Java Lab Exercise #01	**Class Definition and set & get Methods (Solution)**	**Evaluation:**
Student Name & Number		
Instructor:		**Date:**

Q1. What is the output of the following program?

Program

```
class Student
{
   private String StudentName;
   private double GPA;

   public void setStudentName (String StudentName)
   {
      this.StudentName = StudentName;
   }
   public void setGPA(double GPA)
   {
      this.GPA = GPA;
   }
   public String getStudentName ()
   {
      return StudentName;
   }
   public double getGPA()
   {
      return GPA;
   }
   public void print()
   {
      System.out.println("Student Name: " + StudentName +
"\n" + "GPA: " +GPA);
   }
}

public class TestStudent
{
   public static void main( String[] args )
   {
       Student s1 = new Student();
       Student s2 = new Student();

       s1.setStudentName("Huda");
       s1.setGPA(2.50);

       s2.setStudentName(s1. getStudentName());
       s2.setGPA(4.50);

        s1.print();
        s2.print();




   }
}
```

Output

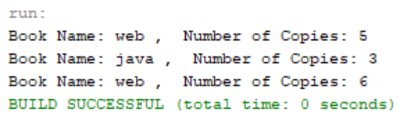
```
Student Name: Huda
GPA: 2.5
Student Name: Huda
GPA: 4.5
```

Q2. Writing program

Write a Java program that define class **Book** with two instance variables **BookName** and **Copies** of String and integer types. Define:

1. **set** and **get** methods for writing and reading values of BookName and Copies variables.
2. **incrementCopies** method for increment the number of copies by 1.
3. **Display** method to print the information for each book
4. Define main method to test class Book.

Sample output:



```
run:
Book Name: web ,  Number of Copies: 5
Book Name: java ,  Number of Copies: 3
Book Name: web ,  Number of Copies: 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
class  Book
{
   private  String  BookName;
   private  int  Copies;



   public  void  setBookName  (String  n)
   {
     BookName  =  n;
   }
   public  void  setCopies(int  c)
   {
     Copies  =  c;
   }
   public  String  getBookName  ()
   {
     return  BookName;
   }
   public  double  getCopies  ()
   {
     return  Copies;
   }

public  void  incrementCopies()
{
Copies++;
}
   public  void  display()
   {
     System.out.println("Book  Name:  "  +  BookName  +  "  ,   Number  of  Copies:  "
+  Copies);
   }
}
```

```
public  class  TestBook
{

public  static  void  main(  String[]  args  )
   {
        Book  b1  =  new  Book();
        Book  b2  =  new  Book();

        b1.setBookName("web");
         b1.setCopies(5);
         b1.display();

          b2.setBookName("java");
         b2.setCopies(3);
         b2.display();
         b1.  incrementCopies();

b1.display();
   }
}
```

Java Lab Exercise #01	**Class Definition and set & get Methods (Solution)**	**Evaluation:**
Student Name & Number		
Instructor:		**Date:**

Q1. What is the output of the following program?

Program

```
class  Book
{
   private  String  bookName;
   private  double  bookPrice;

   public  void  setBookName(String  bookName)
   {
     this.bookName  =  bookName;
   }
   public  void  setBookPrice(double  bookPrice)
   {
     this.bookPrice  =  bookPrice;
   }
   public  void  setBook(String  bookName,  double  bookPrice)
   {
     this.bookName    =  bookName;
     this.bookPrice  =  bookPrice;
   }
   public  String  getName()
   {
     return  bookName;
   }
   public  double  getPrice()
   {
     return  bookPrice;
   }
   public  void  show()
   {
     System.out.println("Book  Name:  "  +  bookName  +  "\n"  +
                              "Book  Price:  "  +  bookPrice);
   }
}
public  class  TestBook
{
   public  static  void  main(  String[]  args  )
   {
       Book  book1  =  new  Book();
       Book  book2  =  new  Book();
       book1.show();
       book2.show();
        System.out.println("-------------------------------");
       book1.setBookName  ("Java  How  to  Program");
       book1.setBookPrice(240.0);
       book2.setBookName  ("Introduction  to  Java");
       book2.setBookPrice(193.0);
       book1.show();
       book2.show();
        System.out.println("-------------------------------");
       book1.setBook(  book2.getName(),  book1.getPrice()+5  );
       book2.setBook(  book1.getName(),  book2.getPrice()-5  );
       book1.show();
       book2.show();    }
}
```

Output

```
Book Name: null
Book Price: 0.0
Book Name: null
Book Price: 0.0
------------------------------
Book Name: Java How to
Program
Book Price: 240.0
Book Name: Introduction to
Java
Book Price: 193.0
------------------------------
Book Name: Introduction to
Java
Book Price: 245.0
Book Name: Introduction to
Java
Book Price: 188.0
```

```
class Demo
{
   int x, y;

   public Demo()
   {
      x = y = 0;
   }
   public Demo(int a)
   {
      x = a;
   }
   public Demo(int a, int b)
   {
      this.x = a;
      this.y = b;
   }
   public void display()
   {
      System.out.println("x = " + x + " y = " + y);
   }
}

public class Client
{
   public static void main( String[] args )
   {
      Demo d1 = new Demo(); d1.display();
      Demo d2 = new Demo(2); d2.display();
      Demo d3 = new Demo(3, -5); d3.display();
   }
}
```

```
x = 0, y = 0
x = 2, y = 0
x = 3, y = -5
```

Q2. Writing program

Define a Java class **Rectangle** that has the following specifications:

private instance variables:

- **length**: of type int
- **width**: of type int

Methods:

- Define two overloaded constructor methods with **zero argument**, **two arguments** (receives length and width).
- ***SetLength(xxx)***: acts as the set method for the attribute **length**.
- ***setWidth(xxx)***: acts as the set method for the attribute **width**.
- ***getLength ()***: acts as the get method for the attribute **length**.
- ***getWidth ()***: acts as the get method for the attribute **width**.
- ***area():*** returns the area of the rectangle. **Area= length * width** .

Define a Java class **Test Rectangle** that contains a **main method** and test the functionality of the above class **Rectangle** as the following:

- Instantiate one object **rec1** of type **Rectangle** using **zero argument constructor.**
- Instantiate one object **rec2** of type **Rectangle** using **two argument constructor** and pass **5 as length and 4 as width** as a parameters**.**
- Change the **length** of **rec1** to 10 and **width** of **rec1** to 6.
- Print the information of the two objects ***rec1*** and ***rec2*** in the following format:

  length: 10, **width**: 6, **area**: 60.
  length: 5, **width**: 4, **area**: 20.

```
public class TestRectangle
{
   public static void main( String[] args )
   {
        public static void main(String[] args) {
      Rectangle rec1 = new Rectangle();
        rec1.setLength(10);
        rec1.setWidth(6);
        System.out.println("length = "+rec1.getLength()+", width =
"+rec1.getWidth()+ ", area = "+rec1.area());
        Rectangle rec2 = new Rectangle(5,4);
        System.out.println("length = "+rec2.getLength()+", width =
"+rec2.getWidth()+ ", area = "+rec2.area());
       }

}
class Rectangle
{
   private int length, width;
   public Rectangle(){}
   public Rectangle(int length, int width){
         this.length=length;
         this.width=width;
}

   public void setLength( int length )
   {
      this.length = length;
   }
   public void setWidth( int width )
   {
      this.width = width;
   }
   public int getLength()
   {
         return length;
   }
   public int getWidth()
   {
         return width;
   }
   public int area()
   {
      return length*width;
   }
}
```

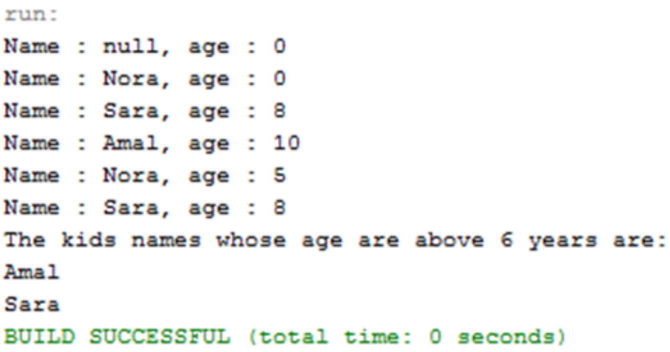
Java Lab Exercise #01	**Class Definition and set & get Methods (Solution)**	**Evaluation:**
Student Name & Number		
Instructor:		**Date:**

Q1. Writing program

Define java class **Kids** that has two private instance variables **name** and **age** (0.5 mark). Define three overloaded constructor methods with **zero argument** (receives no name and no age) (0.5 mark), **one argument** (receives only name) (0.5 mark) and **two arguments** (receives name and age) (0.5 mark). Additionally, define all **sets** and **gets** methods (2 marks). Define method **display** that print all the information of the kid(1 mark).

Define a Java class **TestKids** that contains a **main method** and test the functionality of the above class **Kids** as the following:

- Instantiate one object **k1** of type **Kids** using **zero argument constructor.** (0.5 mark)
- Instantiate one object **k2** of type **Kids** using **one argument constructor** and pass **"Nora"** as a parameter**.** (0.5 mark)
- Instantiate one object **k3** of type **Kids** using **two argument constructor** and pass **"Sara"** as the first parameter and **8** as second parameter**.** (0.5 mark)
- call **display** method for **k1, k2 and ks** objects. (0.5 mark)
- Change the **name** of object **k1** to **"Amal"** and the **age** to **10**. (1 mark)
- Change the **age** of the object **k2** to **5**. (0.5 mark)
- call **display** method for **k1, k2 and ks** objects. (0.5 mark)
- Print the names of all kids objects whose age are greater than 6 years old. (1 mark)



```
run:
Name : null, age : 0
Name : Nora, age : 0
Name : Sara, age : 8
Name : Amal, age : 10
Name : Nora, age : 5
Name : Sara, age : 8
The kids names whose age are above 6 years are:
Amal
Sara
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
package Lab2;
public class Lab2 {
    public static void main(String[] args) {
    Kids k1 = new Kids();
   Kids k2 = new Kids("Nora");
   Kids k3 = new Kids("Sara", 8);
   k1.display();
   k2.display();
   k3.display();

   k1.setName("Amal");
   k1.setAge(10);
   k2.setAge(5);
   k1.display();
   k2.display();
   k3.display();
   System.out.println("The kids names whose age are above 6 years are:");
   if (k1.getAge()>6)
       System.out.println (k1.getName());

   if (k2.getAge() > 6)
       System.out.println (k2.getName());

   if (k3.getAge() > 6)
       System.out.println (k3.getName());
  }
}
class Kids
{
 private String name;
 private int age;

 public Kids()
  {
  }
 public Kids(String n)
  {
   name = n;
   age = 0;
  }
 public Kids(String n, int a)
  {
   name = n;
   age = a;
  }
 public void setName(String n)
  {
     name=n;
  }
 public String getName()
  {
   return name;
  }
 public void setAge(int a)
  {
     age=a;
  }
 public int getAge()
  {
   return age;
  }
 public void display(){
     System.out.println("Name : "+getName()+", age : "+getAge());
  }
}
```

Java Lab Exercise #02	Constructor Method Overloading (Solution)	Evaluation:
Student Name & Number		
Instructor:		Date:

Q1. What is the output of the following program?

