//PATIL HIMANSHU MANOHAR (210550381061)

Basic Programming Concepts (Java) ASSIGNMENT 1

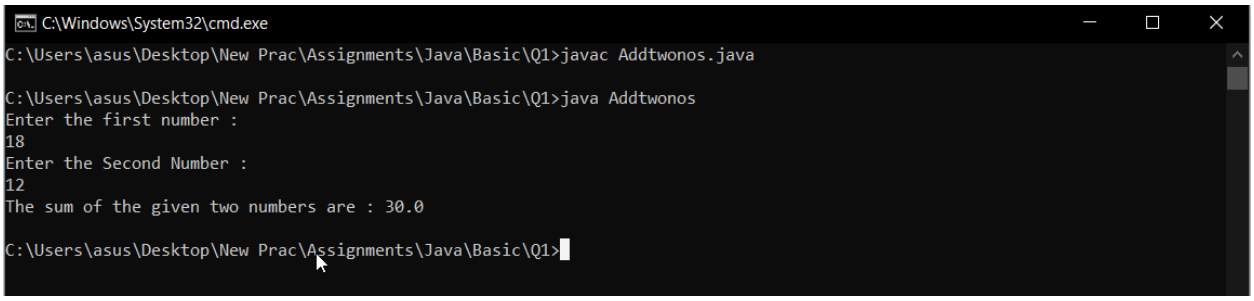
Q1)

a) Print Hello World.

```
public class HelloWorld
{
	public static void main(String args [])
	{
		System.out.println("Hello World");
	}
}
```

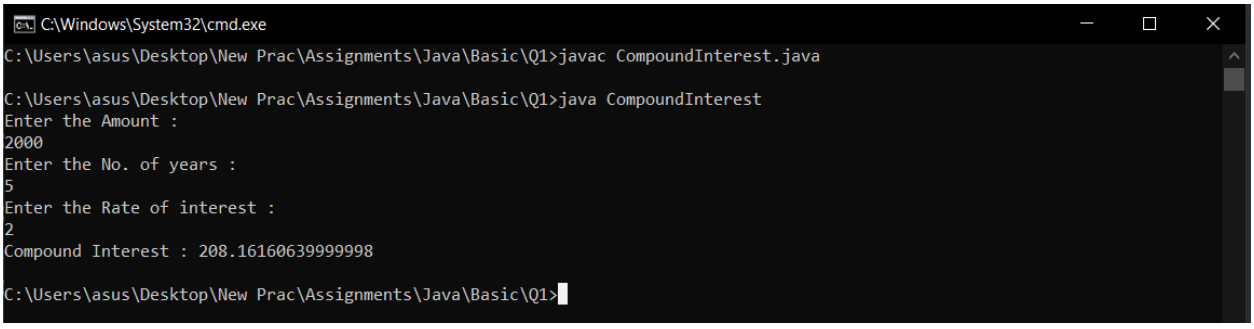
b) Add two numbers/binary numbers/characters.

```
import java.util.Scanner;
public class Addtwonos
{
	public static void main(String args [])
	{
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter the first number : ");
			double a = sc.nextDouble();

		System.out.println("Enter the Second Number : ");
			double b = sc.nextDouble();

		double result = a+b;
			System.out.println("The sum of the given two numbers are : "+result);
	}
}
```



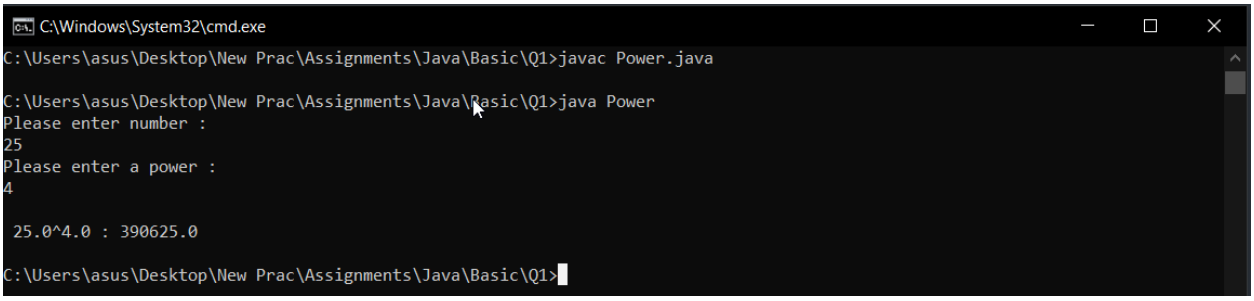
c) Calculate compound interest.

```
import java.util.*;
public class CompoundInterest
{
	public static void main(String args [])
	{
		double p,n,r,ci;

		Scanner s = new Scanner(System.in);

		System.out.println("Enter the Amount : ");
		p = s.nextDouble();

		System.out.println("Enter the No. of years : ");
		n = s.nextDouble();

		System.out.println("Enter the Rate of interest : ");
		r = s.nextDouble();

		ci = p * Math.pow(1.0 + r / 100.0, n) - p;

		System.out.println("Compound Interest : "+ci);

	}
}
```



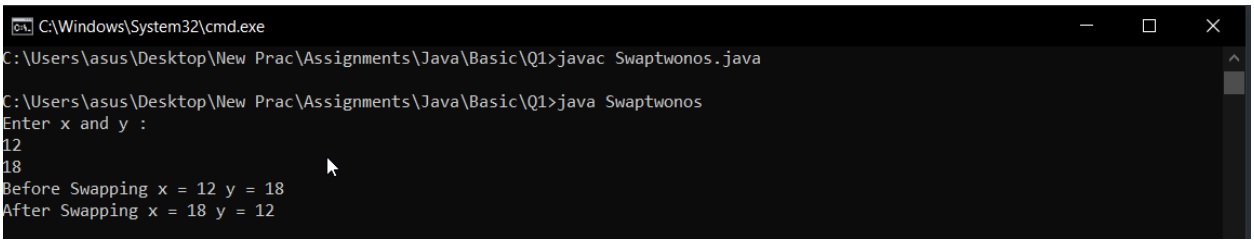
d) Calculate power of a number.

```
import java.util.Scanner;
public class Power
{
	public static void main(String args [])
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Please enter number : ");
		double n =sc.nextDouble();

		System.out.println("Please enter a power : ");
		double power =sc.nextDouble();

		System.out.println("\n "+n+"^"+power+" : "+Math.pow(n,power));
		sc.close();
	}
}
```



e) Swap two numbers.

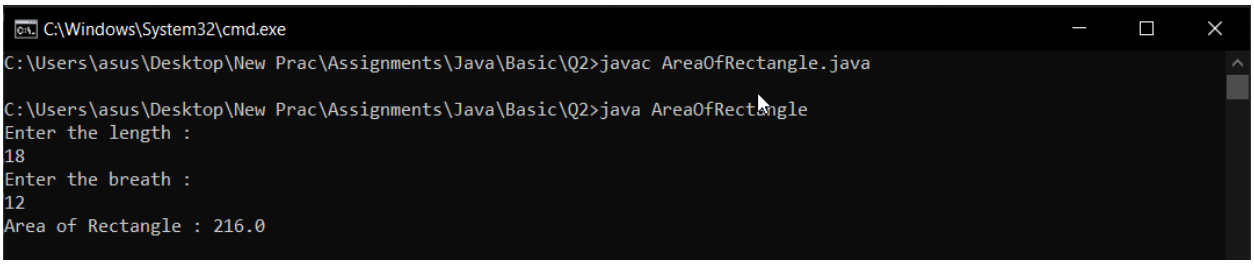
```
import java.util.Scanner;
public class Swaptwonos
{
	public static void main(String args [])
	{
		int x,y,temp;
		System.out.println("Enter x and y : ");
		Scanner sc = new Scanner(System.in);

		x = sc.nextInt();
		y = sc.nextInt();

		System.out.println("Before Swapping x = " + x + " y = " +y);

		temp = x;
		x = y;
		y = temp;

		System.out.println("After Swapping x = " + x + " y = " +y);
	}
}
```



Q2)

a) Calculate area of rectangle.

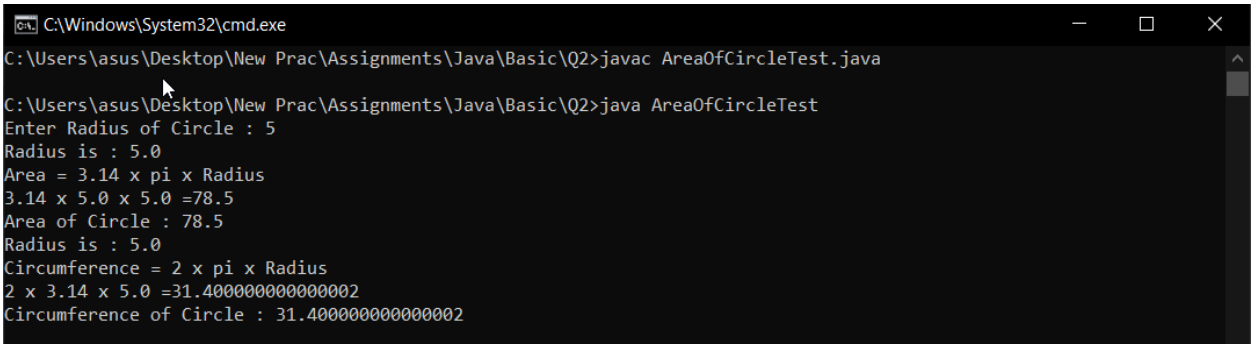
```
import java.util.Scanner;
public class AreaOfRectangle
{
	public static void main(String args [])
	{
		double l,b,a;
		Scanner sc =  new Scanner(System.in);

		System.out.println("Enter the length : ");
		l = sc.nextDouble();

		System.out.println("Enter the breath : ");
		b = sc.nextDouble();

		a = l * b;

		System.out.println("Area of Rectangle : "+a);
	}
}
```



b) Calculate area and circumference of circle using multiple classes.

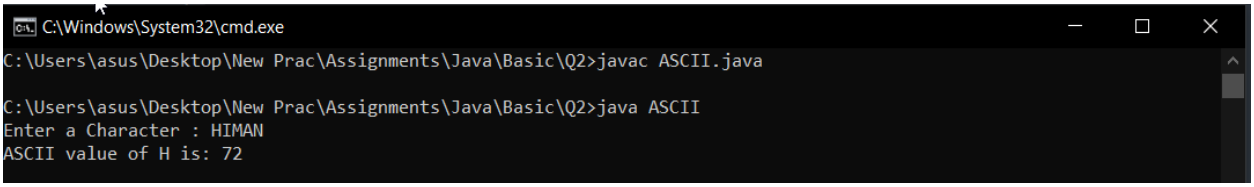
```
import java.util.*;
public class AreaOfCircleTest{

  public static void main(String[] args)
  {
     Scanner sc = new Scanner(System.in);
     System.out.print("Enter Radius of Circle : ");
     double radius = sc.nextDouble();

     AreaOfCircle ac = new AreaOfCircle();
     double area = ac.calculateArea(radius);

     System.out.println("Radius is : "+radius);
     System.out.println("Area = 3.14 x pi x Radius");
     System.out.println("3.14 x "+radius+" x "+radius+" ="+area);
     System.out.println("Area of Circle : "+area+" ");

     AreaOfCircumference acc = new AreaOfCircumference();
     double circumference = acc.calculateCircumference(radius);

     System.out.println("Radius is : "+radius);
     System.out.println("Circumference = 2 x pi x Radius");
     System.out.println("2 x 3.14 x "+radius+" ="+circumference);
     System.out.println("Circumference of Circle : "+circumference+" ");


  }
}

class AreaOfCircle{
  double radius, area;
  double pi = 3.14;

  public double calculateArea(double radius)
  {
      area = pi * radius * radius;
      return area;
  }
}

class AreaOfCircumference {
  double radius, circumference;
```

```
  double pi = 3.14;


  public double calculateCircumference(double radius)
  {
     circumference = 2 * pi * radius;
     return circumference;
  }
}
```



c) Java program to find ASCII value of a character.

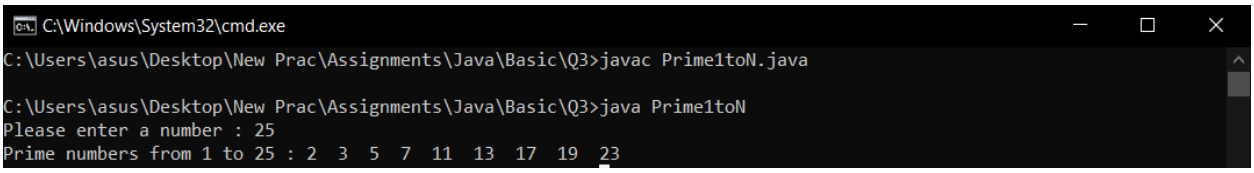
```
import java.util.Scanner;
public class ASCII
{
	public static void main(String args [])
	{
		System.out.print("Enter a Character : ");
		Scanner sc = new Scanner(System.in);
		char chr = sc.next().charAt(0);
		int asciiValue = chr;
		System.out.println("ASCII value of " +chr+ " is: "+asciiValue);
	}
}
```



Q3)

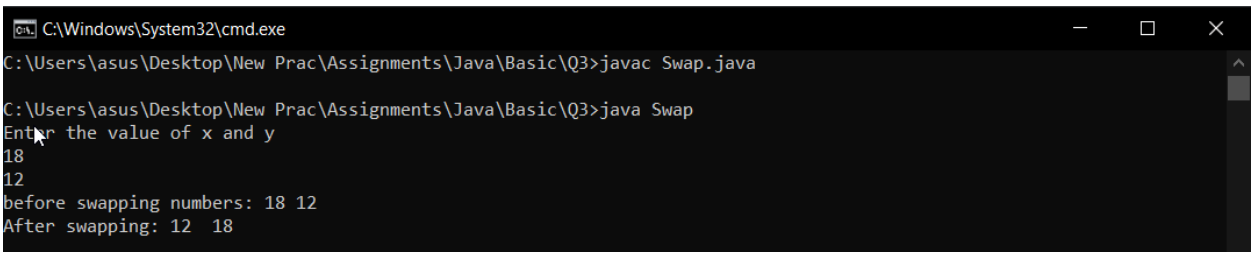
a) Display prime numbers between 1 and 100 or 1 and n.

```
import java.util.Scanner;
public class Prime1toN
{
	public static void main(String[] args)
	{
		int a,x,i,j;
		Scanner sc=new Scanner(System.in);
		System.out.print("Please enter a number : ");
		a=sc.nextInt();


		System.out.print("Prime numbers from 1 to "+a+" : ");
		for(j=2;j<=a;j++)
		{
			x=0;
			for(i=2;i<=j;i++)
				if(j%i==0)
					x++;
			if(x==1)
				System.out.print(j+"  ");
		}
		sc.close();

	}
}
```



