.add(temp);

temp.displayInfo();

}

user = JOptionPane.showInputDialog("Type \" exit\" if you want to exit or type anything");

}

}

}

Insurance class

**package** exercise1;

**public** **abstract** **class** Insurance {

**private** String insuranceType;

**protected** **double** monthlyCost;

**public** Insurance(String type)

{

**this**.insuranceType = type;

}

**public** String getInsuranceType() {

**return** insuranceType;

}

**public** **double** getMonthlyCost() {

**return** monthlyCost;

}

**public** **abstract** **void** setMonthlyCost(**double** cost);

**public** **abstract** **void** displayInfo();

}

Life Insurance class

**package** exercise1;

**public** **class** Life **extends** Insurance {

**public** Life(String type) {

// **TODO** Auto-generated constructor stub

**super**(type);

}

@Override

**public** **void** setMonthlyCost(**double** cost) {

// **TODO** Auto-generated method stub

**this**.monthlyCost = cost;

}

@Override

**public** **void** displayInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Insurance Type: "+**this**.getInsuranceType()+"\n"+"Monthly cost: "+**this**.getMonthlyCost());

}

}

Health insurance class

**package** exercise1;

**public** **class** Health **extends** Insurance {

**public** Health(String type) {

**super**(type);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** setMonthlyCost(**double** cost) {

// **TODO** Auto-generated method stub

**this**.monthlyCost = cost;

}

@Override

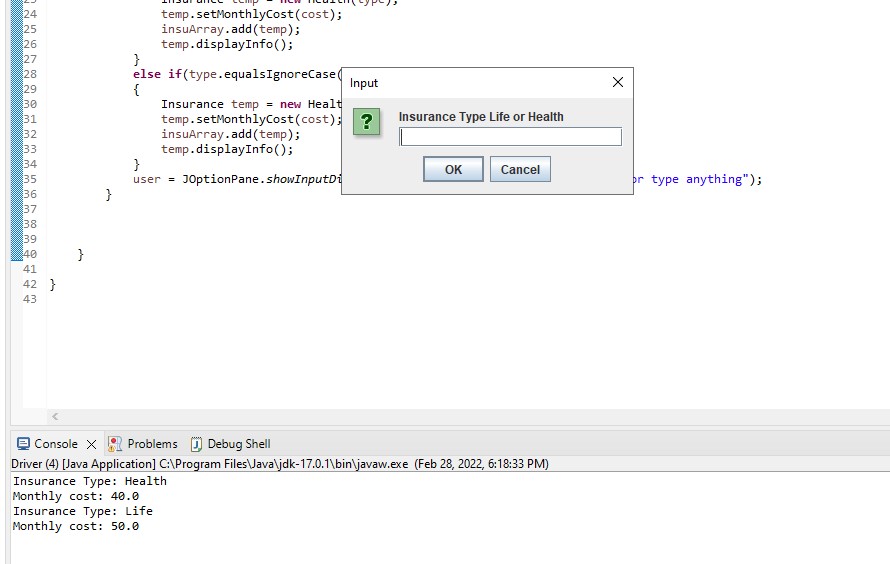
**public** **void** displayInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Insurance Type: "+**this**.getInsuranceType()+"\n"+"Monthly cost: "+**this**.getMonthlyCost());

}

}



**Exercise #2:**

Create an abstract class called GameTester. The GameTester class includes a name for the game tester and a boolean value representing the status (full-time, part-time).

Include an abstract method to determine the salary, with full-time game testers getting a base salary of $3000 and part-time game testers getting $20 per hour.

Create two subclasses called FullTimeGameTester, PartTimeGameTester. Create a console application that demonstrates how to create objects of both subclasses. Allow the user to choose game tester type and enter the number of hours for the part-time testers.