Program

```
class  House
{
      private  int  numberOfRooms  =  8;
      private  double  price  =  450000;

      public  House()
       {
            numberOfRooms  =  0;
       }
      public  House(int  a)
       {
             price  =  200000;
       }
      public  House(int  n,  double  p)
       {
            this.numberOfRooms  =  n;
       }
      public  void  display()
       {
             System.out.println("numberOfRooms  =  "  +
                              numberOfRooms  +  "  price  =  "  +  price);
       }
}

public  class  Client
{
      public  static  void  main(  String[]  args  )
       {
             House  h1  =  new  House();  h1.display();
             House  h2  =  new  House(4);  h2.display();
             new  House(6,  300000).display();
       }
}
```

Output

```
numberOfRooms = 0 price = 450000.0
numberOfRooms = 8 price = 200000.0
numberOfRooms = 6 price = 450000.0
```

Q2. Trace the following program

```
class Student {
    private double sum;
    private static int numStudents = 0;

    public Student(double sum) {
        numStudents++;
        this.sum += sum;
    }
    public static int getNumStudents () {
        return Student.numStudents;
    }
    public double getAverage() {
        return sum / numStudents;
    }
}

public class StudentDemo {
    public static void main(String[] args) {
        Student s1 = new Student(70);
        System.out.println("Average = " + s1.getAverage());
        new Student(80);
        Student s3 = new Student(90);
        System.out.println("Average = " + s3.getAverage());
    }
}
```

1. Write the output of the above program

```
Average = 70.0
Average = 30.0
```

2. Detect the logical error in the above program and correct it

   Error: `private double sum;`
   Correction: `private static double sum;`

3. What would be the output once the logical error is corrected.

```
Average = 70.0
Average = 80.0
```

4. The instance variable `numStudents` is accessed through the class name in `getNumStudents()`. Why?

   Because instance variable `numStudents` is static

5. Will the following statement produce an error if we place it in the main?

   `Student s = new Student();`

   If the answer is yes, why?

   Yes. Because `no-argument constructor` is not defined in the class.

Q3. Write a program

Define class **Continent** that has two instance variables **name** and **numberOfCountries**. Define three overloaded constructor methods: **no-argument constructor** (receives no name and no numberOfCountries), **one-argument constructor** (receives only name) and **two-argument constructor** (receives name and numberOfCountries). Additionally, define all **sets** and **gets** methods. In the **main** method, define three objects of class **Continent** and test the three constructors and print all information of the continent that has largest number of countries. The number of countries in each continent is shown in the following table.

	Continent	Number of countries
1	Asia	50
2	Africa	54
3	North America	23
4	South America	12
5	Antarctica	0
6	Europe	51
7	Autralia	14

```
class Continent
{
  private String name;
  private int numberOfCountries;

  public Continent()
  {
    name = "";
    numberOfCountries = 0;
  }
  public Continent(String name)
  {
    this.name = name;
    numberOfCountries = 0;
  }
  public Continent(String name, int numberOfCountries)
  {
    this.name = name;
    this.numberOfCountries = numberOfCountries;
  }
  public void setName(String name)
  {
    this.name = name;
  }
  public void setNumberOfCountries(int numberOfCountries)
  {
    this.numberOfCountries = numberOfCountries;
  }
  public String gettName()
  {
    return name;
  }
  public int getNumberOfCountries()
  {
    return numberOfCountries;
  }
  public void print() {
      System.out.println("Name: " + gettName() + ", Number of countries: " +
getNumberOfCountries());
  }
}

public class ContinentDemo {
    public static void main( String[] args ) {
        Continent c1 = new Continent();
        Continent c2 = new Continent("Asia");
        Continent c3 = new Continent("North America", 23);
        Continent c = null;
        int max = 0;

        c1.setName("Africa");
        c1.setNumberOfCountries(54);
        c2.setNumberOfCountries(50);

        System.out.println("The continent that has largest number of countries: ");

        if(c1.getNumberOfCountries()>max) {
            max = c1.getNumberOfCountries();
            c = c1;
        }
        if(c2.getNumberOfCountries()>max) {
            max = c2.getNumberOfCountries();
            c = c2;
        }
        if(c3.getNumberOfCountries()>max) {
            max = c3.getNumberOfCountries();
            c = c3;
        }
        c.print();
    }
}
```

Java Lab Exercise #03	Static Methods, Static Fields and Scope of Declarations (Solution)	Evaluation:
Student Name & Number		
Instructor:		Date:

Q1. What is the output of the following program?

Program

```java
public class EnemyAircrafts
{
  private static int enemyAircrafts;
  private static int speed = 100;

  EnemyAircrafts()
  {
    ++enemyAircrafts;
    speed += 10;
  }
  public static void hitAircraft()
  {
    --enemyAircrafts;
  }
  public static void speed()
  {
    System.out.println("Speed = " + speed);
  }
  public static void main(String args[])
  {
    int speed = 50;

    new EnemyAircrafts();
    EnemyAircrafts.hitAircraft();
    System.out.println("No. of enemy's aircrafts ="
                 + EnemyAircrafts.enemyAircrafts);

    EnemyAircrafts e = new EnemyAircrafts();
    new EnemyAircrafts();
    System.out.println("No. of enemy's aircrafts ="
                        + e.enemyAircrafts);

    speed(); //class
    ++speed;
    System.out.println("Speed = " + speed); //local
  }
}
```

Output

```
Number of enemy's aircrafts = 0
Number of enemy's aircrafts = 2
Speed = 130
Speed = 51
```

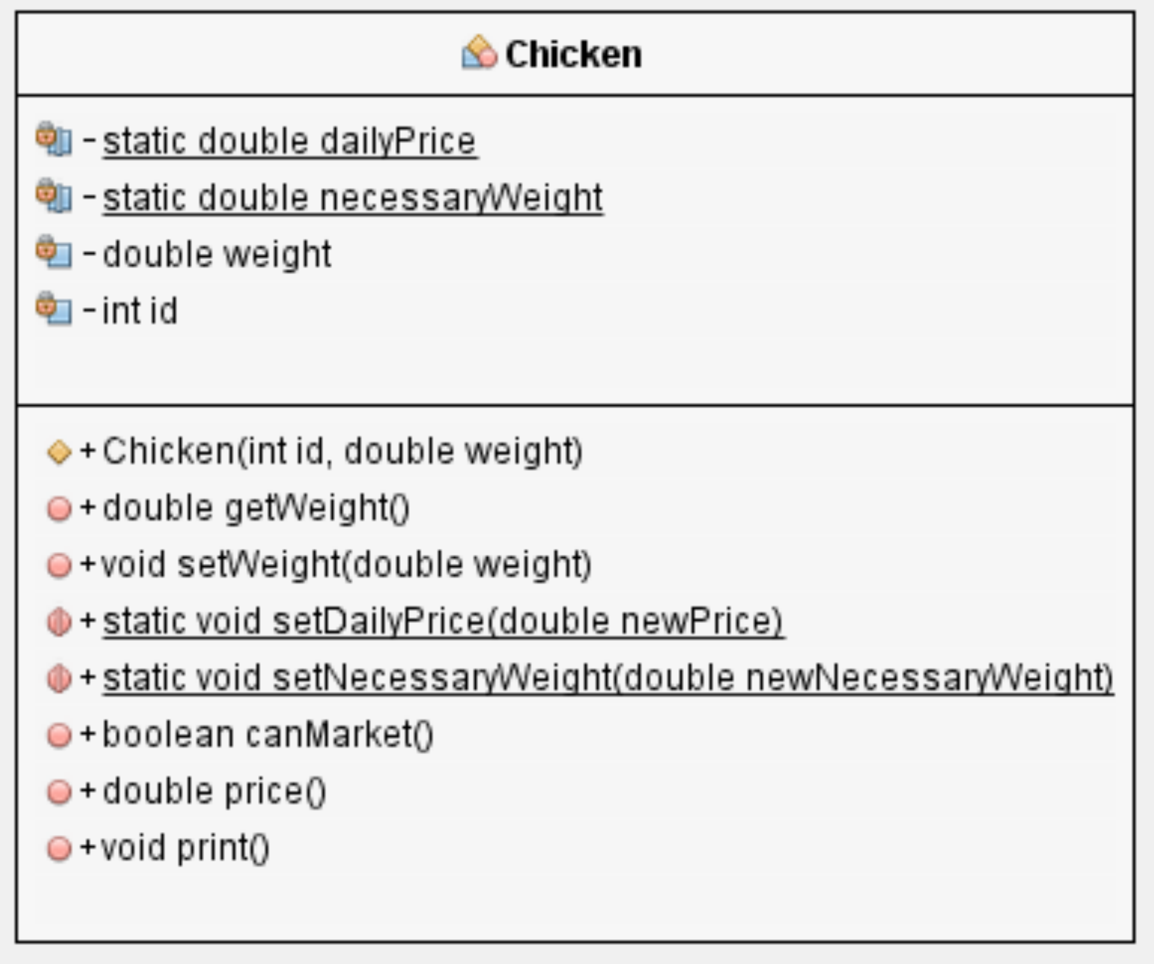
Q2. Writing program

Write a Java program that define class **Maximum** with three overloaded methods with the same name **Maximum( )**. The first method receives three integers and returns the value of the largest integer. The second is a static method receives three doubles and returns the largest double. The third method receives two characters and returns the largest character. Define main method to test class Maximum.

```java
class Maximum
{
    public int Maximum(int a, int b, int c)
    {
        int max = a;
        if (b > max)
            max = b;
        if (c > max)
            max = c;

        return max;
    }

    public static double Maximum(double a, double b, double c)
    {
        double max = a;
        if (b > max)
            max = b;
        if (c > max)
            max = c;

        return max;
    }
    public char Maximum(char a, char b)
    {
        if (a > b)
            return a;
        else
            return b;
    }
}

public class MaximumApplication
{
    public static void main(String[] args)
    {
        System.out.println (new Maximum().Maximum(1,5,2));
        System.out.println (Maximum.Maximum(1.0,2.0, 3.0)); // class
        System.out.println (new Maximum().Maximum('s','d'));
    }
}
```