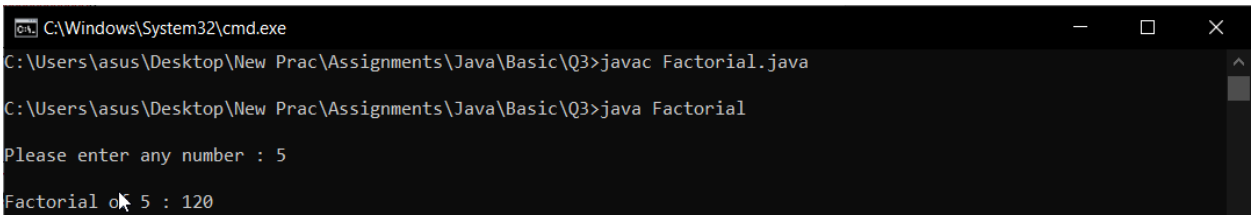
b) Swap two variables without using the third variable.

```
import java.util.*;
class Swap
{
   public static void main(String a[])
   {
                    int x,y;

      System.out.println("Enter the value of x and y");
      Scanner sc = new Scanner(System.in);

      x = sc.nextInt();
      y = sc.nextInt();

      System.out.println("before swapping numbers: "+x +" "+ y);

      x = x + y;
      y = x - y;
      x = x - y;

      System.out.println("After swapping: "+x +"  " + y);
   }
}
```



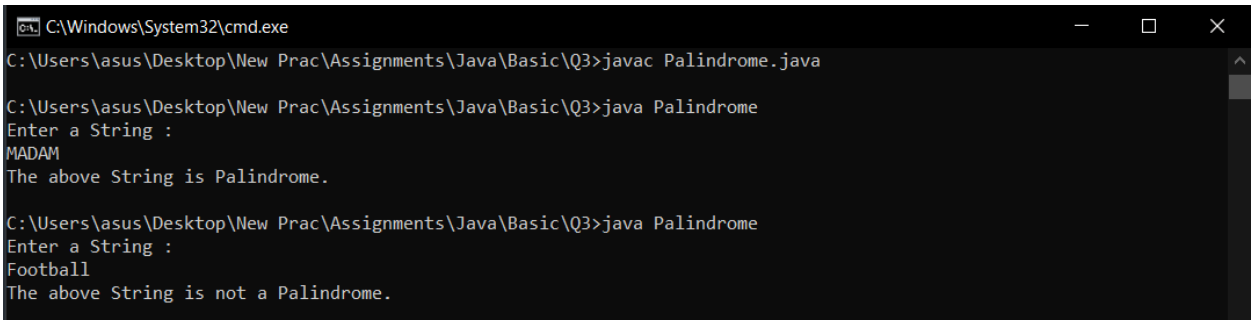
c) Find the factorial of a number.

```
import java.util.Scanner;
public class Factorial
{

	public static void main(String[] args)
	{
		Scanner sc=new Scanner(System.in);
		System.out.print("\nPlease enter any number : ");
		int a=sc.nextInt();

		int fact=1;
		for(int i=1;i<=a;i++)
			fact*=i;

		System.out.print("\nFactorial of "+a+" : "+fact);
		sc.close();
	}
}
```



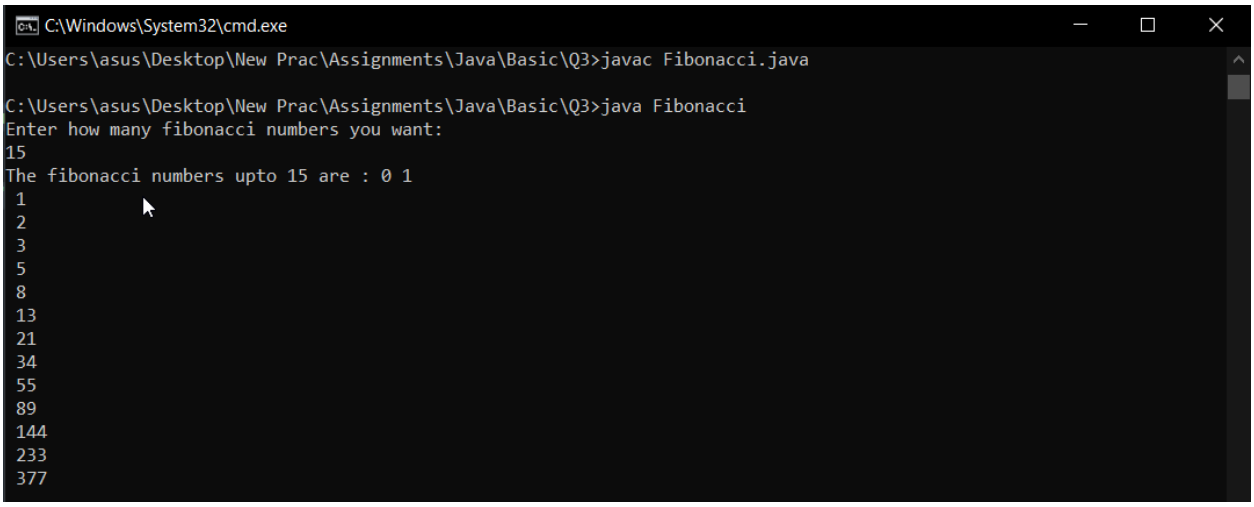
d) Check if a number is palindrome or not.

```
import java.util.*;

class Palindrome
{
	public static void main(String args[])
	{
		String a,b="";
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter a String : ");
		a=sc.nextLine();

		int length = a.length();

		for (int i = length-1; i>=0; i--)
		{
			b=b+a.charAt(i);
		}
		if(a.equals(b)){
			System.out.println("The above String is Palindrome.");
		}
		else{
		System.out.println("The above String is not a Palindrome.");
		}
	}
}
```



e) Print Fibonacci series till n.

```
import java.util.Scanner;
class Fibonacci
{
	public static void main(String args [])
	{
		int n,n1=0,n2=1,temp,i;
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter how many fibonacci numbers you want:  ");
		n = sc.nextInt();

		System.out.println("The fibonacci numbers upto " + n + " are : "+n1 + " " + +n2);


		for (i=1;i<=n-2;i++)
		{
			temp=n1+n2;
			n1=n2;
			n2=temp;
			System.out.println(" "+temp);
		}
	}

}
```

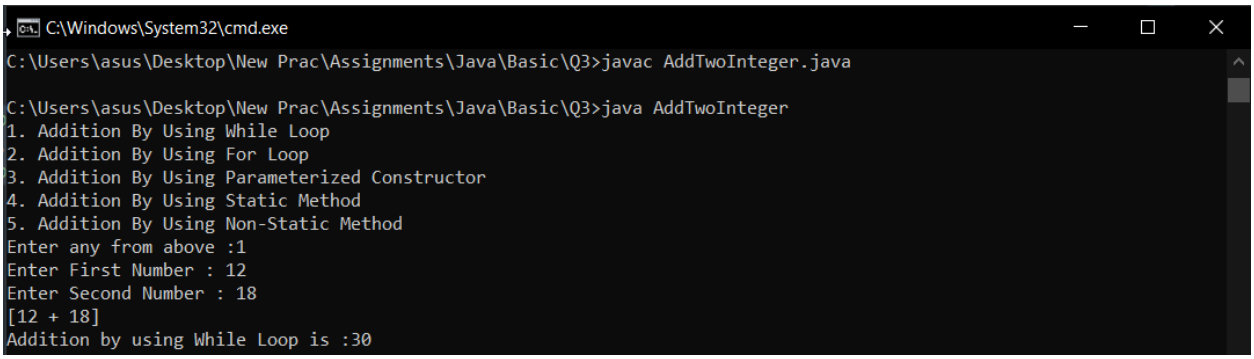


f) Add two integer variables in 5 different ways using functions and control statement.

```
import java.util.*;
public class AddTwoInteger
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
          System.out.println("1. Addition By Using While Loop ");
    System.out.println("2. Addition By Using For Loop ");
    System.out.println("3. Addition By Using Parameterized Constructor");
    System.out.println("4. Addition By Using Static Method");
    System.out.println("5. Addition By Using Non-Static Method");


    System.out.print("Enter any from above :");
    int choice = sc.nextInt();

    System.out.print("Enter First Number : ");
    int firstNo = sc.nextInt();

    System.out.print("Enter Second Number : ");
    int secondNo = sc.nextInt();

    int total=0;
    AddTwoIntegerVar add = new AddTwoIntegerVar();

    switch(choice)
    {
     case 1: add.usingWhile(firstNo, secondNo);
      break;

     case 2: add.usingFor(firstNo, secondNo);
      break;

     case 3: AddTwoIntegerVar addp = new AddTwoIntegerVar(firstNo, secondNo);
          System.out.print("Addition by using Parameterized Constructor : "+addp.result);
      break;

     case 4: total = AddTwoIntegerVar.usingStaticMethod(firstNo, secondNo);
          System.out.print("Addition by using Static Method : "+total);
      break;
```

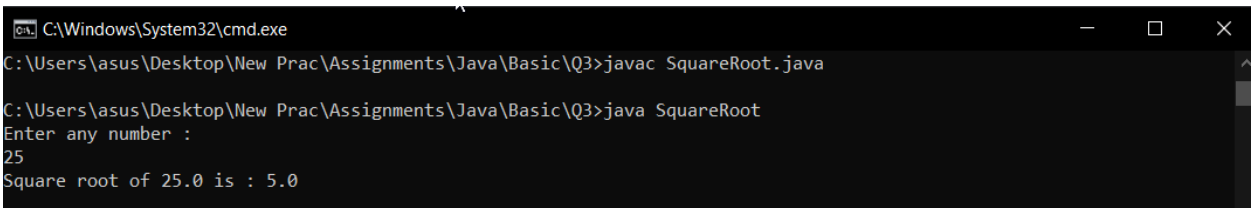
```
        case 5: total = add.usinNonStaticMethod(firstNo, secondNo);
                System.out.print("Addition by using Static Method : "+total);
         break;

         default : System.out.print("Please enter Valid choice :");
       }
    }
}

class AddTwoIntegerVar{
   int result =0;

   AddTwoIntegerVar()
   {

   }

   AddTwoIntegerVar(int num1, int num2)
   {
    result = num1 + num2;
   }

   public void usingWhile(int num1, int num2)
   {
    System.out.println("["+num1+" + "+num2+"]");
    while(num1-- != 0)
    {
     num2++;
    }
    System.out.print("Addition by using While Loop is :"+ num2);
   }

   public void usingFor(int num1, int num2)
   {
    System.out.println("["+num1+" + "+num2+"]");
    for(int i=num1; i!=0 ;i--)
    {
     num2++;
    }
    System.out.print("Addition by using For Loop is :"+ num2);
   }

   static int usingStaticMethod(int x,int y)
```

```
  {
   return x+y;
  }

  public int usinNonStaticMethod(int x,int y)
  {
   result =  x+y;
   return result;
  }


}
```



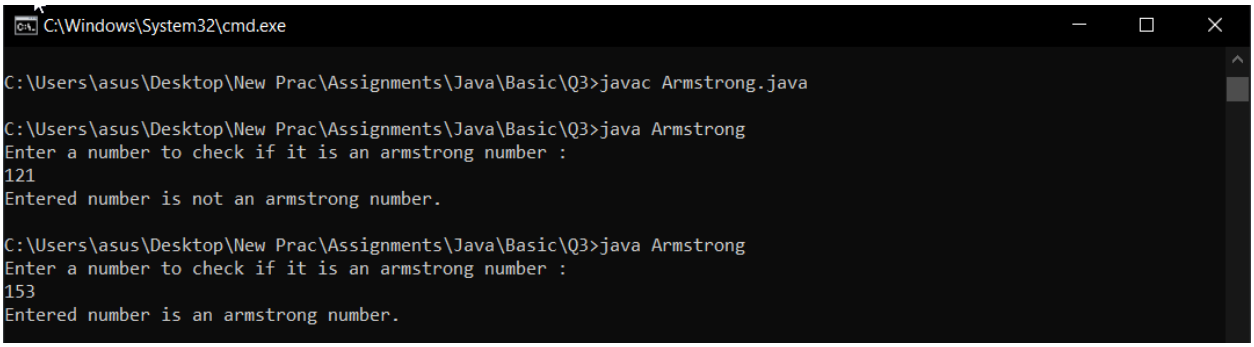
g) Find square root of a number without sqrt method.

```
import java.util.*;

class SquareRoot
{
	public static double squareRoot(double number)
	{
		double temp;
		double sr = number/2;

		do
		{
			temp=sr;
			sr=(temp +(number/temp))/2;
		}
		while((temp-sr)!=0);

		return sr;
	}
	public static void main(String args [])
	{
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter any number : ");

		double num=sc.nextDouble();

		System.out.println("Square root of "+ num +" is : "+squareRoot(num));
	}
}
```



