OOP using JAVA – Practical 04

ID – 28371

Exercise 01:

Create a class called “Employee” which has 3 private variables (empID, empName, empDesignation) and create getters and setters for each field. Please note that this has no main method since this is just a blueprint not a application. Now crate a test class to invoke the Employee class. Create two objects for Mr.Bogdan and Ms.Bird and set required values using setters and print them back on the console using getters.

**package com.company.employeetext;**

**public class Employee {**

**private String empName,empdesignation;**

**private int empID;**

**public void setEmpName(String EmployeeName)(**

**empName = EmployeeName;**

**}**

**public void setEmpDesignation(String EmployeeDesignation){**

**empdesignation = EmployeeDesignation;**

**}**

**public void setEmployeeId(int EmpId){**

**empID = EmpId;**

**}**

**public String getEmpName(){**

**return empName;**

**}**

**public String getEmpDestination(){**

**return empdesignation;**

**}**

**public int getEmployeeId(){**

**return empID;**

**}**

**}**

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

**package com.company.employeetext;**

**public class EmployeeText {**

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

**Employee e1 = new Employee();**

**e1.setEmpName("Mr.Bogdan");**

**e1.setEmpDesignation("President");**

**e1.setEmployeeId(0001);**

**System.out.println("Employee Name is " + e1.getEmpName());**

**System.out.println("Employee Designation is " + e1.getEmpDestination());**

**System.out.println("Employee ID is " + e1.getEmployeeId());**

**Employee e2 = new Employee();**

**e2.setEmpName("Ms.bird");**

**e2.setEmpDesignation("Manager");**

**e2.setEmployeeId(0002);**

**System.out.println("Employee Name is " + e2.getEmpName());**

**System.out.println("Employee Designation is " + e2.getEmpDestination());**

**System.out.println("Employee ID is " + e2.getEmployeeId());**

**}**

**}**

Exercise 02:

Develop the following class execute and discuss the answer: Please note that each class stored in separate files. Write down the answer.

class SuperB {

int x;

void setIt (int n) { x=n;}

void increase () { x=x+1;}

void triple () {x=x\*3;};

int returnIt () {return x;}

}

class SubC extends SuperB {

void triple () {x=x+3;} // override existing method

void quadruple () {x=x\*4;} // new method

}

public class TestInheritance {

public static void main(String[] args) {

SuperB b = new SuperB();

b.setIt(2);

b.increase();

b.triple();

System.out.println( b.returnIt() );

SubC c = new SubC();

c.setIt(2);

c.increase();

c.triple();

System.out.println( c.returnIt() ); }

}

**Result**

**1.We create an instance of SuperB named b.**

**2.We call setIt(2) on b, setting x to 2.**

**3.We call increase() on b, incrementing x by 1 (x = 3).**

**4.We call triple() on b, multiplying x by 3 (x = 9)**

**5.We print the value returned by returnIt() on b, which is 9.**

**6.We create an instance of Subc named c.**

**7.We call setIt(2) on c, setting x to 2.**

**8.We call increase() on c, incrementing x by 1 (x = 3).**

**9.We call triple() on c, adding 3 to x (x = 6).**

**10. We print the value returned by returnIt() on c, which is 6.**

**Therefore, the output of the main method will be:**

**9 and 6**

**This demonstrates the concept of inheritance, where SubC inherits the methods from SuperB and can override them to provide different functionality. In this case, SubC overrides the triple() method to add 3 to x instead of multiplying it by 3.**

Bottom of Form

Exercise 03:

Recall the following scenario discussed during the class. Develop a code base to represent the scenario. Add a test class to invoke Lecturer and Student class by creating at least one object from each.

Note: All the common attributes and behavior stored in the super class and only the specific fields and behavior stored in subclasses.

|  |
| --- |
| Student |
| * name |
| * id |
| * course |
| + setName()/getName() |
| + setID()/getID() |
| + setCourse()/getCourse() |

|  |
| --- |
| Lecturer |
| * name |
| * id |
| * programme |
| + setName()/getName() |
| + setID()/getID() |
| + setProg()/getProg() |

|  |
| --- |
| Person |
| Identify field and attributes to be stored in this class |

**package com.company.textperson;**

**public class Person {**

**private String name;**

**private int id;**

**public Person(String name,int id){**

**this.name = name;**

**this.id = id;**

**}**

**public void setName(String name){**

**this.name = name;**

**}**

**public String getName(){**

**return name;**

**}**

**public void setId(int id){**

**this.id = id;**

**}**

**public int getId(){**

**return id;**

**}**

**}**

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

**package com.company.textperson;**

**public class Student extends Person {**

**private String course;**

**public Student(String name,int id,String course){**

**super(name,id);**

**this.course = course;**

**}**

**public void setCourse(String course){**

**this.course = course;**

**}**

**public String getCourse(){**

**return course;**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**package com.company.textperson;**

**public class Lecturer extends Person{**

**private String programme;**

**public Lecturer(String name,int id,String programme){**

**super(name,id);**

**this.programme = programme;**

**}**

**public void setProgramme(String programme){**

**this.programme = programme;**

**}**

**public String getProgramme(){**

**return programme;**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

package com.company.textperson;

public class TextPerson {

public static void main(String[] args) {

Student S1 = new Student("Rasil Laksika",1234,"Software Engineering");

System.out.println("Student Name is " + S1.getName());

System.out.println("Student ID is " + S1.getId());

System.out.println("Student Course is " + S1.getCourse());

System.out.println("");

S1.setName("Miuru");

S1.setId(2222);

S1.setCourse("Computer Science");

System.out.println("Student Name is " + S1.getName());

System.out.println("Student ID is " + S1.getId());

System.out.println("Student Course is " + S1.getCourse());

System.out.println("");

Lecturer L1 = new Lecturer("Gayan Perera",1111,"C language");

System.out.println("Lecturer Name is " + L1.getName());

System.out.println("Lecturer ID is " + L1.getId());

System.out.println("Lecturer Programme is " + L1.getProgramme());

System.out.println("");

L1.setName("Ravindu Peris");

L1.setId(4444);

L1.setProgramme("Java with OOP");

System.out.println("Lecturer Name is " + L1.getName());

System.out.println("Lecturer ID is " + L1.getId());

System.out.println("Lecturer Programme is " + L1.getProgramme());

}

}

Exercise 04

Develop the following class execute and discuss the answer: Please note that each public class stored in separate files. Write down the answer.

public class Animal{}

public class Mammal extends Animal{}

public class Reptile extends Animal{}

**public class Animal {**

**// Class implementation**

**}**

**public class Mammal extends Animal {**

**// Class implementation**

**}**

**public class Reptile extends Animal {**

**// Class implementation**

**}**

**The “Animal” class serves as the base class or superclass for both “Mammal” and “Reptile” classes. It provides a foundation for common characteristics and behaviors shared by animals in general.**

**The “Mammal” class extends the “Animal” class, inheriting its properties and methods. This means that a “Mammal” object is also an “Animal” object, but with additional attributes and behaviors specific to mammals.**

**Similarly, the “Reptile” class extends the “Animal” class, inheriting its characteristics and functionalities. A “Reptile” object is also an “Animal” object, but it may have additional features and behaviors particular to reptiles.**

**By utilizing inheritance, we can create a hierarchy of classes that represent different types of animals. This approach promotes code reusability, as common features are encapsulated in the base class and inherited by the derived classes**.