Java Lab Exercise #03	Static Methods, Static Fields and Scope of Declarations (Solution)	Evaluation:
Student Name & Number		
Instructor:		Date:

A chikcken farmer receives from a supplier young chickens that he raises until they have the weight needed for marketing. A chicken is characterized by its weight and an identification number.
The price of a chicken varies according to the price of the day and the weight. Suppose that the price of the day is 5 riyals per kilo. So for a chicken with a weight 3.4 kilograms the price will be 17 riyals. The price of the day (dailyPrice) and the necessary weight to market a chicken (necessaryWeight) are the same for all chickens. By default **dailyPrice = 5** and **necessaryWeight=1.2**.
Write the Chicken class according to the class diagram below:



Chicken

-static double dailyPrice
-static double necessaryWeight
-double weight
-int id

+Chicken(int id, double weight)
+double getWeight()
+void setWeight(double weight)
+static void setDailyPrice(double newPrice)
+static void setNecessaryWeight(double newNecessaryWeight)
+boolean canMarket()
+double price()
+void print()

- Declare variables as mentioned in the above figure (0.5 mark)
- full parametarized constructor. (0.5 mark)
- **getWeight** and **setWeight** methods corresponds to getter and setter for the weight attribute. . (1 mark)
- **setDailyPrice :** set the new daily price for all chickens. . (1 mark)
- **setNecessaryWeight:** set the necessary weight to market a chicken. . (1 mark)
- **canMarket :** return true if the weight of a chicken is grater or equal to the necessary weight and false otherwise. . (1 mark)

- **Price :** return the price of chicken (weight * dailyPrice) if the chicken have the necessary weight and 0 otherwise. . **(**1 mark **)**
- **Print :** print the id and the weight of the chicken. If the chicken is ready for market print “ready for market” and give the price. Otherwise print “dont have the necessary weight”. **(**1 mark **)**

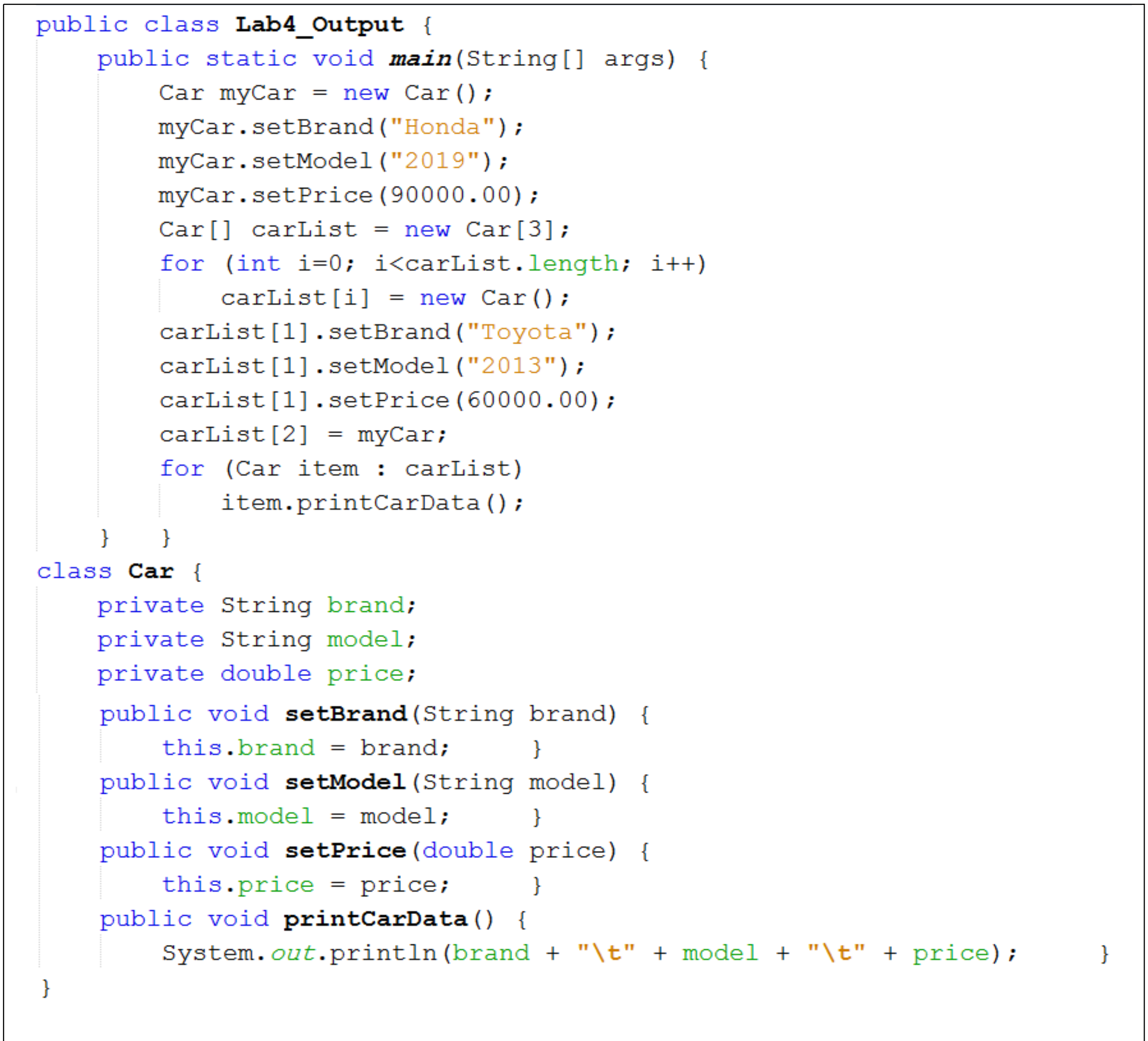
Define a Java class **TestChicken** that contains a **main method** and test the functionality of the above class **Chicken** as the following:

- Instantiate one object **ch1** of type **Chicken** and pass **1** as the first parameter and **1** as second parameter**.** **(**0.5 mark **)**
- call method **Print** for the object **ch1**. **(**0.25 mark **)**
- Instantiate one object **ch2** of type **Chicken** and pass **2** as the first parameter and **2** as second parameter**.** **(**0.5 mark **)**
- call method **Print** for the object **ch2**. **(**0.25 mark **)**
- Change the value of the static varible **dailyPrice** to **8**. (0.5 mark)
- call method **Print** for the object **ch2**. . **(**0.25 mark **)**
- Change the value of the static varible **necessaryWeight** to 3. **(**0.5 mark **)**
- call method **Print** for the object **ch2**. . **(**0.25 mark **)**

```
public class Chicken{
  private static double dailyPrice =5;
  private static double necessaryWeight=1.2;
  private double weight;
  private int id;
  public Chicken(int id, double weight){
    this.id=id;
    this.weight=weight;
  }
  public double getWeight(){
    return weight;
  }
  public void setWeight(double weight){
    this.weight=weight;
  }
  public static void setDailyPrice(double newPrice){
    dailyPrice=newPrice;
  }
  public static void setNecessaryWeight(double newNecessaryWeight){
    necessaryWeight= newNecessaryWeight;
  }
  public boolean canMarket(){
    return(weight>=necessaryWeight);
  }
  public double price(){
    if(this.canMarket())
    return(dailyPrice*weight);
    else
    return 0;
  }
  public void print(){
    System.out.println("Chiken id : "+id);
    System.out.println("Weight : "+weight);
    if(this.canMarket())
    System.out.println("ready for market. Price :"+this.price());
    else
    System.out.println("dont have the necessary weight");
  }
}

class Main {
  public static void main(String[] args) {
        Chicken ch1 = new Chicken(1,1);
        ch1.print();

        Chicken ch2 = new Chicken(2,2);
        ch2.print();

        Chicken.setDailyPrice(8);
        ch2.print();
        Chicken.setNecessaryWeight(3);
        ch2.print();  }
}
```

Java Lab Exercise #04	**Pass by Value, Pass by Reference and Arrays (Solution)**	**Evaluation:**
Student Name & Number		
Instructor:		**Date:**