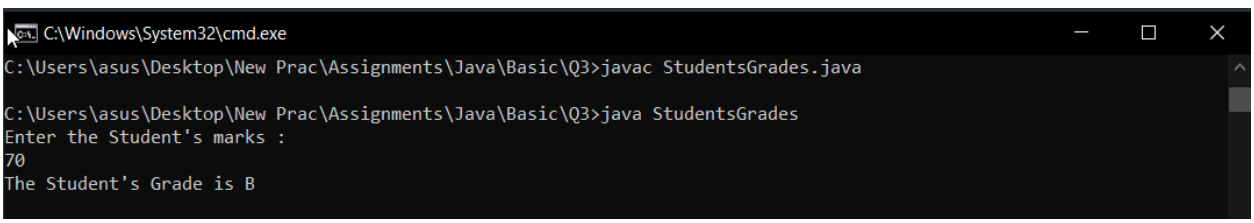
h) Check Armstrong number.

```
import java.util.*;

class Armstrong
{
	public static void main(String args [])
	{
		int n,sum=0,temp,r;
		Scanner sc=new Scanner(System.in);

		System.out.println("Enter a number to check if it is an armstrong number : ");
		n=sc.nextInt();

		temp=n;
		while(temp!=0)
		{
			r=temp%10;
			sum=sum+r*r*r;
			temp=temp/10;
		}
		if(n==sum)
			System.out.println("Entered number is an armstrong number. ");
		else
			System.out.println("Entered number is not an armstrong number.");
	}
}
```



i) Calculate grades of students using their marks.

```
import java.util.*;

class StudentsGrades
{
	public static void main(String args [])
	{
		double marks;
		Scanner sc=new Scanner(System.in);

		System.out.println("Enter the Student's marks : ");
		marks=sc.nextDouble();

		if(marks>=75 && marks<=100)
		{
			System.out.println("The Student's Grade is A ");
		}
		else if(marks>=60 && marks<75)
		{
			System.out.println("The Student's Grade is B ");
		}
		else if(marks>=50 && marks<60)
		{
			System.out.println("The Student's Grade is C ");
		}
		else if(marks>=40 && marks<50)
		{
			System.out.println("The Student's Grade is D ");
		}
		else
		{
			System.out.println("The Student has Failed : ");
		}
	}
}
```

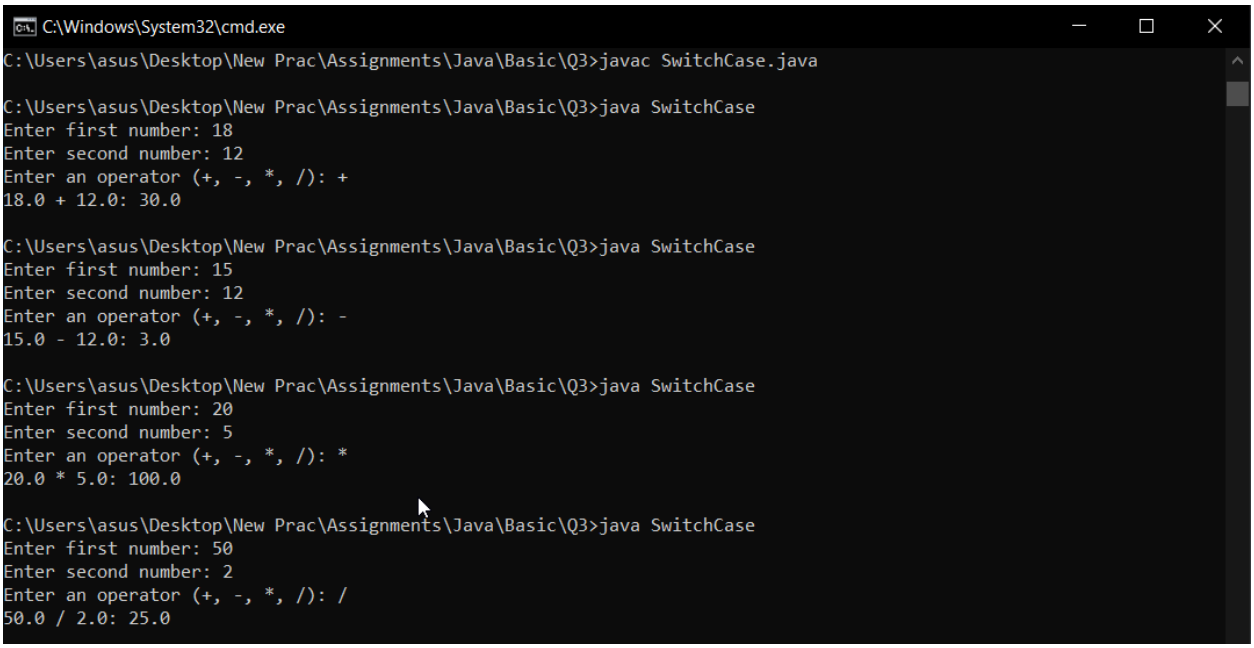


j) Use switch case, recursion, print patterns, etc.

Switch case

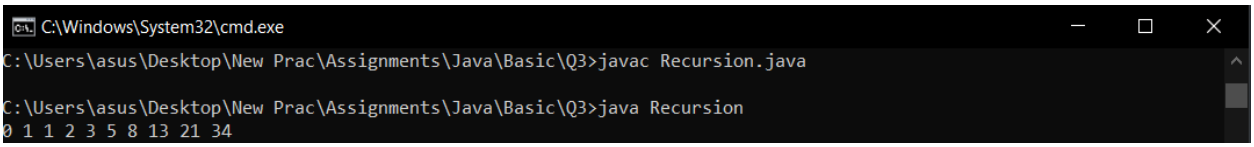
```
import java.util.Scanner;
class SwitchCase
{
   public static void main(String[] args)
         {
      double num1, num2;
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter first number: ");

      num1 = sc.nextDouble();
      System.out.print("Enter second number: ");
      num2 = sc.nextDouble();

      System.out.print("Enter an operator (+, -, *, /): ");
      char operator = sc.next().charAt(0);

      sc.close();
      double output;

      switch (operator)
                  {
         case '+':
            output = num1 + num2;
            break;

         case '-':
            output = num1 - num2;
            break;

         case '*':
            output = num1 * num2;
            break;

         case '/':
            output = num1 / num2;
            break;

         default:
            System.out.printf("You have entered wrong operator");
            return;
```

```
        }

        System.out.println(num1 + " " + operator + " " + num2 + ": " + output);
    }
}
```



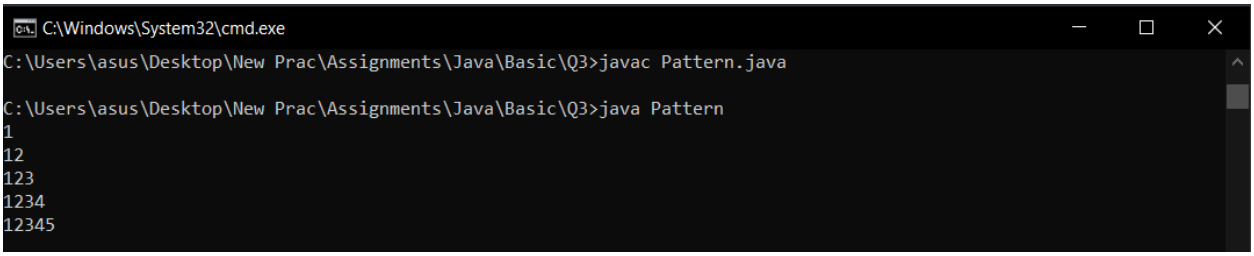
Recursion

```
public class Recursion //fibonacci series using recursion
{
        static int num1=0,num2=1,num3=0;
        static void fibonacci(int n)
        {
        if(n>0){
        num3 = num1 + num2;
        num1 = num2;
        num2 = num3;
        System.out.print(" "+num3);
        fibonacci(n-1);
        }
}

        public static void main(String args[])
        {
        int n=10;
        System.out.print(num1+" "+num2);
        fibonacci(n-2);
        }
}
```



Pattern

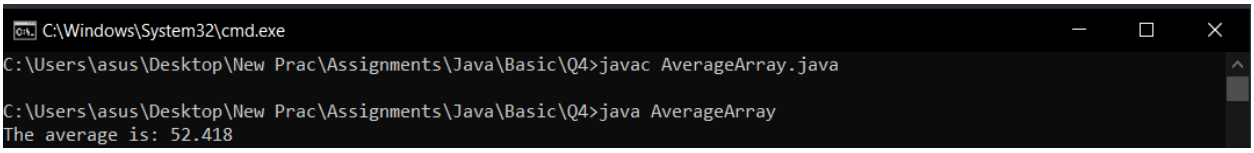
```
class Pattern
{

   public static void main(String arg[])
   {

      for (int i = 1; i <= 5; i++)
      {

         for (int j = 1; j <= i; j++)
         {

            System.out.print(j);

         }

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



Q4)

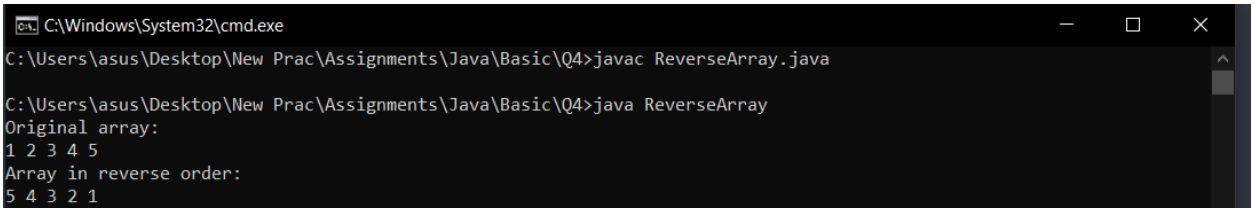
a) Calculate average of numbers using Array.

```
public class AverageArray
{

   public static void main(String[] args)
         {

                  double[] arr = {19, 12.89, 16.5, 200, 13.7};

                  double total = 0;


                  for(int i=0; i<arr.length; i++)
                  {

                  total = total + arr[i];

      }

      double average = total / arr.length;

      System.out.format("The average is: %.3f", average);

   }

}
```



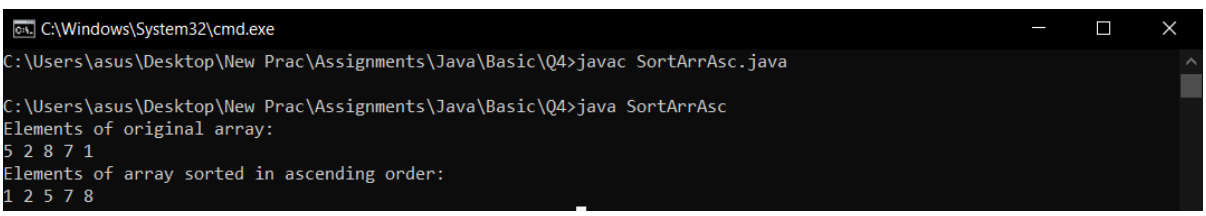
b) Reverse an array.

```
class ReverseArray
{
   public static void main(String[] args)
         {
                  int [] arr = new int [] {1, 2, 3, 4, 5};

                  System.out.println("Original array: ");

                  for (int i = 0; i < arr.length; i++)
                  {

                  System.out.print(arr[i] + " ");

                  }

      System.out.println();

      System.out.println("Array in reverse order: ");

      for (int i = arr.length-1; i >= 0; i--)
                  {

         System.out.print(arr[i] + " ");

      }

   }

}
```



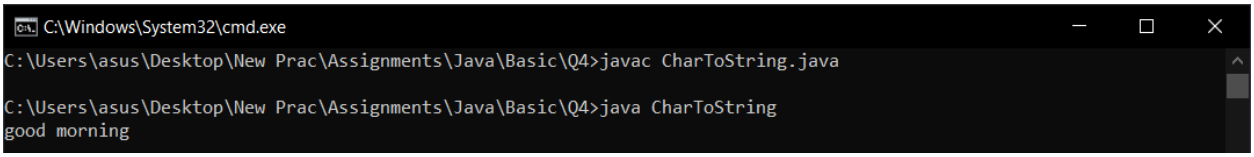
c) Sort an array in ascending order.

```
class SortArrAsc
{
   public static void main(String[] args)
         {
      int [] arr = new int [] {5, 2, 8, 7, 1};
      int temp = 0;

                  System.out.println("Elements of original array: ");

                  for (int i = 0; i < arr.length; i++)
                  {
         System.out.print(arr[i] + " ");
      }

      for (int i = 0; i < arr.length; i++)
                  {
         for (int j = i+1; j < arr.length; j++)
                        {
           if(arr[i] > arr[j])
                                    {
              temp = arr[i];
              arr[i] = arr[j];
              arr[j] = temp;
           }
         }
      }

      System.out.println();

      System.out.println("Elements of array sorted in ascending order: ");
      for (int i = 0; i < arr.length; i++) {
         System.out.print(arr[i] + " ");
      }
   }
}
```



d) Convert char Array to String.

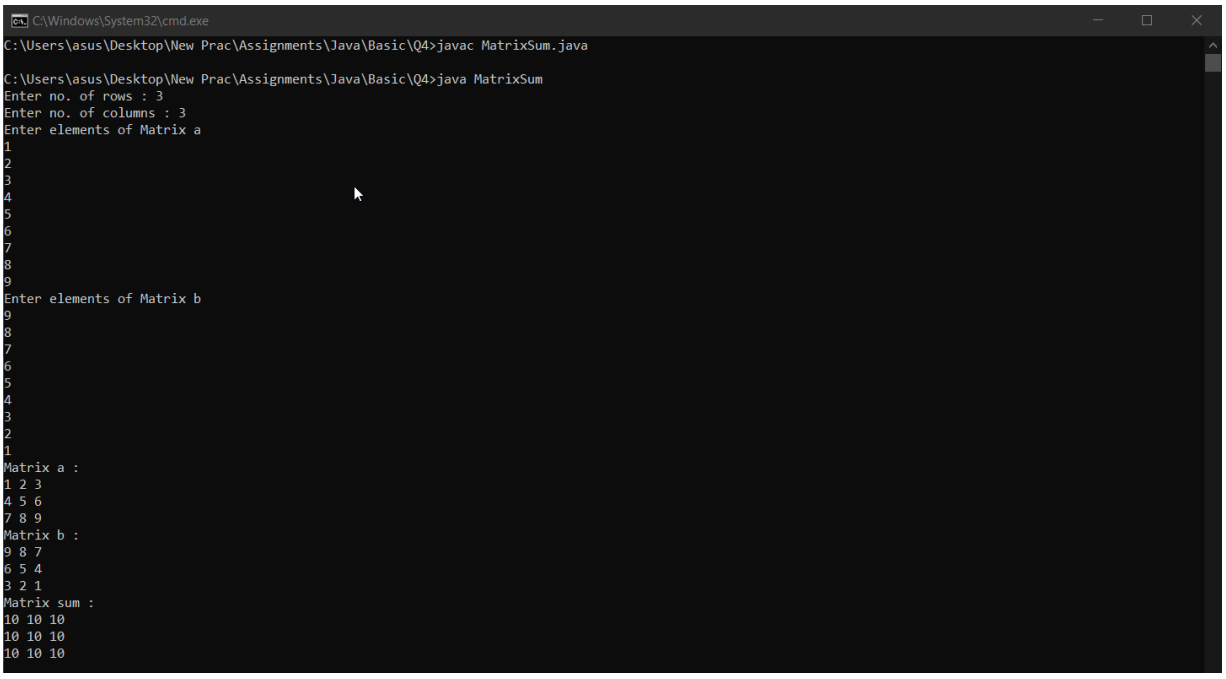
```
class CharToString
{
   public static String toString(char[] a)
   {
      String string = new String(a);
      return string;
   }

         public static void main(String args[])
   {
      char s[] = { 'g', 'o', 'o', 'd',' ',
                   'm', 'o', 'r', 'n',
                   'i', 'n', 'g' };

      System.out.println(toString(s));
   }
}
```



e) Add two Matrix using Multi-dimensional Arrays.

