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.

public class Employee {  
 private int empID;  
 private String empName;  
 private String empDesignation;  
  
 //setters  
 public void setEmpID(int empID) {  
 this.empID = empID;  
 }  
 public void setEmpName(String empName) {  
 this.empName = empName;  
 }  
 public void setEmpDesignation(String empDesignation) {  
 this.empDesignation = empDesignation;  
 }  
  
 //getters  
 public int getEmpID() {  
 return empID;  
 }  
 public String getEmpName() {  
 return empName;  
 }  
 public String getEmpDesignation() {  
 return empDesignation;  
 }  
}

public class Test extends Employee{

public static void main(String[] args){  
 var mr\_Borgan = new Employee();  
 var ms\_Bride = new Employee();  
  
 mr\_Borgan.setEmpID(001);  
 mr\_Borgan.setEmpName("Mr.Borgan");  
 mr\_Borgan.setEmpDesignation("Senior software engineer");  
  
 ms\_Bride.setEmpID(002);  
 ms\_Bride.setEmpName("Ms.Bride");  
 ms\_Bride.setEmpDesignation("Project manager");  
  
 System.*out*.println("Employee ID: "+mr\_Borgan.getEmpID());  
 System.*out*.println("Employee name: "+mr\_Borgan.getEmpName());  
 System.*out*.println("Employee Designation: "+mr\_Borgan.getEmpDesignation());

System.*out*.println(" ");  
  
 System.*out*.println("Employee ID: "+ms\_Bride.getEmpID());  
 System.*out*.println("Employee name: "+ms\_Bride.getEmpName());  
 System.*out*.println("Employee Designation: "+ms\_Bride.getEmpDesignation());  
  
 }  
}

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() ); }

}

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 atleast one object from each.

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

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() |

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 Dog extends Mammal{

public static void main(String args[]){

Animal a = new Animal();

Mammal m = new Mammal();

Dog d = new Dog();

System.out.println(m instanceof Animal);

System.out.println(d instanceof Mammal);

System.out.println(d instanceof Animal);

}

}

//output

true

true

true