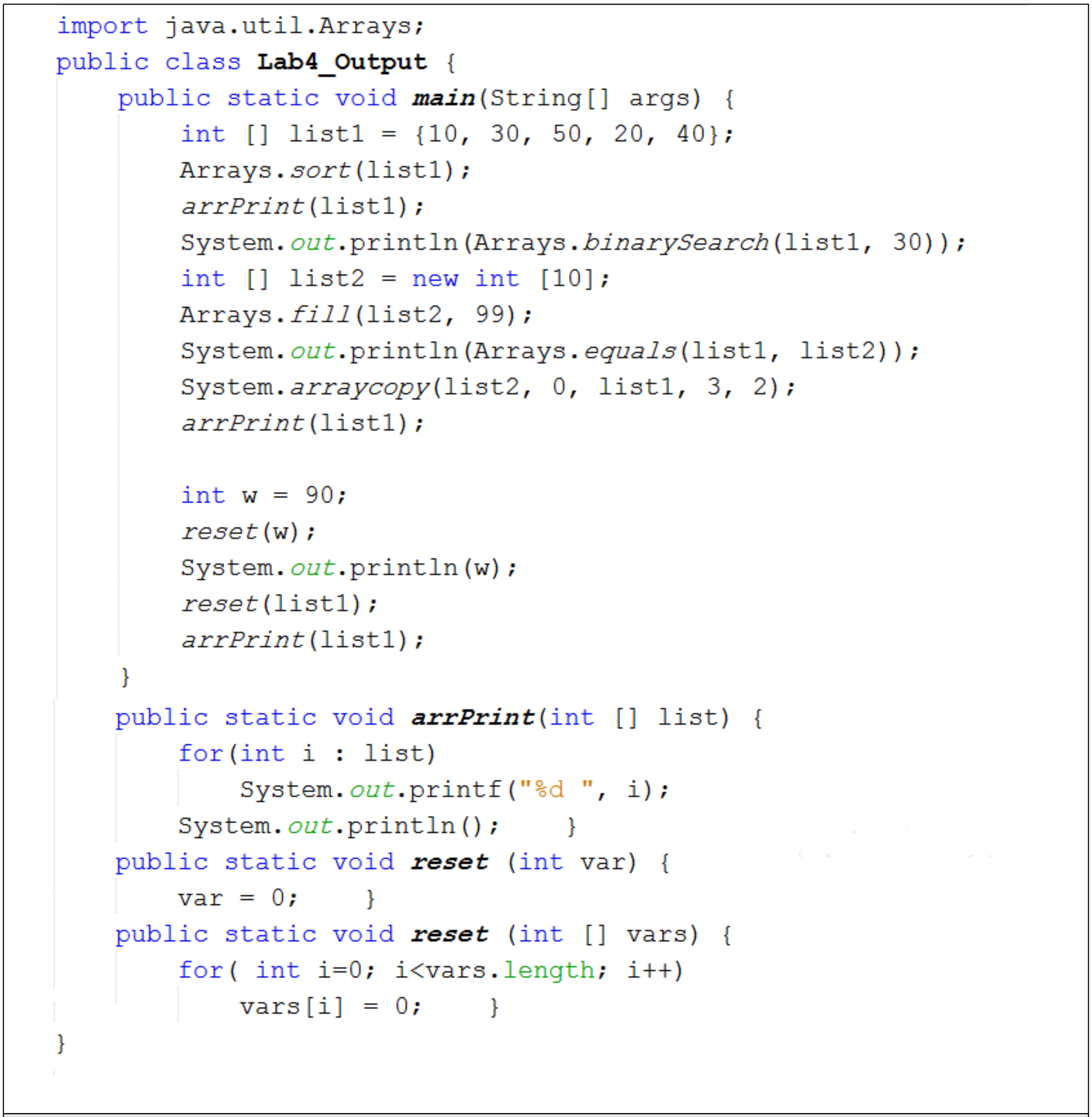
Q1. What is the output of the following program?



```java
public class Lab4_Output {
    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.setBrand("Honda");
        myCar.setModel("2019");
        myCar.setPrice(90000.00);
        Car[] carList = new Car[3];
        for (int i=0; i<carList.length; i++)
            carList[i] = new Car();
        carList[1].setBrand("Toyota");
        carList[1].setModel("2013");
        carList[1].setPrice(60000.00);
        carList[2] = myCar;
        for (Car item : carList)
            item.printCarData();
    }   }
class Car {
    private String brand;
    private String model;
    private double price;
    public void setBrand(String brand) {
        this.brand = brand;       }
    public void setModel(String model) {
        this.model = model;       }
    public void setPrice(double price) {
        this.price = price;       }
    public void printCarData() {
        System.out.println(brand + "\t" + model + "\t" + price);      }
}
```

Output

```
null  null  0.0
Toyota      2013  60000.0
Honda       2019  90000.0
```



```java
import java.util.Arrays;
public class Lab4_Output {
    public static void main(String[] args) {
        int [] list1 = {10, 30, 50, 20, 40};
        Arrays.sort(list1);
        arrPrint(list1);
        System.out.println(Arrays.binarySearch(list1, 30));
        int [] list2 = new int [10];
        Arrays.fill(list2, 99);
        System.out.println(Arrays.equals(list1, list2));
        System.arraycopy(list2, 0, list1, 3, 2);
        arrPrint(list1);

        int w = 90;
        reset(w);
        System.out.println(w);
        reset(list1);
        arrPrint(list1);
    }
    public static void arrPrint(int [] list) {
        for(int i : list)
            System.out.printf("%d ", i);
        System.out.println();     }
    public static void reset (int var) {
        var = 0;     }
    public static void reset (int [] vars) {
        for( int i=0; i<vars.length; i++)
            vars[i] = 0;     }
}
```

Output

```
10 20 30 40 50
2
false
10 20 30 99 99
90
0 0 0 0 0
```

```
import java.util.ArrayList;
import java.util.Arrays;
public class ArrayEx
{
    public static void main(String[] args) {
        ArrayList<String> carList = new ArrayList<String>();
        carList.add("Toyota");
        carList.add("Nissan");
        carList.add("BMW");
        carList.add(1,"Chevrolet");
        carList.remove("Nissan");
        carList.add("Ford");
        carList.add("Honda");
        System.out.println("There are "+carList.size()+" cars.");
        System.out.println("List of car names: " + carList.toString());
        System.out.print("List of car names (another way): ");
        for(String car : carList)
            System.out.printf("%s  ", car);
        System.out.println("\n");

        int[] carModel = {2014, 2011, 2008, 2017, 2013};
        int[] copyModel = new int [carModel.length];

        call_V_R(carModel[1], carModel);
        Arrays.sort(carModel);
        System.out.print("Models are: ");
        for(int i=0; i<carModel.length; i++)
            System.out.printf("%s  ", carModel[i]);
        System.out.println();
        System.arraycopy(carModel, 0, copyModel, 0, carModel.length);
        System.out.println("is carModel = copyModel? " +
                                    Arrays.equals(carModel, copyModel));
    }
    public static void call_V_R(int carElement, int[] carArray) {
        carElement = 2016;
        carArray[4]= 2015;
    }
}
```

Output

```
There are 5 cars.
List of car names: [Toyota, Chevrolet, BMW, Ford, Honda]
List of car names (another way): Toyota   Chevrolet   BMW   Ford   Honda

Models are: 2008   2011   2014   2015   2017
is carModel = copyModel? true
```

Q2. Writing program

Write a Java program to implement the following by using class **Arrays**:

- Read 10 amount of balances and keep them in array *balance*.
- Print the elements of array *balance*.
- Copy the content of array *balance* to another array *copyBalance*.
- Sort the array *balance* in ascending order and print the sorted array.
- Compare whether the two arrays *balance* and *copyBalance* are equal.
- Search for the balance 10000.0 in the array *balance* and print appropriate message.

Hint: use toString() for print, arraycopy() for copy, sort() for sort, equals() for comparison, and binarySearch() for search.

Sample of output:

```
Enter 10 amount of balances:
Balance 1: 2000
Balance 2: 1000
Balance 3: 35000
Balance 4: 10000
Balance 5: 7775
Balance 6: 900000
Balance 7: 81550
Balance 8: 660000
Balance 9: 400500
Balance 10: 5000
Balances: [2000.0, 1000.0, 35000.0, 10000.0, 7775.0, 900000.0, 81550.0, 660000.0, 400500.0, 5000.0]
Copy of balances is made ...
Balances sorted in ascending order:
[1000.0, 2000.0, 5000.0, 7775.0, 10000.0, 35000.0, 81550.0, 400500.0, 660000.0, 900000.0]
is Balance = copy of Balance? false
10000 is found at position 5
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class Balances
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);

        double[] balance = new double[10];
        double[] copyBalance = new double[10];

        System.out.println("Enter 10 amount of balances: ");
        for(int i=0; i<balance.length; i++)
        {
            System.out.print("Balance " + (i+1) + ": ");
            balance[i] = input.nextDouble();
        }

        System.out.print("Balances: ");
        System.out.println(Arrays.toString(balance));

        System.arraycopy(balance, 0, copyBalance, 0, balance.length);
        System.out.println("Copy of balances is made ...");

        System.out.println("Balances sorted in ascending order: ");
        Arrays.sort(balance);
        System.out.println(Arrays.toString(balance));

        System.out.println("is Balance = copy of Balance? " +
                           Arrays.equals(balance,copyBalance));

        int position = Arrays.binarySearch(balance, 10000);
        if (position>=0)
            System.out.println(10000 + " is found at position " +
                               (position+1));
        else
            System.out.println(10000 + " is not found.");
    }
}
```

Q3. Challenge program

In cryptography, a Caesar cipher, also known as the shift cipher, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on.

Write a Java program to encrypt a plaintext to ciphertext using Caesar cipher. Then decrypt ciphertext back to plaintext. And finally compare whether the decrypted message and the original message are identical.

Note: It is not compulsary to submit the answer of this question. This question is designed to improve your skills in programming.