```
import java.util.Scanner;
class MatrixSum
{
	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter no. of rows : ");
		int m = sc.nextInt();
		System.out.print("Enter no. of columns : ");
		int n = sc.nextInt();

		int[][] a = new int[m][n];
		int[][] b = new int[m][n];
		int[][] sum = new int[m][n];
		System.out.println("Enter elements of Matrix a ");
		for(int i=0;i<m;i++)
		{
			for(int j=0;j<n;j++)
			{
				a[i][j] = sc.nextInt();
			}
		}
		System.out.println("Enter elements of Matrix b ");
		for(int i=0;i<m;i++)
		{
			for(int j=0;j<n;j++)
			{
				b[i][j] = sc.nextInt();
			}

		}
		System.out.println("Matrix a :");
		for(int i=0;i<m;i++)
		{
			for(int j=0;j<n;j++)
			{
				System.out.print(a[i][j]+" ");
			}
			System.out.println();
		}
		System.out.println("Matrix b :");
		for(int i=0;i<m;i++)
```

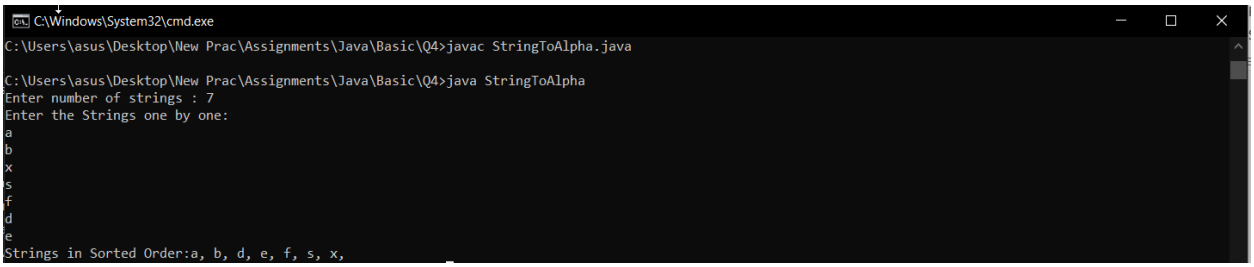
```
            {
                for(int j=0;j<n;j++)
                {
                    System.out.print(b[i][j]+" ");
                }
                System.out.println();
            }
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<n;j++)
                {
                    sum[i][j] = a[i][j]+b[i][j];
                }
            }
            System.out.println("Matrix sum :");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<n;j++)
                {
                    System.out.print(sum[i][j]+" ");
                }
                System.out.println();
            }
        }
}
```



f) Sort strings in alphabetical order.

```
import java.util.Scanner;
class StringToAlpha
{
   public static void main(String[] args)
   {
      int count;
      String temp;
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of strings : ");
      count = sc.nextInt();


      String str[] = new String[count];
      Scanner sc2 = new Scanner(System.in);

      System.out.println("Enter the Strings one by one:");
      for(int i = 0; i < count; i++)
      {
         str[i] = sc2.nextLine();
      }
      sc.close();
      sc2.close();

      for (int i = 0; i < count; i++)
      {
         for (int j = i + 1; j < count; j++) {
            if (str[i].compareTo(str[j])>0)
            {
               temp = str[i];
               str[i] = str[j];
               str[j] = temp;
            }
         }
      }

      System.out.print("Strings in Sorted Order:");
      for (int i = 0; i <= count - 1; i++)
      {
         System.out.print(str[i] + ", ");
      }
   }
```

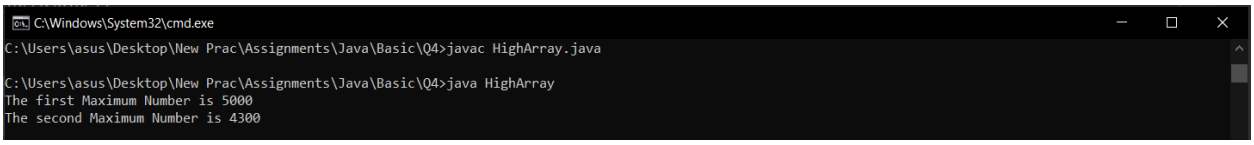
}


C:\Windows\System32\cmd.exe
C:\Users\asus\Desktop\New Prac\Assignments\Java\Basic\Q4>javac StringToAlpha.java
C:\Users\asus\Desktop\New Prac\Assignments\Java\Basic\Q4>java StringToAlpha
Enter number of strings : 7
Enter the Strings one by one:
a
b
x
s
f
d
e
Strings in Sorted Order:a, b, d, e, f, s, x,

g) Find out the highest and second highest numbers in an array.

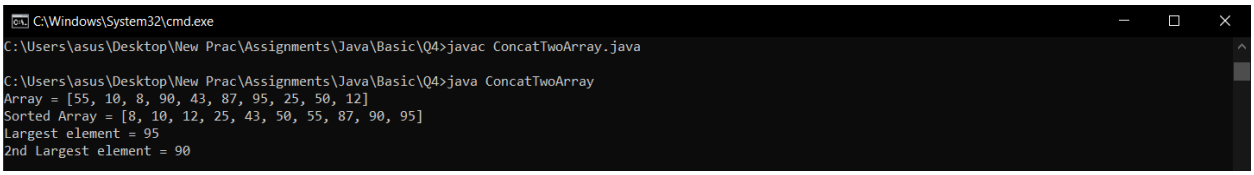
```
class HighArray
{
	public void MaximumNums(int[] num1)
	{
		int max1 = 0, max2 = 0;

		for(int x:num1)
		{
		if(max1 < x)
		{
		max2 = max1;
		max1 = x;
		}

		else if(max2 < x)
		{
		max2 = x;
		}
		}

		System.out.println("The first Maximum Number is " + max1);
		System.out.println("The second Maximum Number is " + max2);
	}

 public static void main(String args[])
{
	int num2[] = {4300,78,9,34,100,5000,432,678,12,7,3,0,654};
```

```
        HighArray sc = new HighArray();
        sc.MaximumNums(num2);
 }

}
```



h) Concatenate two arrays.

```
import java.util.*;
class ConcatTwoArray
{
  public static void main(String []args)
        {
    int arr[] = {55, 10, 8, 90, 43, 87, 95, 25, 50, 12};

        System.out.println("Array = "+Arrays.toString(arr));

    Arrays.sort(arr);

    System.out.println("Sorted Array = "+Arrays.toString(arr));

    System.out.println("Largest element = "+arr[9]);

    System.out.println("2nd Largest element = "+arr[8]);
  }
}
```

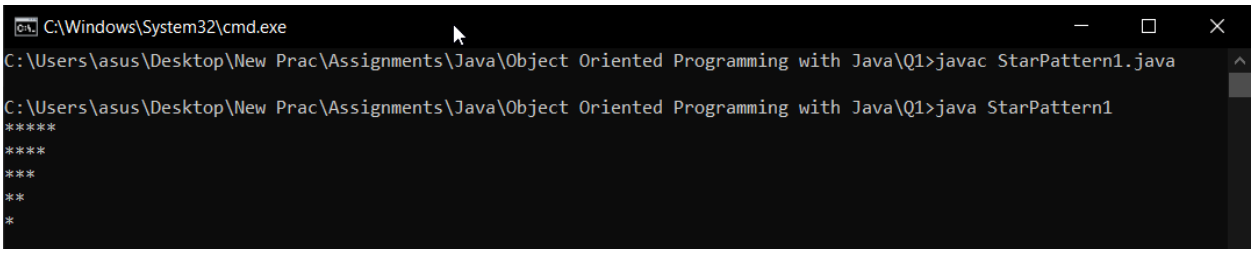


Object Oriented Programming with Java Assignment_2

Q1) Print different patterns of asterisk (*) using loops (e.g. triangle of *).

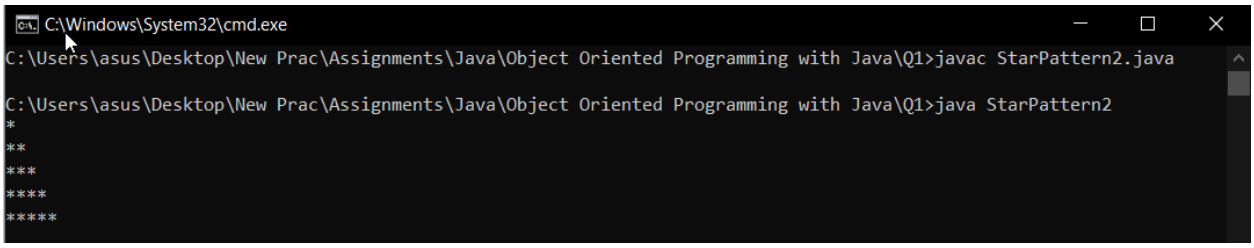
Pattern 1

```
class StarPattern1
{
	public static void main(String args [])
	{
		int i,j;

		for(i=1; i<=5; i++)
		{
			for(j=i; j<=5; j++)
			{
				System.out.print("*");
			}
			System.out.println();
		}
	}
}
```



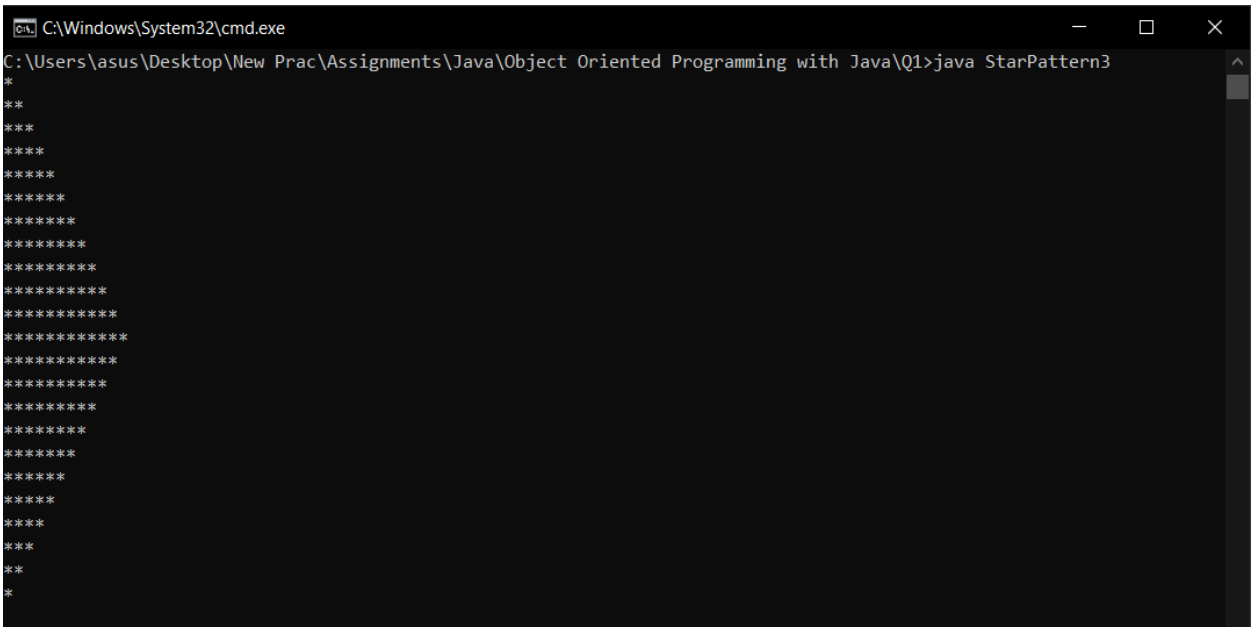
Pattern 2

```
class StarPattern2
{
	public static void main(String args [])
	{
		int i,j;
		for(i=1; i<=5; i++)
		{
			for(j=1; j<=i; j++)
			{
				System.out.print("*");
			}
			System.out.println();
		}
	}
}
```



Pattern 3

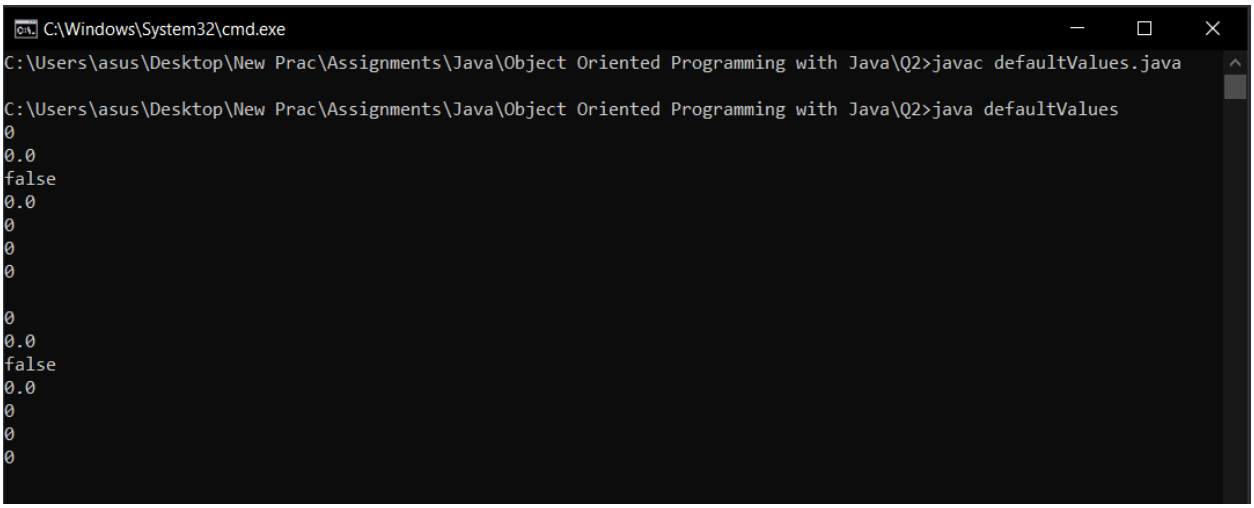
```
class StarPattern3
{
	public static void main(String args[])
	{
		int i,j;
		for(i=1; i<=12; i++)
		{
			for(j=1; j<=i; j++)
			{
				System.out.print("*");
			}
			System.out.println();
		}
		for(i=1; i<=12; i++)
		{
			for(j=i; j<=11; j++)
			{
				System.out.print("*");
			}
			System.out.println();
		}
	}
}
```



Q2) Print default values of static & instance variables for different data types.