(3 marks)

Driver

**package** exercise2;

**import** javax.swing.JOptionPane;

**public** **class** Driver {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

String type;

**double** salary;

String name;

String user = "";

**while**(!user.equalsIgnoreCase("exit"))

{

name = JOptionPane.*showInputDialog*("Game Tester's Name");

type = JOptionPane.*showInputDialog*("Game Tester's job fulltime or parttime?");

**if**(type.equalsIgnoreCase("parttime"))

{

salary = Double.*parseDouble*(JOptionPane.*showInputDialog*("What is Hourly rate"));

GameTester temp = **new** PartTimeGameTester(name,**false**);

temp.setSalary(salary);

temp.displayInfo();

}

**else** **if**(type.equalsIgnoreCase("fulltime"))

{

salary = Double.*parseDouble*(JOptionPane.*showInputDialog*("What is Monthly salary"));

GameTester temp = **new** PartTimeGameTester(name,**true**);

temp.setSalary(salary);

temp.displayInfo();

}

user = JOptionPane.*showInputDialog*("Type \" exit\" if you want to exit or type anything");

}

}

}

GameTester

**package** exercise2;

**public** **abstract** **class** GameTester {

**protected** String gameTesterName;

**protected** **boolean** fullTime;

**protected** **double** salary;

**public** GameTester(String name,**boolean** isFulltime)

{

**this**.gameTesterName = name;

**this**.fullTime = isFulltime;

}

**public** **abstract** **void** setSalary(**double** rate);

**public** **abstract** **void** displayInfo();

}

FullTimeGameTester

**package** exercise2;

**public** **class** FullTimeGameTester **extends** GameTester {

**public** FullTimeGameTester(String name, **boolean** isFulltime) {

**super**(name, isFulltime);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

// **TODO** Auto-generated method stub

}

@Override

**public** **void** displayInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Name: "+**this**.gameTesterName+"\nGame Tester: "+(**this**.fullTime ? "Full timer" : "Part Timer")+"\n"+"Monthly salary: "+**this**.salary);

}

}

PartTimeGameTester

**package** exercise2;

**public** **class** PartTimeGameTester **extends** GameTester {

**public** PartTimeGameTester(String name, **boolean** isFulltime) {

**super**(name, isFulltime);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** setSalary(**double** rate) {

// **TODO** Auto-generated method stub

**this**.salary = rate;

}

@Override

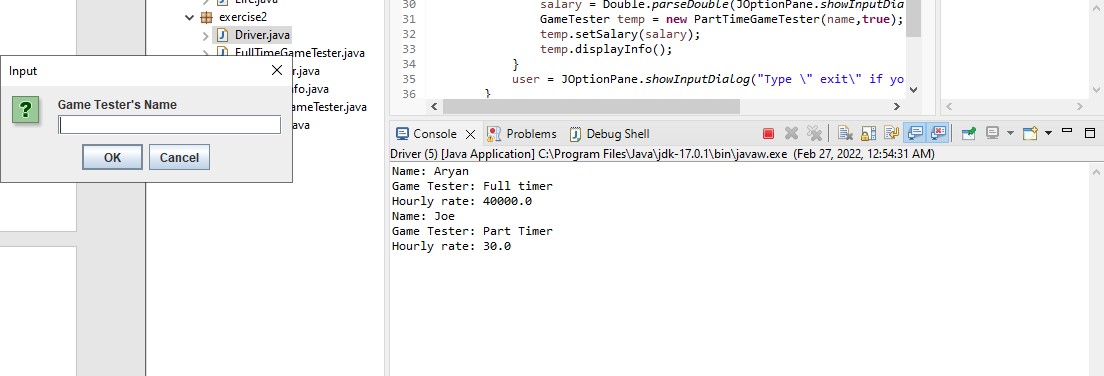
**public** **void** displayInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Name: "+**this**.gameTesterName+"\nGame Tester: "+(**this**.fullTime ? "Full timer" : "Part Timer")+"\n"+"Hourly rate: "+**this**.salary);

}

}



**Exercise #3:**

CityToronto bank provides mortgages for individuals and businesses up to $300,000. Write a Java application that keeps track of mortgages and computes the total amount owed at any time (mortgage amount + interest).

Design the following classes to implement your application:

**Mortgage** – an abstract class that implements the *MortgageConstants* interface. A Mortgage includes a mortgage number, customer name, amount of mortgage, interest rate, and term.

Don’t allow mortgage amounts over $300,000. Force any mortgage term that is not defined in the *MortgageConstants* interface to a short-term, one year loan. Create a *getMortgageInfo* method to display all the mortgage data.

**MortgageConstants** – includes constant values for *short-term* (one year), *medium-term* (three years) and *long-term* (5 years) mortgages. It also contains constants for bank name and the maximum mortgage amount.