Sample of output:

```
Original message is : prince sattam bin abdulaziz university
Enter shift value : 3
Encrypted text is : sulqfh vdwwdp elq degxodclc xqlyhuvlwb
Decrypted text is : prince sattam bin abdulaziz university
Are decrypted message and original message identical? true
```

Java Lab Exercise #04	Pass by Value, Pass by Reference and Arrays (Solution)	Evaluation:
Student Name & Number		
Instructor:		Date:

Q1. What is the output of the following program? (4 marks- a 2.5, b 1.5)

```
a.
import java.util.ArrayList;
import java.util.Arrays;
public class ArrayEx
{
    public static void main(String[] args) {

        ArrayList<String> carList = new ArrayList<String>();
        carList.add("BMW");
        carList.add("Ford");
        carList.add("Chevrolet");
        carList.add("Nissan");
        carList.add(1,"Toyota");
        carList.remove("Chevrolet");
        System.out.println("There are "+carList.size()+" cars.");
        System.out.println("List of car names: " +      carList.toString());
        System.out.print("List of car names (another way): ");
        for(String car : carList)
            System.out.printf("%s  ", car);
        System.out.println("\n");

        int[] carModel = {2020, 2011, 2019, 2017, 2013};
        int[] copyModel = new int [carModel.length];

        call_V_R(carModel[3], carModel);
        Arrays.sort(carModel);
        System.out.print("Models are: ");
        for(int i=0; i<carModel.length; i++)
            System.out.printf("%s  ", carModel[i]);
        System.out.println();
        System.arraycopy(carModel, 0, copyModel, 0, carModel.length);
        System.out.println("is carModel = copyModel? " +
                                    Arrays.equals(carModel, copyModel));
    }
    public static void call_V_R(int carElement, int[] carArray) {
        carElement = 2021;
        carArray[0]= 2015;
    }
}
```

Output

```
There are 4 cars.
List of car names: [BMW, Toyota, Ford, Nissan]
List of car names (another way): BMW  Toyota  Ford  Nissan

Models are: 2011  2013  2015  2017  2019
is carModel = copyModel? true
```

b.

```
import java.util.ArrayList;
import java.util.Arrays;
public class ArrayEx

    public static void main(String[] args) {
        Book myBook = new Book();
        myBook.setName("Java How to Program");
        myBook.setPrice(150.0);
        Book [] BookList = new Book[3];

        for(int i = 0; i<BookList.length;i++)
            BookList[i]=new Book();
        BookList[0].setName("Introduction to Java");
        BookList[0].setPrice(200.0);
        BookList[1]=myBook;
        for (Book i:BookList)
            i.print();
    }
}
class Book{
    private String name;
    private double price;

    public void setName(String name) {
        this.name = name;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
    public void print() {
        System.out.println("Book name : "+getName()+", price : "+getPrice() );
    }
}
```

Output

```
Book name : Introduction to Java, price : 200.0
Book name : Java How to Program, price : 150.0
Book name : null, price : 0.0
```

Q2. Writing program (3 marks)

Write a Java program to implement the following:

- Ask the user to enter the names of 3 Books and store them in an ArrayList called *Book*.
- Remove the second book from the list.
- Add the user to enter a new book name and store to be the first book of the list.
- Ask the user to enter a book name, search for it in the list, and print “exist” or “not exist”.
- Print the book name at index 2.
- Print all list items and its size.

```
import java.util.Scanner;
import java.util.ArrayList;
public class Lab4_Program {
    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);
        ArrayList<String> Book = new ArrayList<String>();
         for (int i = 0; i < 3; i++) {
            System.out.println("Enter Book name");
            Book.add(read.nextLine());
         }
        Book.remove(1);

        System.out.println("Enter new Book name");
        Book.add(0, read.nextLine());

         System.out.println("Enter Book name to search for it");
        if (Book.contains(read.nextLine()))
            System.out.println("exist");
        else
            System.out.println("not exist");

        System.out.println("Book name at index 2 is: " + Book.get(2));


        for (String x : Book)
            System.out.println(x);

        System.out.println(Book.size());
    }
}
```

Q3. Writing program (3 marks)

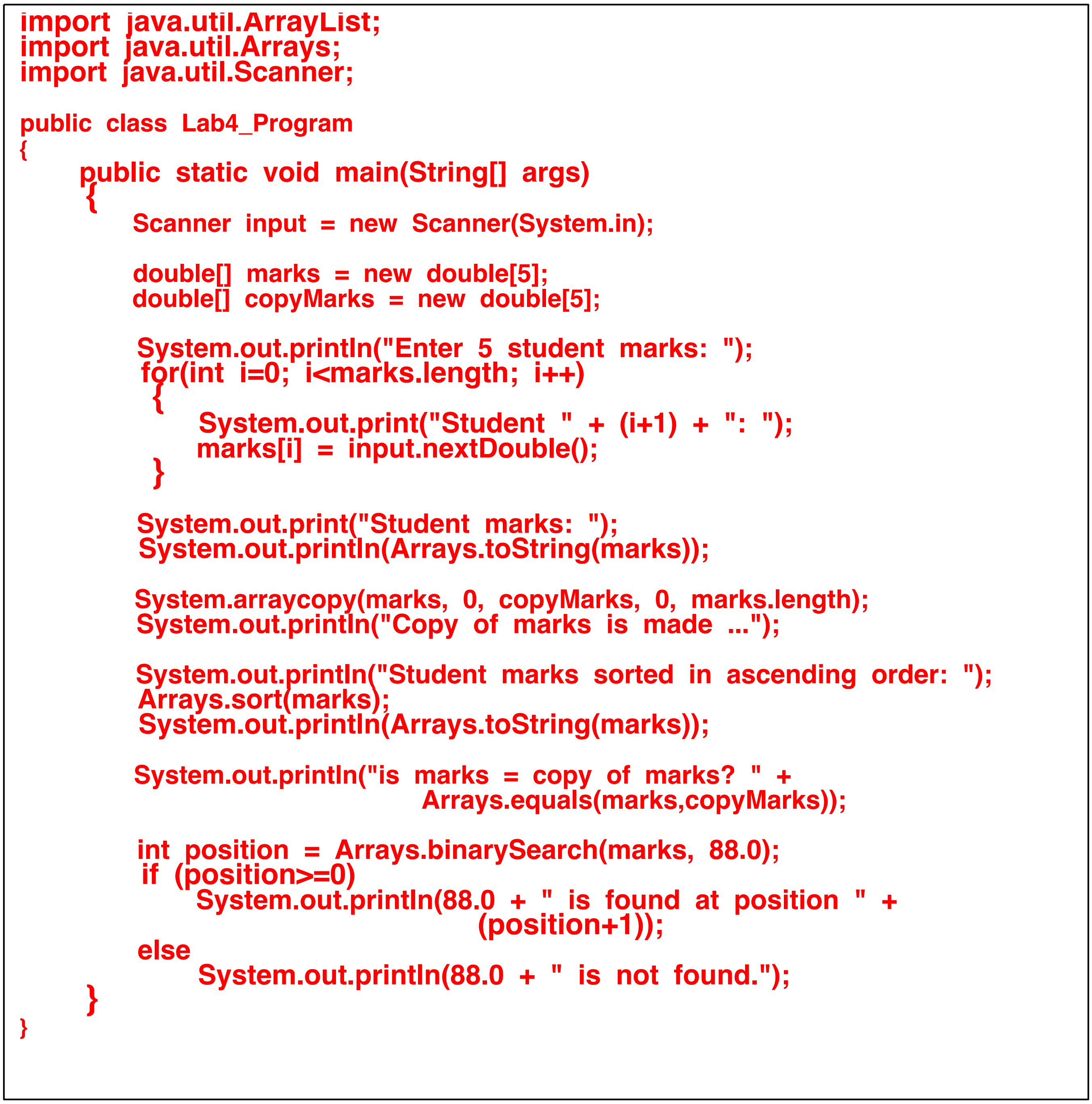
Write a Java program to implement the following by using class **Arrays**:

- Read 5 student marks and keep them in array *marks*.
- Print the elements of array *marks*.
- Copy the content of array *marks* to another array *copyMarks*.
- Sort the array *marks* in ascending order and print the sorted array.
- Compare whether the two arrays *marks* and *copyMarks* are equal.
- Search for the mark 88.0 in the array *markse* and print appropriate message.

Hint: use toString() for print, arraycopy() for copy, sort() for sort, equals() for comparison, and binarySearch() for search.

Sample of output:

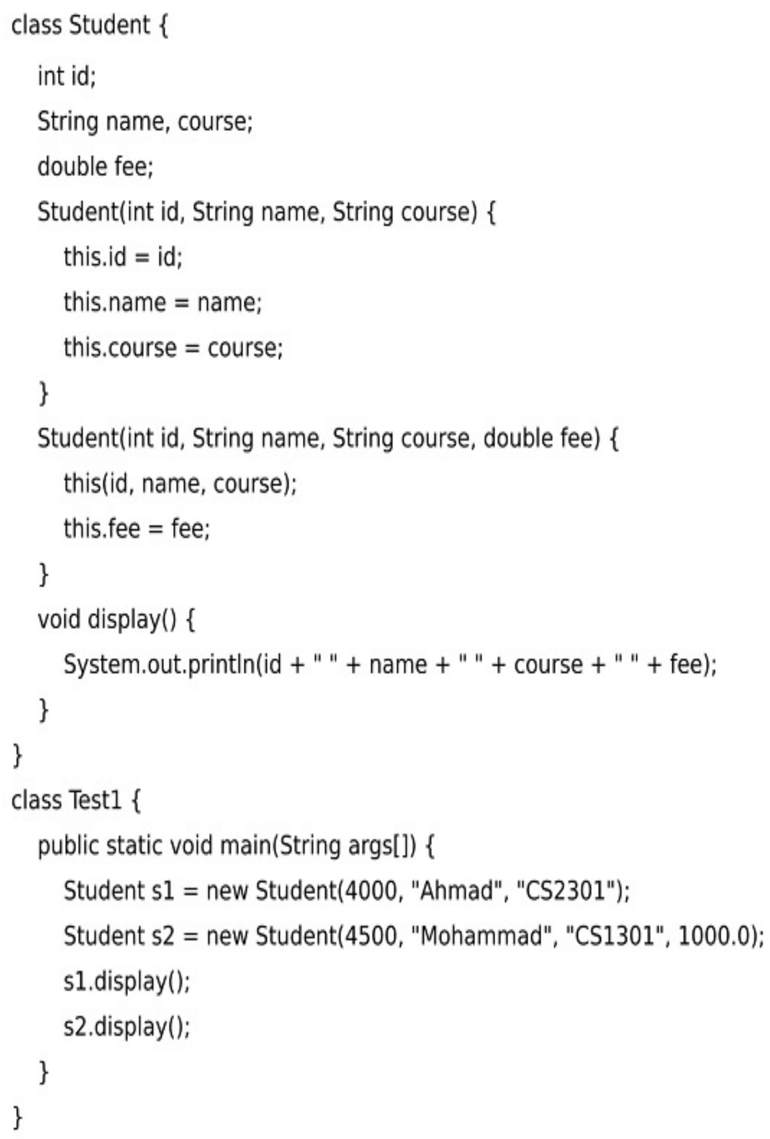
```
Enter 5 student marks:
Student 1: 100
Student 2: 88
Student 3: 70
Student 4: 66
Student 5: 58
Student marks: [100.0, 88.0, 70.0, 66.0, 58.0]
Copy of marks is made ...
Student marks sorted in ascending order:  [58.0, 66.0, 70.0, 88.0, 100.0]
is marks = copy of marks? false
88.0 is found at position 4
```



```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class Lab4_Program
{
     public static void main(String[] args)
     {
          Scanner input = new Scanner(System.in);

          double[] marks = new double[5];
          double[] copyMarks = new double[5];

          System.out.println("Enter 5 student marks: ");
          for(int i=0; i<marks.length; i++)
           {
               System.out.print("Student " + (i+1) + ": ");
               marks[i] = input.nextDouble();
           }

          System.out.print("Student marks: ");
          System.out.println(Arrays.toString(marks));

          System.arraycopy(marks, 0, copyMarks, 0, marks.length);
          System.out.println("Copy of marks is made ...");

          System.out.println("Student marks sorted in ascending order: ");
          Arrays.sort(marks);
          System.out.println(Arrays.toString(marks));

          System.out.println("is marks = copy of marks? " +
                                   Arrays.equals(marks,copyMarks));

          int position = Arrays.binarySearch(marks, 88.0);
          if (position>=0)
               System.out.println(88.0 + " is found at position " +
                                        (position+1));
          else
               System.out.println(88.0 + " is not found.");
     }
}
```

Java Lab Exercise #05	this Reference and Composition	Evaluation:
Student Name & Number		
Instructor:		Date:

Q1. What is the output of the following program?

