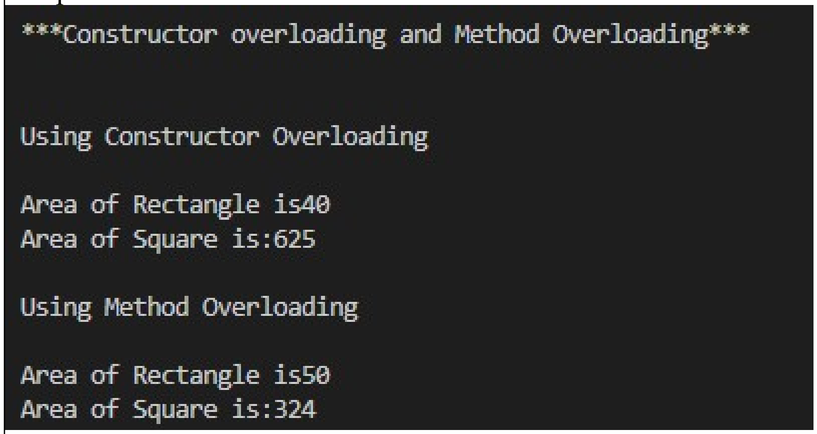
List of Programs

1. a) Write a JAVA Program to demonstrate Constructor Overloading and Method Overloading.

```java
class rectangle {
    int l, b;

    rectangle() {
        l = 0;
        b = 0;
    }

    rectangle(int m) {
        l = m;
        b = m;
    }

    rectangle(int m, int n) {
        l = m;
        b = n;
    }

    int area() {
        return (l * b);
    }

    int area(int a) {
        return (a * a);
    }

    int area(int a, int b) {
        return (a * b);
    }
}

class lab1a {
    public static void main(String args[]) {
        int area1;
        rectangle r1 = new rectangle(5, 8);
        area1 = r1.area();
        System.out.println("\n***Constructor overloading and Method
Overloading***\n");
        System.out.println("\nUsing Constructor Overloading\n");
        System.out.println("Area of Rectangle is" + area1);
        rectangle r2 = new rectangle(25);
        area1 = r2.area();

        System.out.println("Area of Square is:" + area1);
        rectangle r3 = new rectangle();
        area1 = r3.area(5, 10);
        System.out.println("\nUsing Method Overloading\n");
        System.out.println("Area of Rectangle is" + area1);
        rectangle r4 = new rectangle();
        area1 = r4.area(18);
        System.out.println("Area of Square is:" + area1);
    }
}
```

Out put

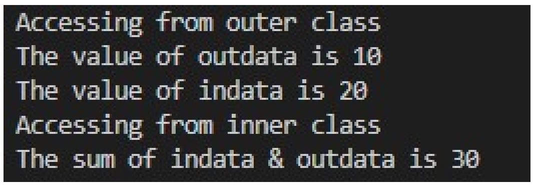


```
***Constructor overloading and Method Overloading***


Using Constructor Overloading

Area of Rectangle is40
Area of Square is:625

Using Method Overloading

Area of Rectangle is50
Area of Square is:324
```

b) Write a JAVA Program to implement Inner class and demonstrate its Access protection.

```
class outer {
    int outdata = 10;

    void display() {
        inner inobj = new inner();
        System.out.println("Accessing from outer class");
        System.out.println("The value of outdata is " + outdata);
        System.out.println("The value of indata is " + inobj.indata);
    }

    class inner {
        int indata = 20;

        void inmethod() {
            System.out.println("Accessing from inner class");
            System.out.println("The sum of indata & outdata is " + (outdata +
indata));
        }
    }
}

class lab1b {
    public static void main(String args[]) {
        outer outobj = new outer();
        outobj.display();
        outer.inner inobj1 = outobj.new inner();
        inobj1.inmethod();
    }
}
```

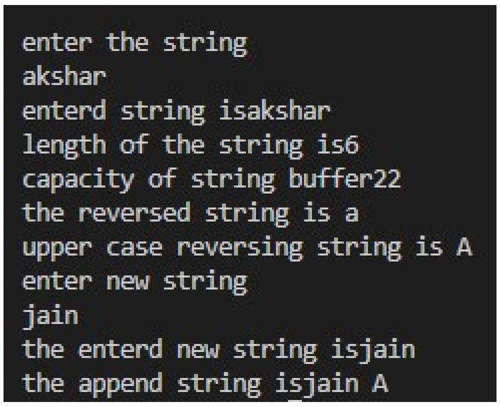
Out put



```
Accessing from outer class
The value of outdata is 10
The value of indata is 20
Accessing from inner class
The sum of indata & outdata is 30
```

2. Write a program in Java for String handling which performs the following:
   i) Checks the capacity of String Buffer objects.
   ii) Reverses the contents of a string given on console and converts the resultant string in upper case.
   iii) Reads a string from console and appends it to the resultant string of (ii).