```
public class defaultValues
{
	int a;
	float b;
	boolean c;
	double d;
	byte e;
	long f;
	short g;
	char h;

	static int aa;
	static float bb;
	static boolean cc;
	static double dd;
	static byte ee;
	static long ff;
	static short gg;
	static char hh;

	public static void main(String args [])
	{
		System.out.println(defaultValues.aa);
		System.out.println(defaultValues.bb);
		System.out.println(defaultValues.cc);
		System.out.println(defaultValues.dd);
		System.out.println(defaultValues.ee);
		System.out.println(defaultValues.ff);
		System.out.println(defaultValues.gg);
		System.out.println(defaultValues.hh);

		defaultValues sc=new defaultValues();

		System.out.println(sc.a);
		System.out.println(sc.b);
		System.out.println(sc.c);
		System.out.println(sc.d);
		System.out.println(sc.e);
		System.out.println(sc.f);
		System.out.println(sc.g);
		System.out.println(sc.h);
```

```
        }

}
```


C:\Windows\System32\cmd.exe
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q2>javac defaultValues.java
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q2>java defaultValues
0
0.0
false
0.0
0
0
0
0
0.0
false
0.0
0
0
0

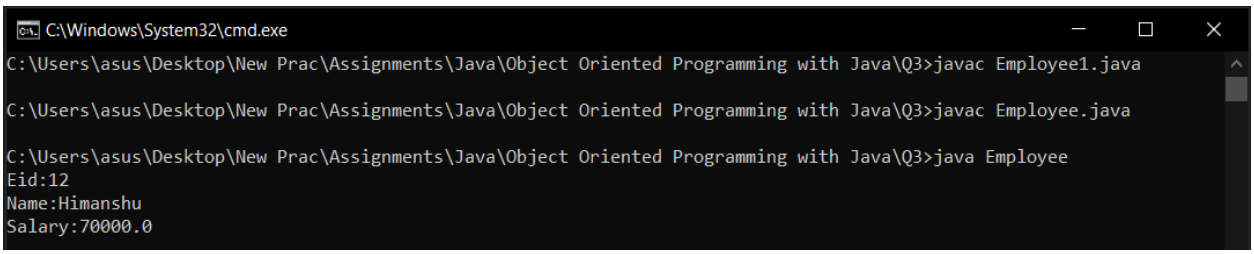
Q3) Build a class Employee which contains details about the employee and compile and run its instance.

Employee1.java

```
import java.util.*;


public class Employee1
{
        private int eid;
        private String ename;
        private double salary;


        public void setEmployee1(int eid,String ename,double salary)
        {
                this.eid=eid;
                this.ename=ename;
                this.salary=salary;
        }
        public void getEmployee1()
        {
                System.out.println("Eid:"+eid);
                System.out.println("Name:"+ename);
                System.out.println("Salary:"+salary);
   }


}
```

Employee.java

```
public class Employee
{
  public static void main(String args[])
  {
        Employee1 e= new Employee1();
        e.setEmployee1(12,"Himanshu",70000);
        e.getEmployee1();
  }
}
```



Q4) Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.

Bank.java

```
public class Bank
{
        private String bank_name;
        private int bank_ifsc_code;
        private String branch_location;
        private Customer customer;

        public Bank()
        {

        }

        public Bank(String bname, int ifsc, String bLoc, Customer cust)
        {
          bank_name=bname;
          bank_ifsc_code=ifsc;
          branch_location=bLoc;
          customer = cust;
        }

   @Override
        public String toString()
        {
```

```
                return "\n Bank name : "+bank_name+"\n Bank ifsc code : "+bank_ifsc_code+"\n
Branch Location : "+branch_location+"\n "+customer;
        }


}
```

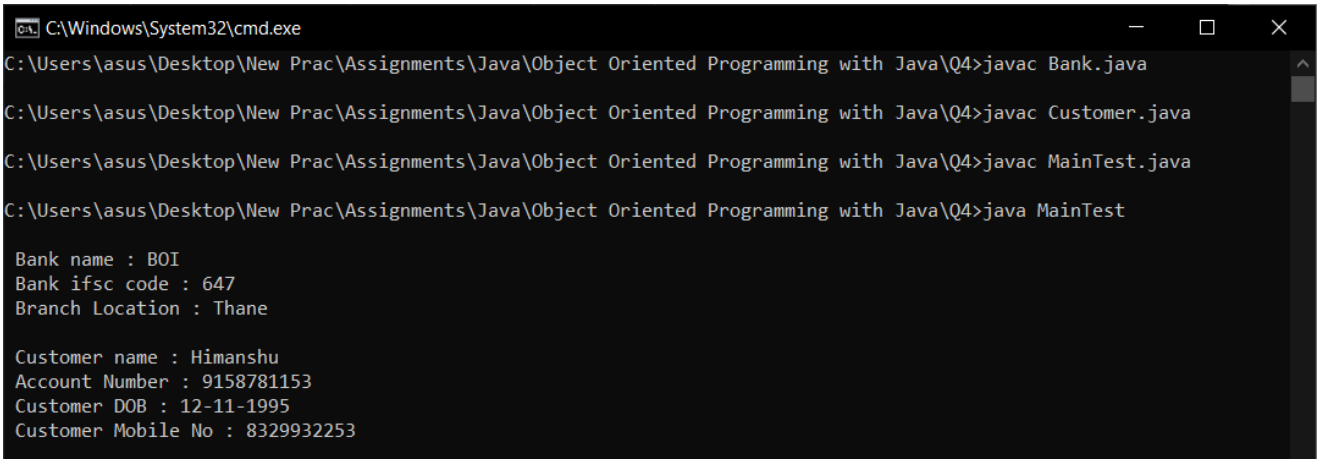
Customer.java

```
public class Customer
{
        private String cust_name;
        private long cust_acc_no;
        private String cust_dob;
        private long cust_mobile_no;

        public Customer(String name, long accNo, String dob, long moNo)
        {
         cust_name=name;
         cust_acc_no=accNo;
         cust_dob=dob;
         cust_mobile_no=moNo;
        }

   @Override
   public String toString()
   {
        return "\n Customer name : "+cust_name+"\n Account Number : "+cust_acc_no+"\n
Customer DOB : "+cust_dob+
```

```
                "\n Customer Mobile No : "+cust_mobile_no;
    }


}
```

MainTest.java

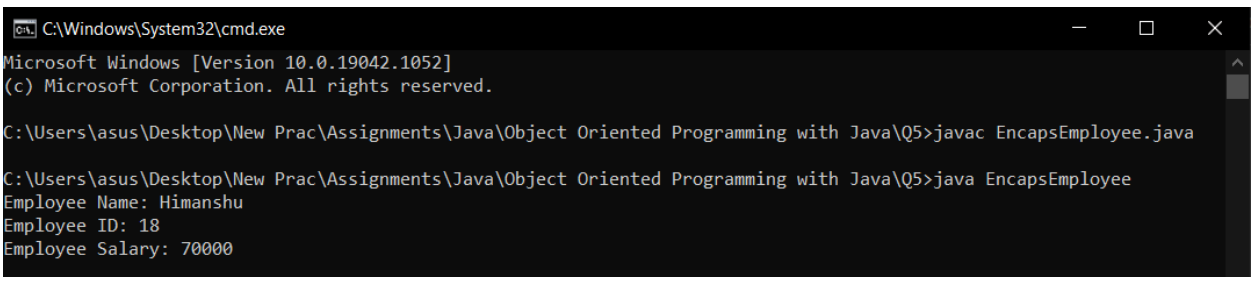
```
public class MainTest
{

        public static void main(String[] args)
        {
    Customer c2 = new Customer("Himanshu",9158781153L,"12-11-1995",8329932253L);
    Bank b = new Bank("BOI",647,"Thane",c2);
    System.out.println(b);
        }
}
```



Q5) Create a class Employee and encapsulate the data members.

```
class EncapsulationDemo
{
   private int eid;
   private String empName;
   private int salary;



   public void setEmpSalary(int newValue)
           {
      salary = newValue;
   }

   public void setEmpName(String newValue)
           {
      empName = newValue;
   }

   public void setEmpID(int newValue)
           {
      eid = newValue;
   }
           public int getEmpID()
           {
      return eid;
   }

   public String getEmpName()
```

```
        {
      return empName;
   }
        public int getEmpSalary()
        {
                return salary;
        }
}
public class EncapsEmployee
{
   public static void main(String args[])
        {
      EncapsulationDemo obj = new EncapsulationDemo();
      obj.setEmpName("Himanshu");
      obj.setEmpSalary(70000);
      obj.setEmpID(18);
      System.out.println("Employee Name: " + obj.getEmpName());
      System.out.println("Employee ID: " + obj.getEmpID());
      System.out.println("Employee Salary: " + obj.getEmpSalary());
   }
}
```

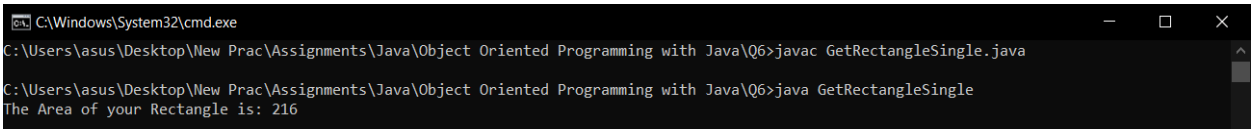


Q6) Create demo applications to illustrate different types of inheritance.

GetRectangleSingle.java

```
class SingleInhArea
{
int length;
int breadth;
}

public class GetRectangleSingle extends SingleInhArea{

int area;

public void TotalArea()
{
area = length*breadth;
}

public static void main(String args[])
{
GetRectangleSingle gr = new GetRectangleSingle();

gr.length = 18;
gr.breadth = 12;

gr.TotalArea();
System.out.println("The Area of your Rectangle is: " + gr.area);
}
```

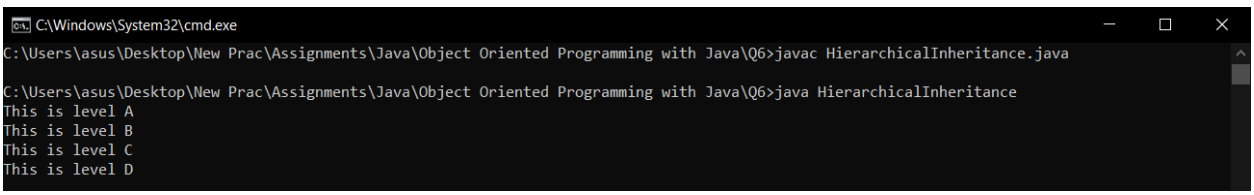
```
}
```


C:\Windows\System32\cmd.exe
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q6>javac GetRectangleSingle.java
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q6>java GetRectangleSingle
The Area of your Rectangle is: 216

HierarchicalInheritance.java

```
class LevelA
 {
 public void showA()
 {
 System.out.println("This is level A");
 }
}

class LevelB extends LevelA
 {
 public void showB()
 {
System.out.println("This is level B");
 }
 }

 class LevelC extends LevelA
 {
 public void showC()
 {
 System.out.println("This is level C");
 }
 }

class LevelD extends LevelA
{
public void showD()
```

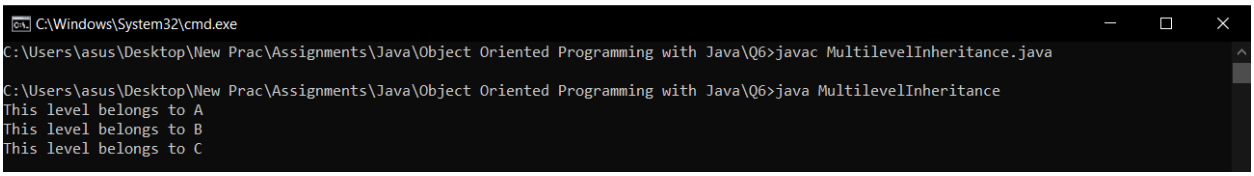
```
{
System.out.println("This is level D");
}
}

public class HierarchicalInheritance {

public static void main(String args[])
{
LevelB lb = new LevelB();
lb.showA();
lb.showB();

LevelC lc = new LevelC();
lc.showC();

LevelD ld = new LevelD();
ld.showD();

}
}
```



MultilevelInheritance.java

```
class LevelA
 {
 public void showA()
 {
 System.out.println("This level belongs to A");
 }
 }

 class LevelB extends LevelA
 {
public void showB()
 {
 System.out.println("This level belongs to B");
}
}

class LevelC extends LevelB
{
 public void showC()
 {
 System.out.println("This level belongs to C");
}
 }

 public class MultilevelInheritance {

 public static void main(String args[])
```

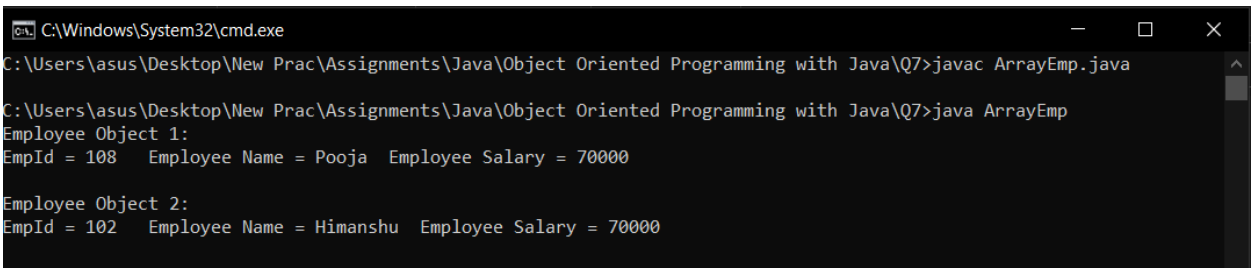
```
{
LevelC lc = new LevelC();

lc.showA();
lc.showB();
lc.showC();
}
}
```



Q7) Create an Array of Employee class and initialize array elements with different employee objects.