Program	Output
	4000 Ahmad CS2301 0.0 4500 Mohammad CS1301 1000.0

```
class Student {
    int id;
    String name, course;
    double fee;
    Student(int id, String name, String course) {
        this.id = id;
        this.name = name;
        this.course = course;
    }
    Student(int id, String name, String course, double fee) {
        this(id, name, course);
        this.fee = fee;
    }
    void display() {
        System.out.println(id + " " + name + " " + course + " " + fee);
    }
}
class Test1 {
    public static void main(String args[]) {
        Student s1 = new Student(4000, "Ahmad", "CS2301");
        Student s2 = new Student(4500, "Mohammad", "CS1301", 1000.0);
        s1.display();
        s2.display();
    }
}
```

Q2. Writing program

Define a class named **Student** containing:

- An instance variable named **name** of type String.
- An instance variable named **id** of type int.
- An instance variable named **dept** of type String.
- A full parameterized **constructor**.

Define another class named **Department** containing:

- An instance variable named **dname** of type String.
- A private instance ArrayList named **students** which stores **Student** objects.
- A full parameterized **constructor**.
- A get method for the **students**.

Define a third class named **University** containing:

- An instance variable named **universityName** of type String.
- An private instance ArrayList named **departments** which stores **Department** objects.
- A full parameterized **constructor**.
- A method which returns the total number of students.

A driver class named **AppDemo** contains a main method that test the functionality of the above classes is given below, complete the missing parts in the code:

```
} class AppDemo
} public static void main(String[] args)
;Student s1 = new Student("Ali", 123, "CS")
;Student s2 = new Student("Anas", 222, "CS")
;Student s3 = new Student("Faisal", 1333, "IS")
;Student s4 = new Student("Tariq", 3222, "IS")

making a List of //
.CS Students //
;()<…………>ArrayList<…………..> cs_students = new ArrayList
;(..……)cs_students.add
;(..……)cs_students.add

making a List of //
IS Students //
;()<..……………>ArrayList<……………..> is_students = new ArrayList
;(.……)is_students.add
;(.……)is_students.add

;Department CS = new Department("CS", cs_students)
;Department IS = new Department("IS", is_students)

;()<…………………>ArrayList<…………………..> departments = new ArrayList
;(..…………)departments.add
;(……………)departments.add

.creating an instance of University //
;University university = new University("PSAU", departments)

;System.out.print("Total students in University: ")
;System.out.println(university.getTotalStudentsInUuiversity())
{
{
```

Solution

```
import java.util.*;

// student class
class Student {

   String name;
   int id;
   String dept;

   Student(String name, int id, String dept) {

      this.name = name;
      this.id = id;
      this.dept = dept;

   }
}

/* Department class contains list of student
Objects. It is associated with student
class through its Object(s). */
class Department {

   String dname;
   private ArrayList<Student> students;

   Department(String name, ArrayList<Student> students) {

      this.dname = name;
      this.students = students;

   }

   public ArrayList<Student> getStudents() {
      return students;
   }
}

/* University class contains list of Department
Objects. It is asoociated with Department
class through its Object(s).*/
class University {

   String university;
   private ArrayList<Department> departments;

   University(String university, ArrayList<Department> departments) {
      this.universityName = universityName;
      this.departments = departments;
   }

   // count total students of all departments
```

```
  // in a given university
  public int getTotalStudentsInUniversity() {
     int noOfStudents = 0;
     ArrayList<Student> students;
     for (Department dept : departments) {
        students = dept.getStudents();
        for (Student s : students) {
          noOfStudents++;
         }
     }
     return noOfStudents;
  }

}

// main method
class Test1 {

  public static void main(String[] args) {
     Student s1 = new Student("Ali", 123, "CS");
     Student s2 = new Student("Anas", 222, "CS");
     Student s3 = new Student("Faisal", 1333, "IS");
     Student s4 = new Student("Tariq", 3222, "IS");

     // making a List of
     // CS Students.
     ArrayList<Student> cs_students = new ArrayList<Student>();
    cs_students.add(s1);
    cs_students.add(s2);

     // making a List of
     // IS Students
     ArrayList<Student> is_students = new ArrayList<Student>();
    is_students.add(s3);
    is_students.add(s4);

     Department CS = new Department("CS", cs_students);
     Department IS = new Department("IS", is_students);

     ArrayList<Department> departments = new ArrayList<Department>();
     departments.add(CS);
     departments.add(IS);

     // creating an instance of University.
     University university = new University("PSAU", departments);

     System.out.print("Total students in University: ");
     System.out.println(university.getTotalStudentsInUniversity());
  }}
```

Java Lab Exercise #05	this Refrence and Composition	Evaluation:
Student Name & Number		
Instructor:		Date:

Q1. What is the output of the following program?

Program

```
public class StudentApp {
   public static void main(String[] args) {
      Student S1 = new Student();
      System.out.println(S1);
      Student S2 = new Student("Sara", 123);
      System.out.println(S2);
      Student S3 = new Student("Nora");
      System.out.println(S3);
      Student S4 = new Student(176);
      System.out.println(S4);
      Student S5 = new Student(S2);
      System.out.println(S5);
   } }
class Student {
   private int ID;
   private String name;
   public Student() {
      this("", 0);    }
   public Student(String name) {
      this(name, 0);    }
   public Student(int id) {
      this("", id);    }
   public Student(String name, int ID) {
      this.name = name;
      this.ID = ID;    }
   public Student(Student stud) {
      this(stud.getName(), stud.getID());    }
   public int getID() {
      return ID;    }
   public void setID(int ID) {
      this.ID = ID;    }
   public String getName() {
      return name;    }
   public void setName(String name) {
      this.name = name;    }
   public String toString() {
      return String.format("Name: %-8s ID: %d", name,ID);
   }  }
```

Output:

```
Name:          ID: 0
Name: Sara     ID: 123
Name: Nora     ID: 0
Name:          ID: 176
Name: Sara     ID: 123
```

Q2. Writing program

Define a class named **Job** containing:

- An instance variable named **title** of type string, initialized to null.
- An instance variable named **rank** of type int, initialized to 0.
- A full parameterized **constructor**.
- set and get method for the **title**.
- set and get methods for the **rank**.
- **toString** method that returns the job title and rank.

Define another class named **Employee** containing:

- An instanc variable named **ID** of type int, initialized to 0.
- An instanc variable named **name** of type string, initialized to null.
- An instance object named **empJob** of type Job.
- A full parameterized **constructor**.
- set and get method for the **ID.**
- set and get methods for the **name**.
- set and get methods for the **empJob**.
- A method named **display** that prints all employee's using the method **toString** of the object empJob.

Define a third class named **EmployeeApp** contains a main method that test the functionality of the above classes and call the required methods to generate the sample output shown.

Sample output:

```
ID : 1, Name : Sara Ali, Job title: Programmer, Rank :6
ID : 2, Name : Noura Fahad, Job title: Developer, Rank :8
```

Solution

```
package employeeapp;
public class EmployeeApp {
   public static void main(String[] args) {
      Employee emp1 = new Employee(1, "Sara Ali", new Job("Programmer", 6));
      Employee emp2 = new Employee(2, "Noura Fahad", new Job("Developer", 8));
      emp1.display();
      emp2.display();
   }
}

class Job {
   private String title;
   private int rank;
   public Job(String title, int rank) {
      this.title = title;
      this.rank = rank;
   }
   public String getTitle() {
      return title;
   }
   public void setTitle(String title) {
      this.title = title;
   }
   public int getRank() {
      return rank;
```

```
    }
    public void setRank(int rank) {
        this.rank = rank;
    }
    public String toString() {
        return ("Job title: " + getTitle() + ", Rank :" + getRank());
    }
}
class Employee {
    private int id;
    private String name;
    private Job Empjob;
    public Employee(int id, String name, Job Empjob) {
        this.id = id;
        this.name = name;
        this.Empjob = Empjob;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Job getEmpjob() {
        return Empjob;
    }
    public void setEmpjob(Job Empjob) {
        this.Empjob = Empjob;
    }
    public void display() {
        System.out.println("ID : " + id + ", Name : " + name + ", " + Empjob.toString());
    } }
```

;Java Lab Exercise #07	Inheritance – Part 1	Evaluation:
Student Name & Number		
Instructor:		Date:

Q1. What is the output of the following programs (3 marks, 0.25 for each, a 1.5, b 1.5)

