**Department of Computer Science and Engineering**

|  |  |
| --- | --- |
| **Course Code: CSE111** | **Credits: 1.5** |
| **Course Name: Programming Language II** | **Semester: Sum’18** |

**Lab 09  
Introduction to Abstract class and Interfaces**

1. **Topic Overview:**

The students will solve problems to familiarize themselves with Abstract class and Interfaces. There are in total of 7 problems in this lab.

1. **Lesson Fit:**

Lab8 (Inheritance) is the pre-requisite to this lab.

1. **Learning Outcome:**

After this lab, the students will be able to:

* 1. Design abstract classes and interfaces
  2. Solve reverse engineering problems related to abstract and interfaces.

1. **Anticipated Challenges and Possible Solutions**
   1. Task 5: Students will make mistakes while designing abstract methods inside the abstract class.

**Solutions:**

**Abstract method structure:** abstract *type name (param-list);*

* 1. Task 6: Students will make mistakes while designing the interface.

**Solutions:**The instructor will discuss the structure of the interface.

1. **Acceptance and Evaluation**

Students will show their progress as they complete each problem. They will be marked according to their class performance. Their maybe students who might not be able to finish all 13 tasks, they will submit them later and give a viva to get their performance mark.

1. **Activity Detail**
   1. **Hour: 1  
      Discussion:**Explain how to write the Abstract classes, what are the benefits of it and why it needs to be extended.  **Problem Task:**
      1. Task 1 to 4
   2. **Hour: 2**

**Discussion:**

Check task 1 to 4 while the students continue with the rest.

Explain how to write the interface classes, what are the benefits of it, how Interfaces can be extended too. Also explain why Interfaces are about polymorphism, whereas, inheriting methods is about code reuse.

**Problem Task:**

* + 1. Task 5 to 6
  1. **Hour: 3**

**Discussion:**

Check task 5 to 6 while the students continue with the rest.

**Problem Task:**

* + 1. Task 7

1. **Home tasks**
   1. Unfinished tasks

**Lab 1 Activity List**

**Task 1**

Write the **CheckingAccount** class so that the following code generates the output below:

**public abstract class Account{**

**private static long accountNumber = 100;**

**private String name;**

**private long accNumber;**

**private double balance;**

**public Account(String name){**

**this.name = name;**

**accNumber = accountNumber++;**

**balance = 0.0;**

**}**

**protected String getName(){**

**return name;**

**}**

**protected double getBalance(){**

**return balance;**

**}**

**protected void setBalance(double bal){**

**if (bal > 0){**

**balance = bal;**

**}**

**}**

**protected long getAccountNumber(){**

**return accNumber;**

**}**

**public abstract void printAccountInfo();**

**}**

**public class AccountTester{**

**public static void printInfo(Account a){**

**a.printAccountInfo();**

**}**

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

**CheckingAccount c1 = new CheckingAccount("Nadeem", 400);**

**CheckingAccount c2 = new CheckingAccount("Susan", 600);**

**CheckingAccount c3 = new CheckingAccount("Raqib", 1000);**

**printInfo(c1);**

**printInfo(c2);**

**printInfo(c3);**

**System.out.println(c1.deposit(-100.00));**

**System.out.println(c1.deposit(0.00));**

**System.out.println(c1.deposit(200.00));**

**System.out.println(c1.withdraw(1000.00));**

**System.out.println(c1.withdraw(0.00));**

**System.out.println(c1.withdraw(300.00));**

**printInfo(c1);**

**}**

**}**

**Output of AccountTester:**

**=======================================**

**Name : Nadeem**

**Account Number : 100**

**Balance : 400.0**

**Your balance is less than the minimum amount**

**=======================================**

**=======================================**

**Name : Susan**

**Account Number : 101**

**Balance : 600.0**

**=======================================**

**=======================================**

**Name : Raqib**

**Account Number : 102**

**Balance : 1000.0**

**=======================================**

**Please enter amount greater than Zero**

**Please enter amount greater than Zero**

**Deposit Successful**

**You do not have enough funds**

**Please enter amount greater than Zero**

**Withdraw Successful**

**=======================================**

**Name : Nadeem**

**Account Number : 100**

**Balance : 300.0**

**Your balance is less than the minimum amount**

**=======================================**

**Task 2**

Write the BBAStudent class so that the following code generates the output below

|  |
| --- |
| **public abstract class Student{** |
| **private String name = "Just a Student";** |
| **protected String department=”nothing”;** |
| **public abstract void setDepartment();** |
| **protected final String getName(){** |
| **return name;** |
| **}** |
| **protected final void setName(String name){** |
| **this.name=name;** |
| **}** |
| **public String toString(){** |
| **return “Name: ”+name+” Department: ”+department;** |
| **}** |
| **}** |
| **public class TestStudent{** |
| **public static void printName(Student s){** |
|  |
| **System.out.println(s);** |
| **}** |
| **public static void main(String [] args){** |
| **printName(new BBAStudent ());** |
| **printName(new BBAStudent ("Barack Obama"));** |
| **printName(new BBAStudent("John McCain"));** |
|  |
| **}** |
| **}** |