```
class ArrayEmp
{
  public static void main(String args[]){

   Employee[] obj = new Employee[2] ;


    obj[0] = new Employee(108,"Pooja",70000);
    obj[1] = new Employee(102,"Himanshu",70000);


    System.out.println("Employee Object 1:");
    obj[0].showData();
          System.out.println();
    System.out.println("Employee Object 2:");
    obj[1].showData();
          System.out.println();


 }
}

class Employee
{
 int empId;
```

```
 String name;
 int salary;
Employee(int eid, String n, int s)
{
   empId = eid;
   name = n;
         salary=s;
 }
public void showData(){
  System.out.print("EmpId = "+empId + "  " + " Employee Name = "+name + "  "+ "Employee Salary = "+salary);
  System.out.println();
 }
}
```



Q8) Create a demo application to understand the role of access modifiers.

Laptop.java

```
public class Laptop
{
	private int modelNo;
	float price;
	protected long registerNo;
	public String companyName;
	protected String modelName;

	public Laptop()
	{

	}

	public Laptop(int modelNo, float price, long registerNo, String companyName, String modelName)
	{
		this.modelNo = modelNo;
		this.price = price;
		this.registerNo = registerNo;
		this.companyName = companyName;
		this.modelName = modelName;
	}

	public Laptop(float price, long registerNo, String companyName, String modelName)
```

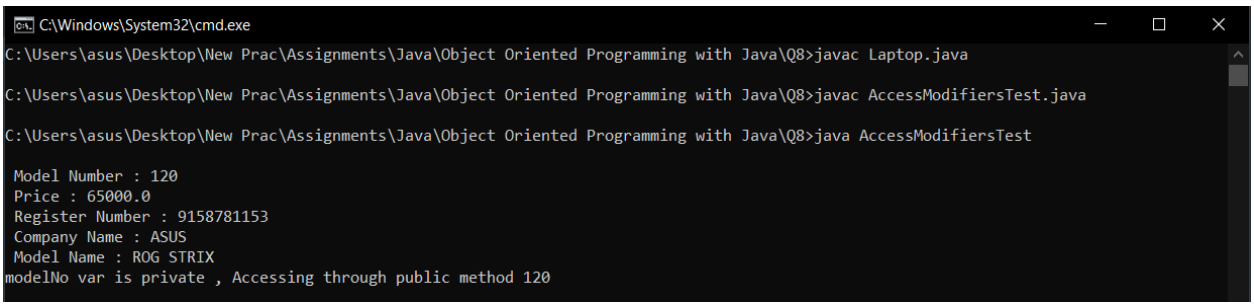
```
	{
		this.price = price;
		this.registerNo = registerNo;
		this.companyName = companyName;
		this.modelName = modelName;
	}

	public void setModelNo(int modelNo)
	{
		this.modelNo = modelNo;
	}

protected void setPrice(float price)
	{
		this.price = price;
	}

	void RegisterNo(long registerNo)
	{
		this.registerNo = registerNo;
	}

	private void setCompanyName(String companyName)
	{
		this.companyName = companyName;
	}
```

```
    protected void setModelName(String modelName)
    {
        this.modelName = modelName;
    }

    public int getModelNo()
    {
        return modelNo;
    }

protected float getPrice()
    {
        return price;
    }

    public long getRegisterNo()
    {
        return registerNo;
    }

    private String getCompanyName()
    {
        return companyName;
    }

    protected String getModelName()
    {
```

```
		return modelName;
	}

	public String toString(){
		return "\n Model Number : "+modelNo+"\n Price : "+price+"\n Register Number
: "+registerNo+"\n Company Name : "+companyName+"\n Model Name : "+modelName;
	}

}
```

AccessModifiersTest.java

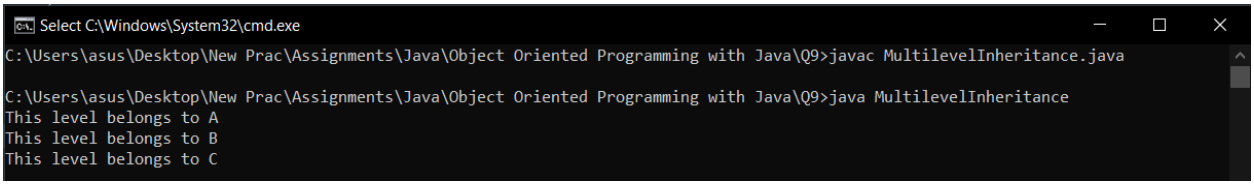
```
public class AccessModifiersTest
{

        public static void main(String[] args)
        {
    Laptop lap = new Laptop(120,65000.0f,9158781153L,"ASUS","ROG STRIX");


    System.out.println(lap);


    System.out.println("modelNo var is private , Accessing through public method
"+lap.getModelNo());


        }
}
```



Q9) Implement multilevel inheritance using different packages.

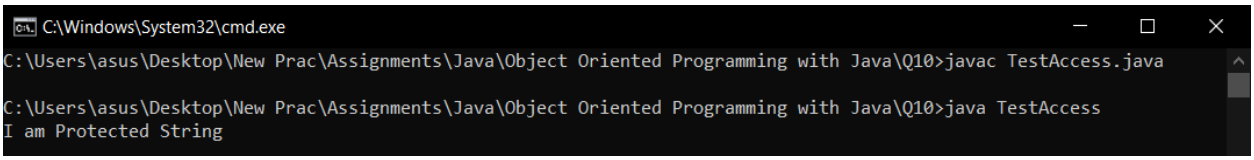
```
class LevelA
{
	public void showA()
	{
		System.out.println("This level belongs to A");
	}
}

class LevelB extends LevelA
{
	public void showB()
	{
		System.out.println("This level belongs to B");
	}
}

class LevelC extends LevelB
{
	public void showC()
	{
		System.out.println("This level belongs to C");
	}
}

public class MultilevelInheritance
```

```
{
	public static void main(String args[])
	{
		LevelC lc = new LevelC();

		lc.showA();
		lc.showB();
		lc.showC();
	}
}
```



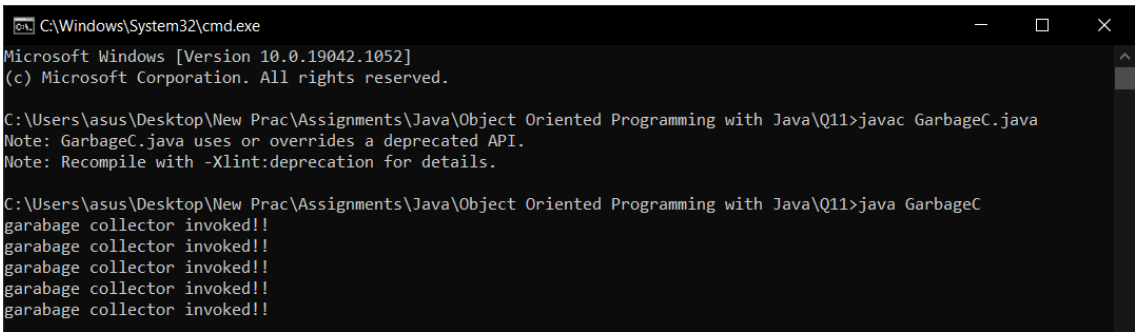
Q10) Access/invoke protected members/methods of a class outside the package.

```
class Access
{
	protected String str="I am Protected String";
	public String getStr()
	{
		return str;
	}

// TestAccess.java
}


public class TestAccess extends Access
{
	public static void main(String[] args)
	{
		Access a=new Access();
		System.out.println(a.getStr());
	}
}
```



Q11) Override finalize method to understand the behavior of JVM garbage collector.

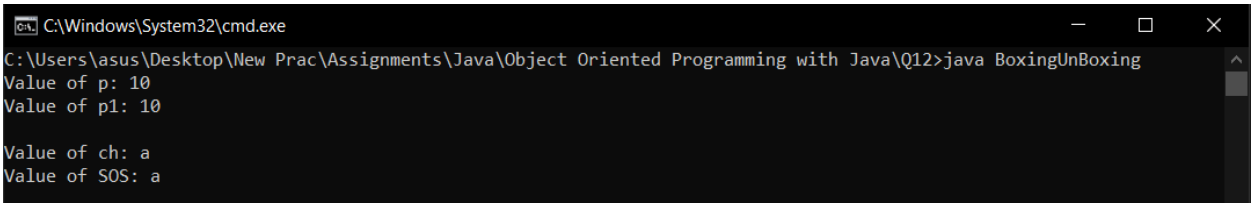
```
public class GarbageC
{
	public void finalize()
	{
		System.out.println("garabage collector invoked!!");
	}

public static void main(String[] args)
	{
		GarbageC [] gc=new GarbageC[5];

		for(int i=0;i<5;i++)
		{
			gc[i]=new GarbageC();
		}

		gc=null;
		System.gc();
	}
}
```



Q12) Create sample classes to understand boxing & unboxing.

```
import java.io.*;
class Boxing
{
        Integer p =new Integer(10);

                int p1 = p;

                public void dispBox()
                {

                System.out.println("Value of p: " + p);
                System.out.println("Value of p1: " + p1);
            }
}

class UnBoxing
{

        Character SOS = 'a';
        char ch = SOS;

                  public void dispUnBox()
                  {
                    System.out.println("Value of ch: " + ch);
                    System.out.println("Value of SOS: " + SOS);
```

```
        }
}
   public class BoxingUnBoxing
        {
          public static void main (String[] args)
        {
          Boxing b= new Boxing ();
          UnBoxing ub=new UnBoxing();
          b.dispBox();
          System.out.println();
          ub.dispUnBox();

        }
}
```



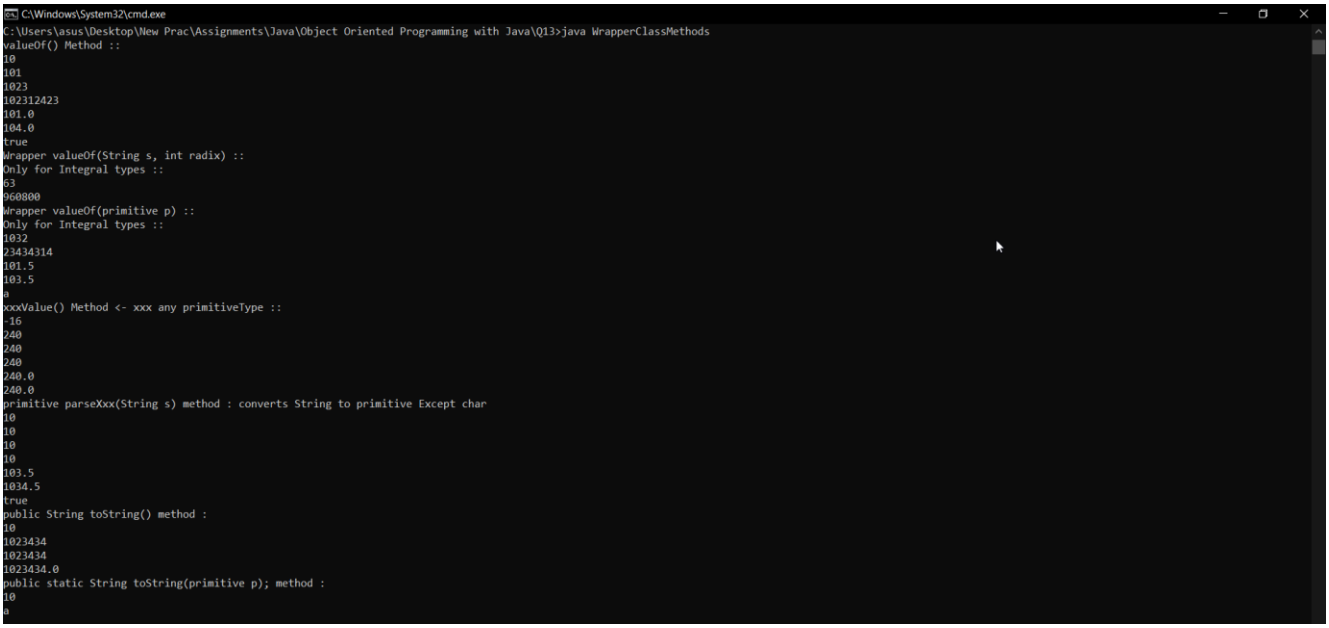
Q13) Use different methods of java defined wrapper classes.