Program

Output

```
a
public class InheritanceQ{
    public static void main(String[] args) {
    CommunityMember c = new CommunityMember("Fahd", 1234);
    Employee e = new Employee("Ahmed", 1235, 5000.75f);
    Faculty f = new Faculty("Omar", 1236, 4000, 16);
    }
}
class CommunityMember {
    private String name;
    private int ID;
    public CommunityMember() {
        System.out.println("Community Member");
    }
    public CommunityMember(String name, int ID) {
        this();
        this.name = name;
        this.ID = ID;
    }
}
class Employee extends CommunityMember {
    private float salary;
    public Employee() {
        System.out.println("Employee");
    }
    public Employee(String name, int ID, float salary) {
        this();
        this.salary = salary;
    }
}
class Faculty extends Employee {
    private int numHours;
    public Faculty(String name, int ID, float salary, int
numHours) {
        super();
        System.out.println("Faculty");
        this.numHours = numHours;
    }
}
```

```
Community Member
Community Member
Employee
Community Member
Employee
Faculty
```

```
b
public class InheritanceQ1 {
    public static void main(String[] args) {
    public static void main(String[] args) {
        A a = new A();
        a.methA("#");
        a.methA(6);

        B b= new B();
        b.methA("Inheritance");
        b.methA(1);
    }
}
```

```
Lab#7
Topic: Inheritance
2
```

```
class A {
    public void methA(int var) {
        System.out.println(++var);
    }
    public void methA(String var) {
        System.out.print("Lab" + var);
    }
}
class B extends A {
    public void methA(String var) {
        System.out.println("Topic: " + var);
    }
}
```

Q2. Write a program

⍰ Define a class name **Person** with the following specifications:

a) A private variable **name** of type String. **(**0.25 mark **)**
b) A private variable **age** of type int. **(**0.25 mark **)**
c) A zero-parameterized constructor to initialize all the variables to default. **(**0.25 mark **)**
d) A full parameterized constructor to initialize the variables. **(**0.25 mark **)**
e) A public method **setPerson** with 2 parameters String and int to set name and age instance variables. **(**0. 5 mark **)**
f) A public method **toString** to return the person data. **(**0.5 mark **)**

⍰ Define another class name **Student** which is extended form **Person** with the following specifications:

a) A private variable **gpa** of type double. **(**0.25 mark **)**
b) A zero-parameterized constructor which calls super class zero-parameter constructor, and also initializes gpa to zero. **(**0.5 mark **)**
c) A full parameterized constructor which calls the super class parameterized constructor and initialize all the variables. **(**0.5 mark **)**
d) A setter and getter method for gpa variable. **(**0.5 mark **)**
e) A public method **toString** to return the student data, this method should call the toString method from the supper class. **(**0.5 mark **)**

⍰ Define a Java class **Test** that contains a **main method** and test the functionality of the above classes as the following:

- Define an array **S** of type **Student** with size 2**. (**0.25 mark **)**
- In the first element of the array S:
    - instantiate an object of type **Student** using zero-parameterized constructor. **(**0.5 mark **)**
    - change the **name** of that object to **"Nora"** and the **age** to **19** using **setPerson** method. (0.5 mark)
    - Ask the user to enter a student **gpa** and store it in that object. **(**0.5 mark **)**
- In the second element of the array S, instantiate an object of type **Student** using full parametrized constructor with these information: **(**0.5 mark **)**

name	**age**	**gpa**
Sara	20	3.5

- Print all students data. **(**0.5 mark **)**
- sample output:

```
Enter new gpa: 4.5
Name is: Nora, Age is :19, GPA is: 4.5
Name is: Sara, Age is :20, GPA is: 3.5
```

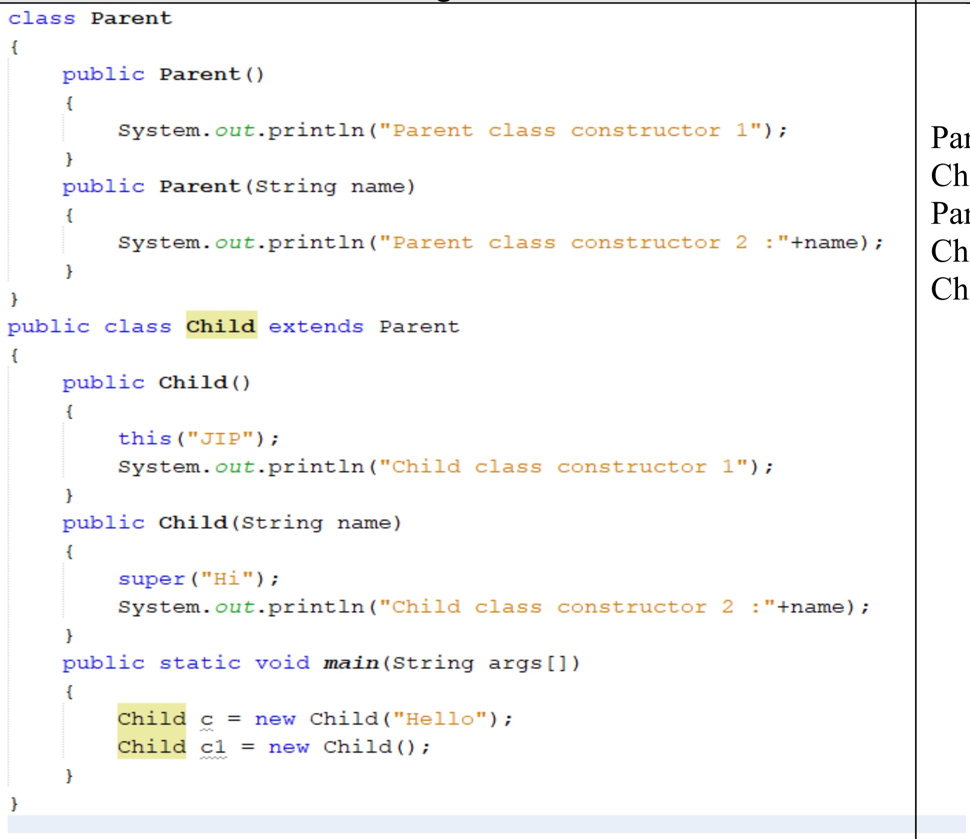
```
import java.util.Scanner;
public class CS2310Prog
{
   public static void main(String[] args)
   {
      Scanner myInput = new Scanner(System.in);
      Student [] Students = new Student[2];
      Students[0] = new Student();
      Students[0].setPerson("Nora", 19);
      System.out.print("Enter new gpa: ");
      double gpa = myInput.nextDouble();
      Students[0].setGpa(gpa);
      Students[1] = new Student("Sara", 20, 3.5);

      for( Student item: Students )
        System.out.println(item);
   }}
class Person
{
   private String name;
   private int age;
   public Person()
   {
      name = null;
      age = 0;
   }
   public Person( String name, int age )
   {
      this.name = name;
      this.age = age;
   }
   public void setPerson(String name, int age )
   {
      this.name = name;
      this.age = age;
   }
   public String toString()
   {
      return "Name is: " + name + ", Age is :" + age ;
   }
}

class Student extends Person
{
   private double gpa;
   public Student()
```

```
  {
     super();
    gpa = 0;
  }
  public Student( String name, int age, double gpa )
  {
     super(name, age);
     this.gpa = gpa;
  }
  public double getGpa() {
     return gpa;
  }
  public void setGpa(double gpa) {
     this.gpa = gpa;
  }
  @Override
  public String toString()
   {
     return super.toString()+", GPA is: " + getGpa() ;
  }
}
```

Lab7	Inheritance	Evaluation:
Student Name & Number		

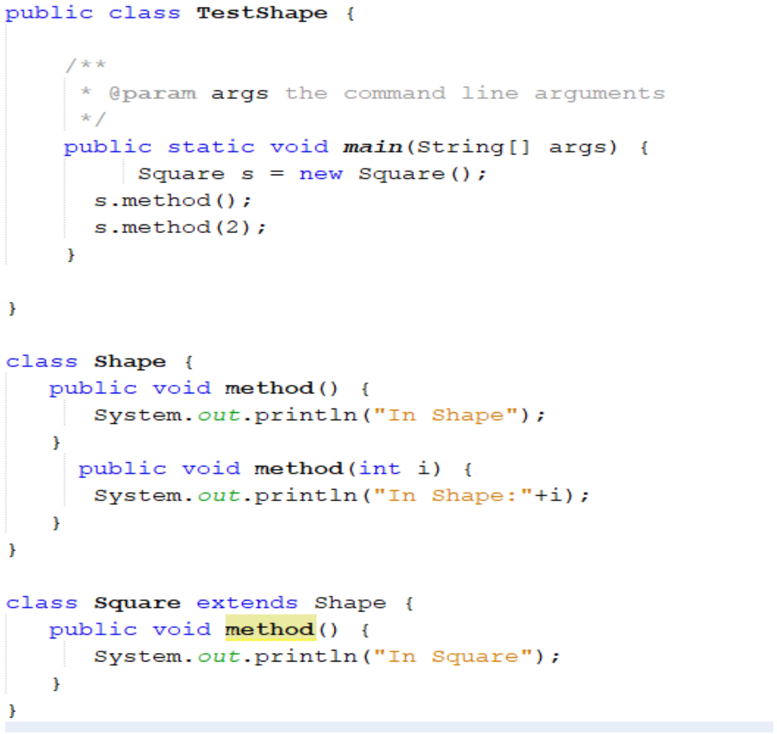
Q1. What is the output of the following program?

