1. Write a java program that has a class named emp13 which has four members, firstname and lastname as private members, and rate and hours as public members. The constructor will assign value to the rate and hour member. The class also has a method named wagecalulate which will calculate the wage of an employee by using the following hours*rate. If the employee works more than 40 hours then the method will calculate the overtime. The overtime wage will be 1.5 times more than the regular rate. Call the wagecalculate method in the main class.

   Employee = 45 hours

   Overtime= Total hours-40;

   Overtime wage should be 1.5 times more than regular wage

   Overtime =5 *1.5* rate

   Overtime= regular time wage + Overtime wage = hours*rate+ Overtime*1.5*rate

```
class employee13{
    private String firstname;
    private String lastname;
    public double rate;
    public double hours;
    employee13(double rate, double hours){
        this.rate=rate;
        this.hours=hours;
    }
    void setfirstname(String firstname) {
        this.firstname=firstname;
    }
    String getfirstname() {
        return firstname;
    }
    void setlastname(String lastname) {
        this.lastname=lastname;
    }
    String getlastname() {
        return lastname;
    }
    double wagecalculate() {
        double overtime, hour;
```

```
        overtime=hours-40; // overtime hour
        hour= hours-overtime// to get the regular
hour
        if(hours>40) {

    return(hour*rate)+(overtime*1.5*rate);//regular
hour wage+overtime wage
        }
        else {
            return hour*rate;//regular hour wage
        }
    }


}

public class emp13 {

    public static void main(String[] args) {
        employee13 emp= new employee13(10.5, 45);//
TODO Auto-generated method stub
        employee13 emp1 = new employee13(12,40);
        emp1.setfirstname("Mahbub");
        emp1.setlastname("Hasan");
        System.out.println(emp1.getfirstname());
        System.out.println(emp1.getlastname());
        System.out.println("The wage is
"+emp.wagecalculate());
        System.out.println("The wage
is"+emp1.wagecalculate());
    }

}
```

In this code we are using setter and getter to access the private members

2. Write a java program that has a class named staticexample which has a static member , a static method which will calculate the square root from 1 to the member's value, a static block which will print out "static block initialized". Call the static method inside the main class

```
class staticexample2{
    static double n=10;// this is a static member
    static void squareroot() {// this is a static method
        double s;
        for(int i=1; i<=10; i++) {
            s=Math.sqrt(i);
            System.out.println(s);
            }

        }
        static {// this is a static block
            System.out.println("static block initialized");
        }
    }
    public class staticexample {

        public static void main(String[] args) {
            staticexample2.squareroot();//

        }

}
```

Since squareroot is a static method, we do not need to create any object to access that method. We can access it with the classname.methodname().If there is a static block

, then it will be executed when we will call the static method. The static block will be executed first then the method result will get executed.

3. Write a java program that has a class named staticexample which has a static member , a static method which will calculate the square root from 1 to the member's value, a static block which will add two numbers and print the value . The static method will use method overloading and the 2nd static method will square the value from 1 to n. Call the static method inside the main class

```
class staticexample2{
    static double n=10;
    static void squareroot() {
        double s;
        for(int i=1; i<=10; i++) {
            s=Math.sqrt(i);
            System.out.println(s);
        }

    }
    static void squareroot(int n) { // here static
method squareroot is using method overloading

        double s1;
        for(int i=1; i<=n; i++) {
            s1=i*i;
            System.out.println(s1);
        }
    }
    static {
        double x,y;
        x=5;
        y=6;
        double z;
        z=x+y;
```

```
        System.out.println("The static value is "
+z);

    }
}
public class staticexample {

    public static void main(String[] args) {
        staticexample2.squareroot();// TODO Auto-
generated method stub
        staticexample2.squareroot(10);
    }

}
```

Here squareroot is a static method so we don't need to create any object for that. If you want to use a static method for method overloading, do not forget to add the static keyword before the function return type.

4. Write a java program which will have a class named inn that has a static array member, a normal member n, the constructor will assign the value to the members. inn class has a static inner class which will have a method to determine the smallest value from the array member. Inn class also has another class named squareroot which will have a member s and a method to determine the square root values from 1 to the member n. Inn class has a method named analyze which will call the inner classes. Call the analyze method

```
class inn{
    static int arr[];
    int n;
    inn(int arr[], int n){
        this.arr=arr;
        this.n=n;
    }
    static class smallest{ // here smallest is an
inner static class
        static double small() {// if the class is
static, then methods and members of that class has to
be static
```

```
            double s=arr[0];
            for(int i=0; i<arr.length; i++) {
                if(arr[i]<s) {
                    s=arr[i];
                }
            }
            return s;
        }
    }
    class squareroot{// squareroot is an inner class
        double s;
        void sroot() {
            for(int i=1; i<=n; i++) {
                s=Math.sqrt(i);
                System.out.println(s);

            }

        }
    }
    void analyze() {// we are using analyze method
inside the inn class so that we can call the inner
classes
        squareroot s= new squareroot();// we are
creating object to access the class squareroot
        s.sroot();
        System.out.println("smallest value is
"+smallest.small());// since smallest class is a
static class, so we do not create any object for
that.
    }
}

public class innerclassexample {
```

```
    public static void main(String[] args) {
        inn i= new inn(new int[] {45,23,12, 1,
5,6,7}, 10);// TODO Auto-generated method stub
        i.analyze(); // by using the analyze method,
the main class is calling the inner classes
    }

    }
```