**BusinessMortgage** – extends Mortgage. Its constructor sets the interest rate to 1% over the current prime rate.

**PersonalMortgage** - extends Mortgage. Its constructor sets the interest rate to 2% over the current prime rate.

**ProcessMortgage** – a main class that create an array of 3 mortgages. Prompt the user for the current interest rate. Then in a loop prompts the user for a mortgage type and all relevant information for that mortgage. Store the created Mortgage objects in the array. When data entry is complete, display all mortgages.

(4 marks)

Driver

package exercise3;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JOptionPane;

public class Driver {

public static void main(String[] args) {

// TODO Auto-generated method stub

List<Mortgage> mg = new ArrayList<>();

int i = 0;

while(i<3)

{

String cname = JOptionPane.showInputDialog("What is customer name?");

String type = JOptionPane.showInputDialog("Mortgage Type 'buisness' or 'personal'");

double MortgageAmount =Double.parseDouble(JOptionPane.showInputDialog("Total Mortgage Amount"));

int term = Integer.parseInt(JOptionPane.showInputDialog("how many years 1,3,5"));

while(MortgageAmount > MortgageConstants.maxMortgageAmount)

{

System.out.println("Morgage is not allowed mor than "+MortgageConstants.maxMortgageAmount);

MortgageAmount =Double.parseDouble(JOptionPane.showInputDialog("Total Mortgage Amount"));

}

if(type.equalsIgnoreCase("buisness"))

{

mg.add(new BuisnessMortgage(cname, MortgageAmount, 12, term ));

}

else if(type.equalsIgnoreCase("personal"))

{

mg.add(new PersonalMortgage(cname, MortgageAmount, 12, term ));

}

i++;

}

for(Mortgage m: mg)

{

m.getMortgageInfo();

}

}

}

MortgageConstants

**package** exercise3;

**public** **interface** MortgageConstants {

**final** **int** ***shTMortgage*** = 1;

**final** **int** ***mTMortgage*** = 3;

**final** **int** ***lTMortgage*** = 5;

**static** **final** String ***bankName*** = "RBC";

**static** **final** **int** ***maxMortgageAmount*** = 300000;

}

Mortgage

**package** exercise3;

**public** **abstract** **class** Mortgage **implements** MortgageConstants {

**public** **int** mNumber;

**public** String cName;

**public** **double** amountOfMortgage;

**protected** **float** rate;

**public** **int** term = ***shTMortgage***;

**private** **static** **int** *id* = 1;

**public** Mortgage(String cName, **double** amountOfMortgage, **float** rate, Integer...term)

{

**this**.amountOfMortgage = amountOfMortgage;

**this**.cName = cName;

**this**.rate = rate;

**if**(term.length > 0)

{

**this**.term = term[0];

}

**this**.mNumber = Mortgage.*id*++;

}

**public** **abstract** **void** getMortgageInfo();

}

BusinessMortgage

**package** exercise3;

**public** **class** BuisnessMortgage **extends** Mortgage {

**public** BuisnessMortgage(String cName, **double** amountOfMortgage, **float** rate, Integer... term) {

**super**(cName, amountOfMortgage, rate+1, term);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** getMortgageInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Type of Mortgage:- Business"+"\nCustomer Name:- "+ **this**.cName+"\nMortgage Number:- "+**this**.mNumber+"\nAmount of Mortgage:- "+ **this**.amountOfMortgage +"\n Interest rate:- "+**this**.rate+"\nTerm:- "+**this**.term+"\n");

}

}

PersonalMortgage

**package** exercise3;

**public** **class** PersonalMortgage **extends** Mortgage {

**public** PersonalMortgage(String cName, **double** amountOfMortgage, **float** rate, Integer... term) {

**super**(cName, amountOfMortgage, rate+2, term);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** getMortgageInfo() {

// **TODO** Auto-generated method stub

System.***out***.println("Type of Mortgage:- Personal"+"\nCustomer Name:- "+ **this**.cName+"\nMortgage Number:- "+**this**.mNumber+"\nAmount of Mortgage:- "+ **this**.amountOfMortgage +"\n Interest rate:- "+**this**.rate+"\nTerm:- "+**this**.term+"\n");

}

}

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of classes (instance variable declarations, constructors, getter and setter methods, etc.) | 40% |
| Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results) | 40% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 15% |
| **Total** | 100% |