**Name : Default BBA Student Department :BBA**

**Name : Barack Obama Department :BBA**

**Name : John McCain Department :BBA**

**Task 3**

Write the **Cat** and **Dog** class so that the following code generates the output below

|  |
| --- |
| public abstract class Animal { |
| protected String sound = "Animal Sound"; |
| public Animal(){ |
| } |
| Animal(String \_sound){ |
| this.sound = \_sound; |
| } |
| public abstract String makeSound(); |
| } |
|  |
| public class AnimaDriver{ |
| public static void printSound(Animal a){ |
| System.out.println(a.makeSound()); |
| } |
| public static void main(String [] args){ |
| Dog d1 = new Dog("bark"); |
| Cat c1 = new Cat("meow"); |
| printSound(c1); |
| printSound(d1); |
| } |
| } |

**meow**

**bark**

**Task 4**

Given the following classes, write the code for the **Dog** class so that the following output is created [Dog extends animal].

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Rover is crying.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**abstract public class Animal {**

**//Name of the Animal**

**private String name;**

**//Constructor**

**Animal(String \_name){**

**this.name = \_name;**

**}**

**//Return name**

**protected String getName(){**

**return name;**

**}**

**}**

**//Driver class**

**public class AnimalTorture {**

**private void hugAnimal(SoundSource a){**

**a.makeSound();**

**}**

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

**Dog d = new Dog("Rover");**

**AnimalTorture at = new AnimalTorture();**

**at.hugAnimal(d);**

**}**

**}**

**//Interface for making sound.**

**public interface SoundSource {**

**public void makeSound();**

**}**

**Task 5**

Write an abstract class ‘**Instrument’** which will have abstract method ‘**play’**, ‘**adjust’** & concrete method ‘**compose’**.   
  
Use the abstract class ‘Instrument’ to create class ‘**Guitar’**, ‘**Keyboard’** & ‘**Violin’**.

Create instance (object) of every class invoking (calling) every method.   
  
The method will print any message with ‘Instrument name’ and ‘Purpose’. The method ‘play’ for ‘Violin’ class will print “In the playing method of Violin”

https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

**Task 6**

Write a java application as follows:Create a Student Interface ‘**StInterface’** with the methods ‘**setName’**, ‘**setID’**, ‘**getName’** **and ‘getID’**.

Create the class ‘**Student’** with ‘name’, ‘id’ and ‘address field’ implementing the ‘**StInterface’** to manipulate the Student information using the necessary methods.

Create an array of objects of **Student**. Then input the number of students to allocate student array dynamically and take Student information. Now print the student list alphabetically.

**Task 7**

**Refactoring:**

Given the three existing classes: Point, Circle, and Rectangle, establish a properly designed inheritance hierarchy with an abstract class named **Shape**. On the following page, not inherited version of the classes are already given. You do not have to rewrite the Point class.

*Implement the hierarchy here: Shape, Circle, and Rectangle (no need to modify the Point class)*

|  |
| --- |
| **Required Output** |
| 0) 3.141592653589793 at 5, 10 |
| 1) 18.6875 at 40, 10 |
| 2) 12.566370614359172 at 15, 30 |

public class ShapesTwo{

public static void main(String[] args){

int n = 3;

Shape[] shapes = new Shape[n];

shapes[0] = new Circle(5, 10, 2.0);

shapes[1] = new Rectangle(40, 10, 3.25, 5.75);

shapes[2] = new Circle(15, 30, 4.0);

for (int i = 0; i < n; i++){

System.out.println(i + ") " + shapes[i].getArea() + " at " + shapes[i].getX() + ", " +

shapes[i].getY());

}

}

}

|  |
| --- |
| **public class Point {**  **private int my\_xPos;**  **private int my\_yPos;**    **public Point(int x, int y) {**  **my\_xPos = x;**  **my\_yPos = y;**  **}**    **public int getX() {**  **return my\_xPos;**  **}**  **public int getY() {**  **return my\_yPos;**  **}**  **}**  **public class Rectangle{**  **private Point my\_upperLeft;**  **private double my\_width;**  **private double my\_height;**    **public Rectangle(int x, int y, double height, double width){**  **my\_upperLeft = new Point(x, y);**  **my\_width = width;**  **my\_height = height;**  **}**    **public int getX() {**  **return my\_upperLeft.getX();**  **}**    **public int getY() {**  **return my\_upperLeft.getY();**  **}**    **public double getArea() {**  **return my\_width \* my\_height;**  **}**  **}**  **public class Circle{**  **private Point my\_upperLeftCorner;**  **private double my\_radius;**    **public Circle(int x, int y, double diameter) {**  **my\_upperLeftCorner= new Point(x,y);**  **my\_radius = diameter / 2;**  **}**  **public int getX() {**  **return my\_upperLeftCorner.getX();**  **}**  **public int getY() {**  **return my\_upperLeftCorner.getY();**  **}**  **public double getArea(){**  **return Math.PI\*Math.pow(my\_radius,2);**  **}**  **}** |