```
class WrapperClassMethods
{
	public static void main(String[] args)
	{
		System.out.println("valueOf() Method ::");
		Byte B = Byte.valueOf("10");
	System.out.println(B);
	Short S = Short.valueOf("101");
	System.out.println(S);
	Integer I = Integer.valueOf("1023");
	System.out.println(I);
		Long L = Long.valueOf("102312423");
	System.out.println(L);
	Float F = Float.valueOf("101.0");
	System.out.println(F);
	Double D = Double.valueOf("104.0");
	System.out.println(D);
	Boolean BO = Boolean.valueOf("true");
	System.out.println(BO);

	System.out.println("Wrapper valueOf(String s, int radix) ::");
	System.out.println("Only for Integral types ::");

	Integer I1 = Integer.valueOf("111111", 2);
	System.out.println(I1);
```

```
Long L1 = Long.valueOf("11111111", 7);
System.out.println(L1);

System.out.println("Wrapper valueOf(primitive p) ::");
System.out.println("Only for Integral types ::");

Integer I2 = Integer.valueOf(1032);
System.out.println(I2);
Long L2 = Long.valueOf(23434314L);
System.out.println(L2);
Float F2 = Float.valueOf(101.5f);
System.out.println(F2);
Double D2 = Double.valueOf(103.5d);
System.out.println(D2);
Character C = Character.valueOf('a');
System.out.println(C);

System.out.println("xxxValue() Method <- xxx any primitiveType ::");
Integer I3 = new Integer(240);
System.out.println(I3.byteValue());
System.out.println(I3.shortValue());
System.out.println(I3.intValue());
System.out.println(I3.longValue());
System.out.println(I3.floatValue());
System.out.println(I3.doubleValue());
```

```
System.out.println("primitive parseXxx(String s) method : converts String to primitive Except char");

byte bb = Byte.parseByte("10");

System.out.println(bb);

short ss = Short.parseShort("10");

System.out.println(ss);

int ii = Integer.parseInt("10");

System.out.println(ii);

long ll = Long.parseLong("10");

System.out.println(ll);

float ff = Float.parseFloat("103.5");

System.out.println(ff);

double dd = Double.parseDouble("1034.5");

System.out.println(dd);

boolean bb1 = Boolean.parseBoolean("true");

System.out.println(bb1);


System.out.println("public String toString() method : ");


Integer I12 = new Integer(10);

String str2 = I12.toString();

System.out.println(str2);


Long l12 = new Long(1023434L);

String str3 = l12.toString();

System.out.println(str3);
```

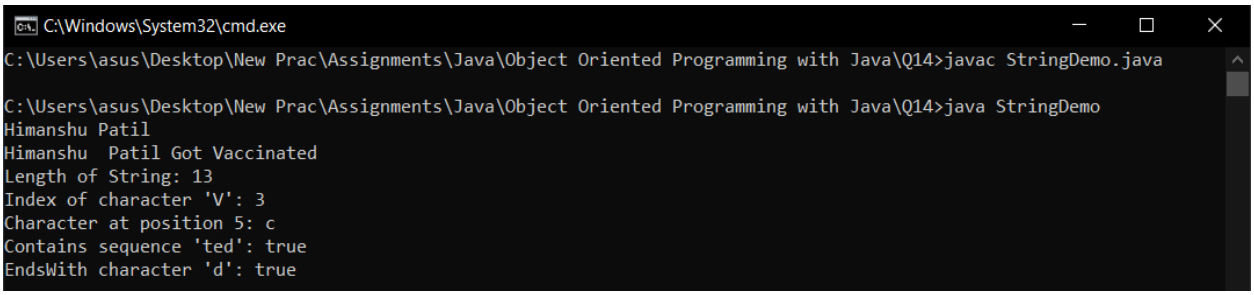
```
        Float f12 = new Float(10234.0);
        String str4 = l12.toString();
        System.out.println(str4);

        Double d12 = new Double(1023434.00);
        String str5 = d12.toString();
        System.out.println(str5);

        System.out.println("public static String toString(primitive p); method : ");

        String s = Integer.toString(10);
        System.out.println(s);
        String s1 = Character.toString('a');
        System.out.println(s1);

    }
}
```



Q14) Create StringDemo class and perform different string manipulation methods.

```
public class StringDemo
{
  public static void main(String[] args)
  {

String str1 = "Himanshu ";
String str2 = "Patil";
String str3 = "GotVaccinated";
String str4 ="Got";
String str5= "Vaccinated";

String str6= str1.concat(str2);
System.out.println(str6);



String str7= str1+" "+str2+" "+str4+" " +str5;
System.out.println(str7);



System.out.println("Length of String: " + str3.length());
System.out.println("Index of character 'V': " + str3.indexOf('V'));
System.out.println("Character at position 5: " + str3.charAt(5));
System.out.println("Contains sequence 'ted': " + str3.contains("ted"));
System.out.println("EndsWith character 'd': " + str3.endsWith("d"));
  }
```

}


C:\Windows\System32\cmd.exe
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q14>javac StringDemo.java
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q14>java StringDemo
Himanshu Patil
Himanshu Patil Got Vaccinated
Length of String: 13
Index of character 'V': 3
Character at position 5: c
Contains sequence 'ted': true
EndsWith character 'd': true

Q15) Create user defined checked and unchecked exceptions.

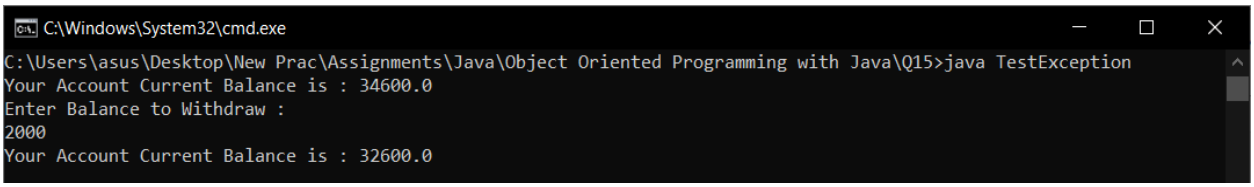
TestException.java

```
import java.util.*;
public class TestException
{
        static double amount = 34600.0;


        public static void withdrawAmount(double withdrawAmount) throws
InsufficientBalanceException
        {
      if(amount < withdrawAmount){
                    throw new InsufficientBalanceException("Withdraw amount :
"+withdrawAmount+" greater than Current Balance : "+amount);
        }
      else{
        amount = amount - withdrawAmount;
        System.out.println("Your Account Current Balance is : "+amount);
    }
        }


        public static void main(String[] args)
        {
    Scanner sc = new Scanner(System.in);
    System.out.println("Your Account Current Balance is : "+amount);
    System.out.println("Enter Balance to Withdraw : ");
    double withdrawAmount = sc.nextDouble();
```

```
    try
          {
        withdrawAmount(withdrawAmount);
    }
    catch(Exception e)
          {
        e.printStackTrace();
    }


        }
}
```

InsufficientBalanceException.java

```
public class InsufficientBalanceException extends Exception
{
        public InsufficientBalanceException(String s)
        {
    super(s);
        }
}
```



Q16) Create a Demo class to Read & write image/text files.

```
import java.io.*;
class RWDemo
{
	public static void main(String args[])throws IOException
	{
		FileReader fr =new FileReader("abc.txt");

		FileWriter fw=new FileWriter("rw.txt");
		int c;
		while((c=fr.read())!=-1)
		{
			//fw.write((byte)c);
			//System.out.print((char)c);
			fw.write(c);//write the char that are read
			System.out.print((char)c);//print on the console

		}
		fr.close();
		fw.close();

	}
}
```

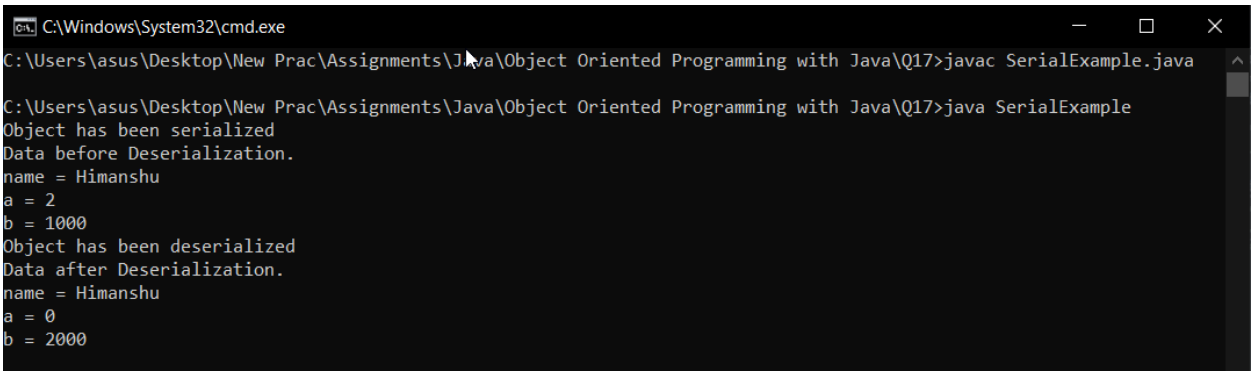
Q17) Create SerializationDemo class to illustrate serialization and de-serialization process.

```
import java.io.*;

class Emp implements Serializable
{
private static final long serialversionUID =
                                                        129348938L;
        transient int a;
        static int b;
        String name;




public Emp(String name, int a, int b)
        {
                this.name = name;
                this.a = a;
                this.b = b;
        }

}

public class SerialExample
{
public static void printdata(Emp object1)
```

```
        {

                System.out.println("name = " + object1.name);
                System.out.println("a = " + object1.a);
                System.out.println("b = " + object1.b);
        }

public static void main(String[] args)
        {
                Emp object = new Emp("Himanshu", 2, 1000);
                String filename = "himanshu.txt";

                try {


                        FileOutputStream file = new FileOutputStream
                                                                (filename);
                        ObjectOutputStream out = new ObjectOutputStream
                                                                (file);



                        out.writeObject(object);


                        out.close();
                        file.close();

                        System.out.println("Object has been serialized\n"
```

```
                                        + "Data before Deserialization.");
            printdata(object);

object.b = 2000;
    }

    catch (IOException ex)
    {
            System.out.println("IOException is caught");
    }

    object = null;


    try
    {


            FileInputStream file = new FileInputStream
                                                        (filename);
            ObjectInputStream in = new ObjectInputStream
                                                        (file);

            object = (Emp)in.readObject();

            in.close();
            file.close();
```

```
                System.out.println("Object has been deserialized\n"
                                                + "Data after Deserialization.");
                printdata(object);


            }


            catch (IOException ex)
            {
                System.out.println("IOException is caught");
            }


            catch (ClassNotFoundException ex)
            {
                System.out.println("ClassNotFoundException" +
                                                " is caught");
            }
        }
}
```



Q18) Create an Employee HashSet collection and override equals & hashCode methods to understand how the set maintains uniqueness using these methods.

Employee.java

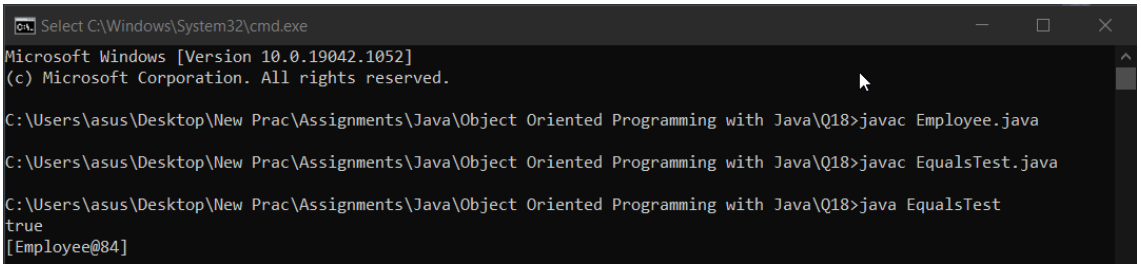
```
public class Employee
{
	private Integer id;
	private String firstname;
	private String lastName;
	private String department;

	public Integer getId()
	{
		return id;
	}
	public void setId(Integer id)
	{
		this.id = id;
	}
	public String getFirstname()
	{
		return firstname;
	}
	public void setFirstname(String firstname)
	{
		this.firstname = firstname;
	}
```

```
public String getLastName()
{
	return lastName;
}
public void setLastName(String lastName)
{
	this.lastName = lastName;
}
public String getDepartment()
{
	return department;
}
public void setDepartment(String department)
{
	this.department = department;
}
@Override
public int hashCode()
{
	final int prime = 31;
	int result = 1;
	result = prime * result + ((id == null) ? 0 : id.hashCode());
	return result;
}
@Override
public boolean equals(Object obj)
{
```

```
        if (this == obj)
                return true;
        if (obj == null)
                return false;
        if (getClass() != obj.getClass())
                return false;
        final Employee other = (Employee) obj;
        if (id == null) {
                if (other.id != null)
                        return false;
        } else if (!id.equals(other.id))
                return false;
        return true;
    }

}
```

EqualsTest.java

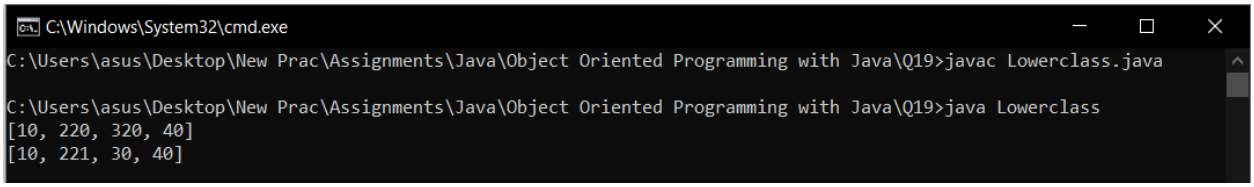
```
import java.util.HashSet;
import java.util.Set;


public class EqualsTest
{
	public static void main(String[] args)
	{
		Employee e1 = new Employee();
		Employee e2 = new Employee();
		e1.setId(101);
		e2.setId(101);
		//Prints 'true'
		System.out.println(e1.equals(e2));
		Set<Employee> employees = new HashSet<Employee>();
		employees.add(e1);
		employees.add(e2);

		System.out.println(employees);
	}
}
```



Q19) Create a Sample class to understand generic assignments using “? extends SomeClass” , “? super someclass ” and “?”.

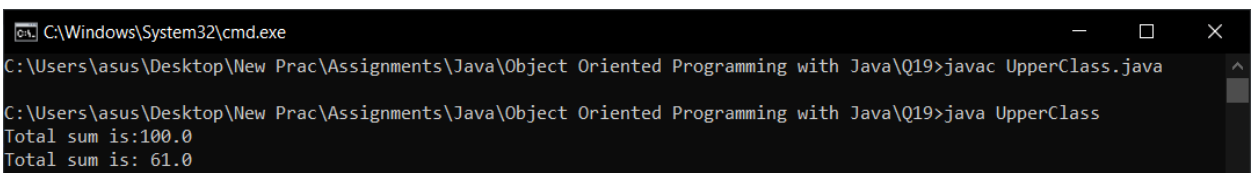
Lowerclass.java

```
import java.util.* ;

public class Lowerclass

{

   public static void main(String[] args)

   {


      List<Integer> himanshunumber = Arrays.asList(10, 220, 320, 40);

      printOnlyIntegerClassorSuperClass(himanshunumber);



      List < Number > patildoublenumber = Arrays.asList(10, 221, 30, 40);

      printOnlyIntegerClassorSuperClass(patildoublenumber);

   }


   public static void printOnlyIntegerClassorSuperClass(List < ?super Integer > list) {
//?super Integer =“? super someclass"

      System.out.println(list);