Program | **Output**



```
class Parent
{
    public Parent()
    {
        System.out.println("Parent class constructor 1");
    }
    public Parent(String name)
    {
        System.out.println("Parent class constructor 2 :"+name);
    }
}
public class Child extends Parent
{
    public Child()
    {
        this("JIP");
        System.out.println("Child class constructor 1");
    }
    public Child(String name)
    {
        super("Hi");
        System.out.println("Child class constructor 2 :"+name);
    }
    public static void main(String args[])
    {
        Child c = new Child("Hello");
        Child c1 = new Child();
    }
}
```

```
Parent class constructor 2 :Hi
Child class constructor 2 :Hello
Parent class constructor 2 :Hi
Child class constructor 2 :JIP
Child class constructor 1
```

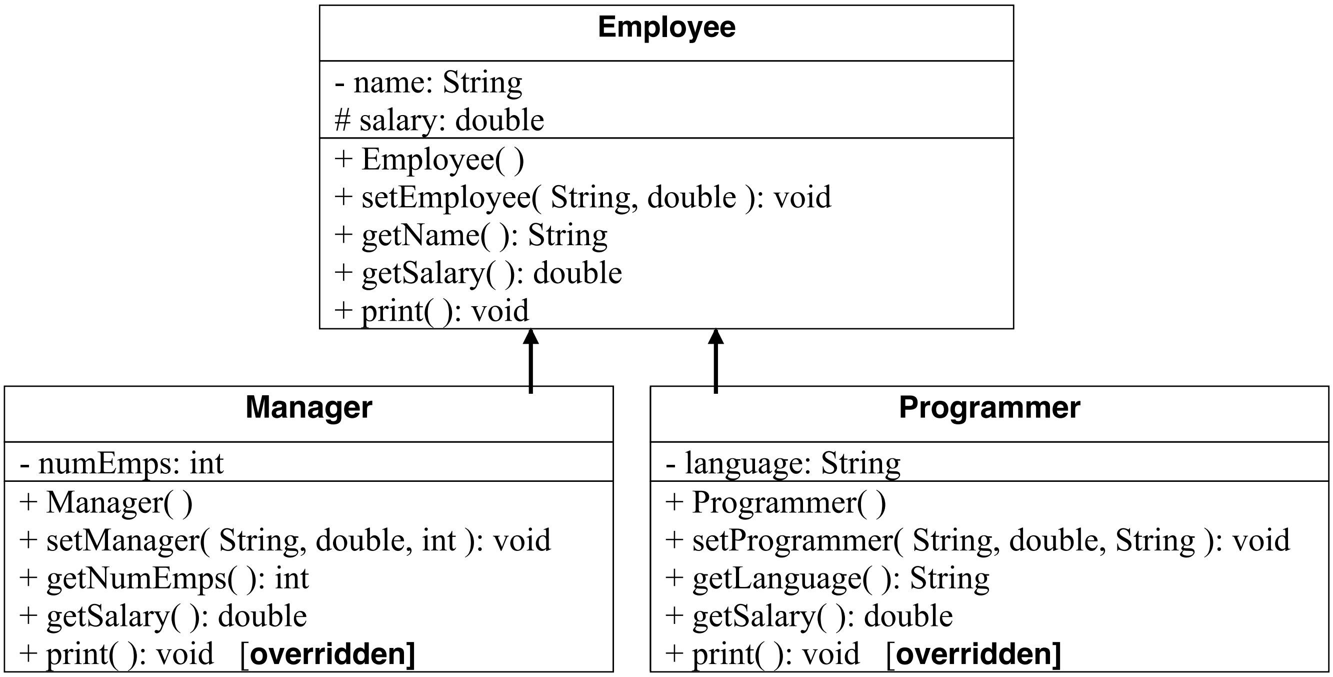


```
public class TestShape {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
          Square s = new Square();
       s.method();
       s.method(2);
    }

}

class Shape {
   public void method() {
      System.out.println("In Shape");
   }
     public void method(int i) {
      System.out.println("In Shape:"+i);
   }
}

class Square extends Shape {
   public void method() {
      System.out.println("In Square");
   }
}
```

```
In Square
In Shape:2
```

Q#2:

Exercise: **Three UML class diagrams are given below. Class Employee is the superclass and the other two classes, Manager and Programmer, are subclasses of Employee.**



Employee

- name: String
# salary: double

+ Employee( )
+ setEmployee(String, double): void
+ getName( ): String
+ getSalary( ): double
+ print( ): void

Manager

- numEmps: int

+ Manager( )
+ setManager(String, double, int): void
+ getNumEmps( ): int
+ getSalary( ): double
+ print( ): void **[overridden]**

Programmer

- language: String

+ Programmer( )
+ setProgrammer(String, double, String): void
+ getLanguage( ): String
+ getSalary( ): double
+ print( ): void **[overridden]**

Note: The symbols -, #, and + mean private, protected, and public members respectively.

Answer the following questions based on the above diagrams:

1. Write the Java statement that makes the class **Manager** inherits the class **Employee**.

```
class  Manager  extends  Employee
```

2. Implement the default constructor of the class **Manager**.

```
public  Manager(  )
{
    super(  );
    numEmpls  =  0;
}
```

3. Implement the method setEmployee(…) of the class **Employee**.

```
public  void  setEmployee(  String  n,  double  s  )
{
    name  =  n;
    salary  =  s;
}
```

4. Implement the method setProgrammer(…) of the class **Programmer**.

```
public void setProgrammer( String n, double s, String lang )
{
    setEmployee( n, s );
    language = lang;
}
```

5. Implement the method getSalary( ) of the class **Manager**.

```
public double getSalary( )
{
    return salary;
}
```

6. Implement the overridden method print( ) of the class **Manager**.

```
public void print( )
{
    super.print();
    system.out.println("Number of employees is "+ getNumEmps());
}
```

7. Declare an object named m of type **Manager**. Use the method setManager(…) to set name, salary, and number of employees of object m to "Saleh Khalid", 17310, and 35.

```
Manager m = new Manager();
m.setManager("Saleh Khalid", 17310, 35);
```

8. Declare an object named p of type **Programmer**. Use the method setProgrammer(…) to set name, salary, and language of object p to "Ahmed Hasan", 8550, and "Java".

```
Programmer p = new Programmer();
p.setProgrammer("Ahmed Hasan", 8550, "Java");
```

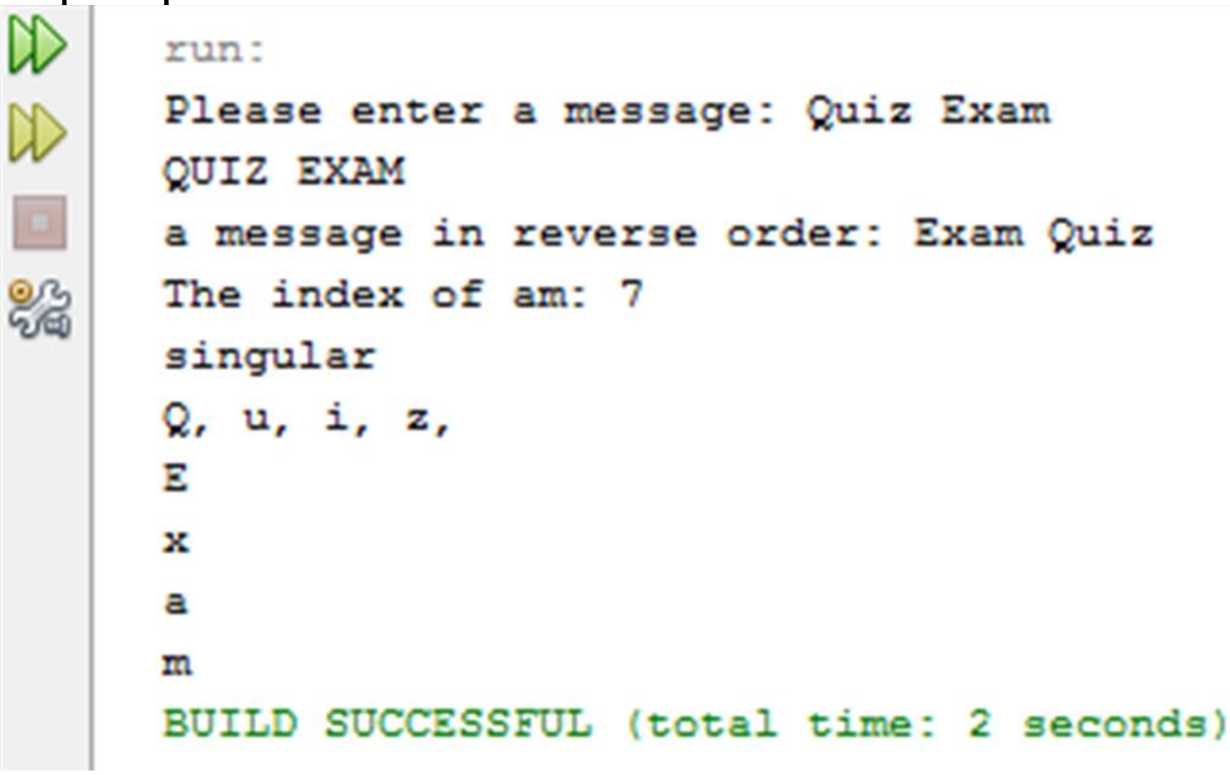
9. Declare a reference variable named e of type **Employee**. Make e a reference to m. What e.print() will print? Then make e a reference to p. What e.print() will print?

```
Employee  e;
e  =  m;
e.print();      ⍰      Saleh  Khalid,  17310,  35

e  =  p;
e.print();      ⍰      Ahmed  Hasan,  8550,  Java
```

Java Lab Exercise #10	Strings & Characters	Evaluation:
Student Name & Number		
Instructor:		Date:

Write a complete Java program to implement the following:

- Read a string from the user and store it in a String variable **msg**.
- Print **msg** as capital letters.
- Print the string in **msg** in reverse order. (hint: use *substring* and *indexOf* method)
- Create an array of characters **arr** with size 4.
- Store the first 4 characters of **msg** in the array **arr.** (hint: use *getChars* method)
- Print the index of "am" in **msg**. (hint: use *indexOf* method)
- Print "plural" if **msg** ends with "s", otherwise print "singular". (hint: use *endsWith* method)
- Print the content of **arr** in one line separated by a comma.
- Print the letters of **msg** from the 5th letter to the last letter. Each letter should be printed in one line.
- **Sample Output:**



```
run:
Please enter a message: Quiz Exam
QUIZ EXAM
a message in reverse order: Exam Quiz
The index of am: 7
singular
Q, u, i, z,
E
x
a
m
BUILD SUCCESSFUL (total time: 2 seconds)
```

```
System.out.print("Please enter a message: ");
        Scanner read = new Scanner(System.in);
        String msg = read.nextLine();

        System.out.println(msg.toUpperCase());
        String first, second;
        first = msg.substring(0 , msg.indexOf(" "));
```

```
second = msg.substring(msg.indexOf(" ")+1,msg.length());

System.out.print("a message in reverse order: ");
System.out.println(second +" " +first );

char[] arr = new char[4];
msg.getChars(0, 4, arr, 0);
System.out.println("The index of am: "+ msg.indexOf("am"));
System.out.println(msg.endsWith("s")?"plural":"singular");

for(char a: arr){
   System.out.print(a+", ");
}
System.out.println();
for(int i =5; i<msg.length() ; i++){
System.out.println(msg.charAt(i));
}
```