   }

}
```



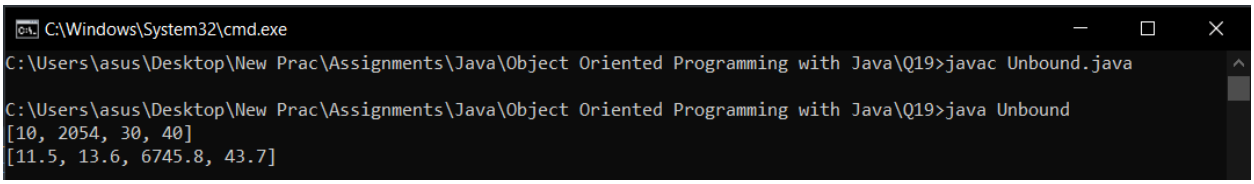
UpperClass.java

```
import java.util.*;
public class UpperClass
{
   public static void main(String[] args) {

      List<Integer> himanshunumber = Arrays.asList(10, 20, 30, 40);
      System.out.println("Total sum is:" + sum(himanshunumber));



      List < Double > patildoublenumber = Arrays.asList(13.2, 15.6, 9.7, 22.5);
      System.out.print("Total sum is: " + sum(patildoublenumber));
   }

   private static double sum(List < ?extends Number > myList) {
      double sum = 0.0;
      for (Number iterator: myList) {
         sum = sum + iterator.doubleValue();
      }
      return sum;
   }
}
```



Unbound.java

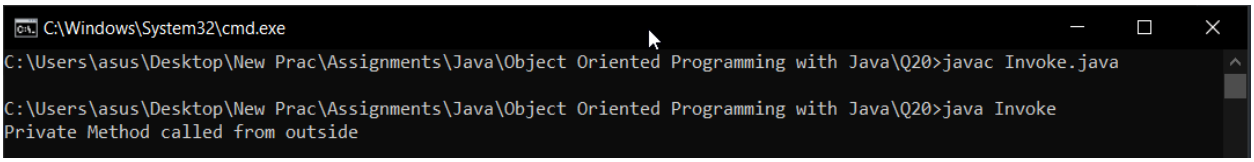
```
import java.util. * ;
public class Unbound
{
   public static void main(String[] args)
         {

      List < Integer > intList = Arrays.asList(10, 2054, 30, 40);
      List < Double > doubleList = Arrays.asList(11.5, 13.6, 6745.8, 43.7);

      printList(intList);
      printList(doubleList);
   }
   private static void printList(List < ?>list) {
      System.out.println(list);
   }
}
```



Q20) Invoke private methods of some other class using reflection.

```
import java.lang.reflect.Method;
class Check
{
private void private_Method()
        {
                System.out.println("Private Method "
                                                + "called from outside");
        }

public void printData()
        {
                System.out.println("Public Method");
        }
}



class Invoke
{

        public static void main(String[] args)throws Exception
        {
                Check c = new Check();

      Method m = Check.class.getDeclaredMethod("private_Method");
                m.setAccessible(true);
```

```
            m.invoke(c);
        }
}
```


C:\Windows\System32\cmd.exe
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q20>javac Invoke.java
C:\Users\asus\Desktop\New Prac\Assignments\Java\Object Oriented Programming with Java\Q20>java Invoke
Private Method called from outside

Q21) Create multiple threads using Thread class and Runnable interfaces.

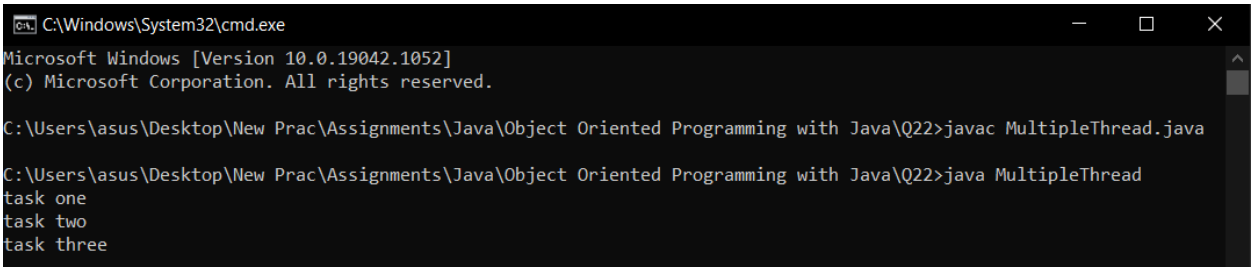
```
class RunnableDemo implements Runnable
{
  private String message;

  public RunnableDemo(String message)
  {
    this.message = message;
  }

  public void run()
  {
    while(true)
          {
      System.out.println(message);
    }
  }
}
public class TestThread
{
  public static void main(String [] args)
  {

    Runnable hello = new RunnableDemo("Hello, Greetings!!!");
    Thread thread1 = new Thread(hello);
    thread1.setDaemon(true);
```

```
    thread1.setName("hello");
    System.out.println("Starting First thread...");
    thread1.start();

    Runnable bye = new RunnableDemo("Bye for now!!");
    Thread thread2 = new Thread(bye);    thread2.setPriority(Thread.MIN_PRIORITY);
    System.out.println("Starting goodbye thread...");
    thread2.start();
    System.out.println("main() is ending...");
  }
}
```

Q22) Assign same task and different task to multiple threads.

```
class MultipleThread
{
 public static void main(String args[])
 {
  Runnable r1=new Runnable()
  {
    public void run(){
      System.out.println("task one");
  }
 };

  Runnable r2=new Runnable()
  {
    public void run()
          {
      System.out.println("task two");
          }
  };
  Runnable r3=new Runnable()
  {
    public void run()
          {
      System.out.println("task three");
    }
  };
```

```
  Thread t1=new Thread(r1);
  Thread t2=new Thread(r2);
   Thread t3=new Thread(r3);

  t1.start();
  t2.start();
  t3.start();
 }
}
```

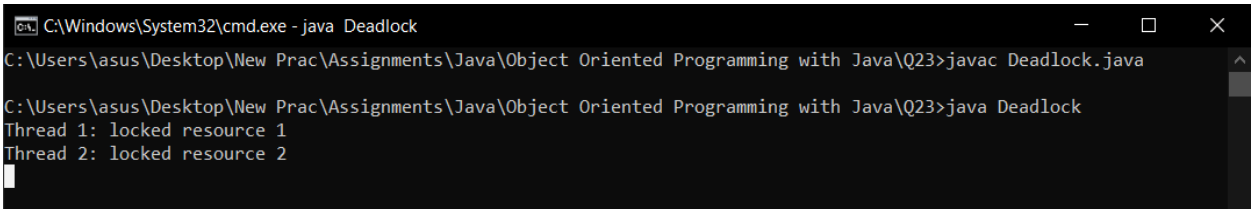


Q23) Create a Deadlock class to demonstrate deadlock in multithreading environment.

```
public class Deadlock
{
  public static void main(String[] args)
  {
    final String resource1 = "Ross";

    final String resource2 = "Rachel";

    Thread t1 = new Thread()
          {
      public void run()
            {
         synchronized (resource1)
                    {
         System.out.println("Thread 1: locked resource 1");

         try
                    {

                    Thread.sleep(100);

                    }

                    catch (Exception e) {}
```

```
      synchronized (resource2)
                    {
       System.out.println("Thread 1: locked resource 2");
      }
     }
  }
};

Thread t2 = new Thread()
       {
  public void run()
        {
   synchronized (resource2)
                 {
    System.out.println("Thread 2: locked resource 2");

    try
                 {

                 Thread.sleep(100);

                 }

                 catch (Exception e)
                 {
```

```
                }

        synchronized (resource1)
                {
          System.out.println("Thread 2: locked resource 1");
        }
       }
     }
   };
  t1.start();
  t2.start();
  }
}
```



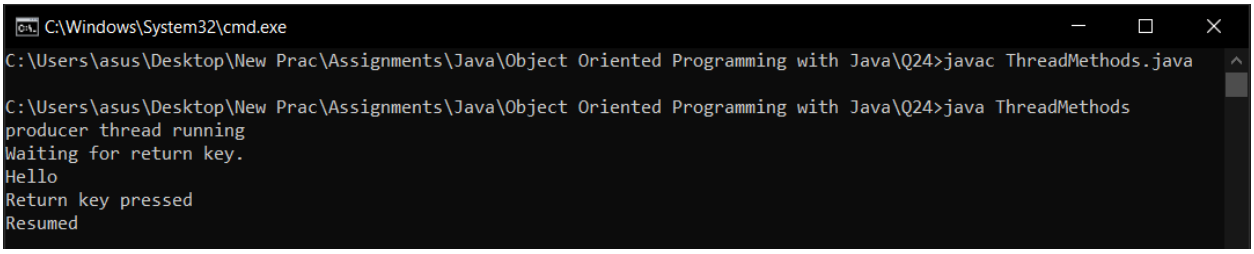
Q24) Implement wait, notify and notifyAll methods.

```
import java.util.*;
public class ThreadMethods
{
   public static void main(String[] args)
                        throws InterruptedException
   {
      final PC pc = new PC();

      Thread t1 = new Thread(new Runnable()
      {
         @Override
         public void run()
         {
            try
            {
               pc.produce();
            }
            catch(InterruptedException e)
            {
               e.printStackTrace();
            }
         }
      });

      Thread t2 = new Thread(new Runnable()
```

```
    {
        @Override
        public void run()
        {
            try
            {
                pc.consume();
            }
            catch(InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    });


    t1.start();
    t2.start();


    t1.join();
    t2.join();
}


public static class PC
{
```

```
public void produce()throws InterruptedException
{

   synchronized(this)
   {
      System.out.println("producer thread running");



      wait();



      System.out.println("Resumed");
   }
}



public void consume()throws InterruptedException
{

   Thread.sleep(1000);
   Scanner s = new Scanner(System.in);



   synchronized(this)
   {
      System.out.println("Waiting for return key.");
```

```
            s.nextLine();
            System.out.println("Return key pressed");



            notify();



            Thread.sleep(2000);
        }
    }
  }
}
```



Q25) Demonstrate how to share threadlocal data between multiple threads.

```
public class ThreadLoc
{
  public static void main (String[] args)
  {
    Tex t1 = new Tex("t1"), t2 = new Tex("t2");
    new Thread(t1).start();
    try
    {
      Thread.sleep(100);
    }
    catch (InterruptedException e)
    {}
    new Thread(t2).start();
    try
    {
      Thread.sleep(1000);
    }
    catch (InterruptedException e)
    {}
    t1.kill = true;
    t2.kill = true;
  }

  private static class Tex implements Runnable
  {
```

```
   final String name;
   Tex (String name)
   {
    this.name = name;
   }
   public boolean kill = false;
   public void run ()
   {
    TLocWrapper.get().tlint.set(System.currentTimeMillis());
    while (!kill)
    {

     System.out.println(name + ": " + TLocWrapper.get().tlint.get());
    }
   }
  }
}
class TLocWrapper
{
  public ThreadLocal<Long> tlint = new ThreadLocal<Long>();
  static final TLocWrapper self = new TLocWrapper();
  static TLocWrapper get ()
  {
   return self;
  }
  private TLocWrapper ()
  {
```

```
  }
}
```

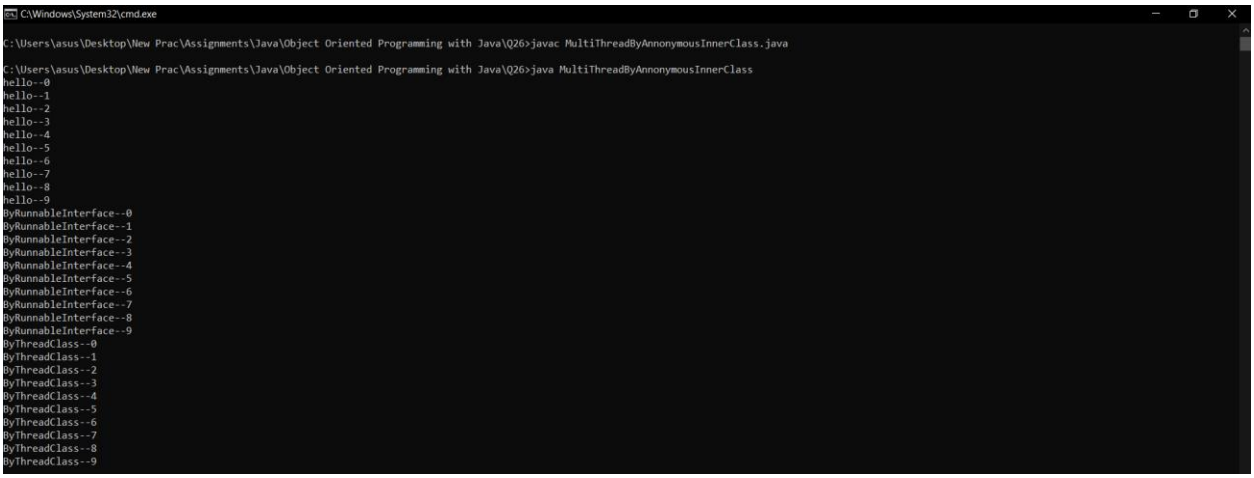

C:\Windows\System32\cmd.exe
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t1: 1624614766230
t2: 1624614766344
t2: 1624614766344

Q26) Create multiple threads using anonymous inner classes.

```
import java.io.*;
import java.util.*;

public class MultiThreadByAnnonymousInnerClass
{
        public static void main(String[] args) throws IOException
  {
     new Thread()
                {
      public void run()
                 {
       for(int i=0; i<100; i++)
                        {
         Thread.currentThread().setName("ByThreadClass");
         System.out.println(Thread.currentThread().getName()+"--"+i);
       }
      }
     }.start();


     new Thread(new Runnable()
                {
       @Override
       public void run()
                        {
```

```
   for(int i=0; i<100; i++)
                        {
      Thread.currentThread().setName("ByRunnableInterface");
      System.out.println(Thread.currentThread().getName()+"--"+i);
   }
 }

}){

}.start();

new Thread(new Runnable()
              {
   @Override
   public void run()
                        {
      for (int i = 0; i < 100; i++)
                              {
         Thread.currentThread().setName("ByRunnable");
         System.out.println("java"+"--"+i);
      }
   }
}){
   public void run()
                        {
      for (int i = 0; i < 100; i++)
                              {
```

```
            Thread.currentThread().setName("ByThread");
            System.out.println("hello"+"--"+i);
        }
      }
    }.start();


  }
}
```



Thank you,

Himanshu M Patil.

(210550381061)