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class StringHandler {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String s1, s2, s3, s4, s5;
        int i, l;
        s2 = " ";
        System.out.println("enter the string");
        s1 = br.readLine();
        System.out.println("enterd string is" + s1);
        System.out.println("length of the string is" + s1.length());
        StringBuffer sb = new StringBuffer(s1);
        System.out.println("capacity of string buffer" + sb.capacity());
        l = s1.length();
        if (l == 0)
            System.out.println("string is empty cannot be reversed");
        else {
            for (i = 1 - 1; i >= 0; i--) {
                s2 = s2 + s1.charAt(i);

            }
            System.out.println("the reversed string is" + s2);
            s3 = s2.toUpperCase();
            System.out.println("upper case reversing string is" + s3);
            System.out.println("enter new string");
            s4 = br.readLine();
            System.out.println("the enterd new string is" + s4);
            StringBuffer sb1 = new StringBuffer(s4);
            s5 = sb1.append(s3).toString();
            System.out.println("the append string is" + s5);

        }

    }
}
```



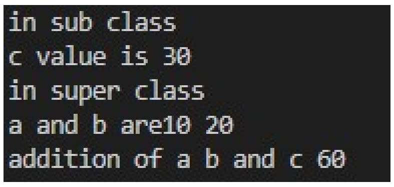
```
enter the string
akshar
enterd string isakshar
length of the string is6
capacity of string buffer22
the reversed string is a
upper case reversing string is A
enter new string
jain
the enterd new string isjain
the append string isjain A
```

3 a) Write a JAVA Program to demonstrate multi-level Inheritance.

```
class SuperClass {
    int a, b;

    SuperClass(int x, int y) {
        a = x;
        b = y;

    }

    void show() {
        System.out.println("in super class");
        System.out.println("a and b are" + a + " " + b);
    }
}

class SubClass extends SuperClass {
    int ans, add;

    SubClass(int a, int b, int c) {
        super(a, b);
        ans = c;

    }

    void show() {
        System.out.println("in sub class");
        System.out.println("c value is " + ans);
        super.show();
```

```java
        add = a + b + ans;
        System.out.println("addition of a b and c " + add);

    }
}

class lab3a {
    public static void main(String[] args) {
        SubClass ob = new SubClass(10, 20, 30);
        ob.show();
    }
}
```

Out put

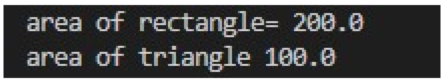


```
in sub class
c value is 30
in super class
a and b are10 20
addition of a b and c 60
```

B) Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle.

```java
interface area {
    float compute(float x, float y);

}

class rectangle {
    public float compute(float x, float y) {
        return (x * y);

    }
}

class triangle {
    public float compute(float x, float y) {
        return (x * y / 2);

    }
}

class result extends rectangle implements area {
    public float compute(float x, float y) {
        return (x * y);
    }
}

class result1 extends triangle implements area {
    public float compute(float x, float y) {
        return (x * y / 2);

    }
}

class lab3b {
    public static void main(String[] args) {
        result rect = new result();
        result1 tri = new result1();
        area a;
        a = rect;
        System.out.println("area of rectangle= " + a.compute(10, 20));
        a = tri;
        System.out.println("area of triangle " + a.compute(10, 20));

    }
}
```

Out put



```
area of rectangle= 200.0
area of triangle 100.0
```

4 Write a JAVA program which has

i) A Class called Account that creates account with 500Rs minimum balance, a deposit()method to deposit amount, a withdraw() method to withdraw amount and also throws Less Balance Exception if an account holder tries to withdraw money which makes the balance become less than 500Rs.

ii) A Class called LessBalanceException which returns the statement that says withdraw amount (Rs) is not valid.

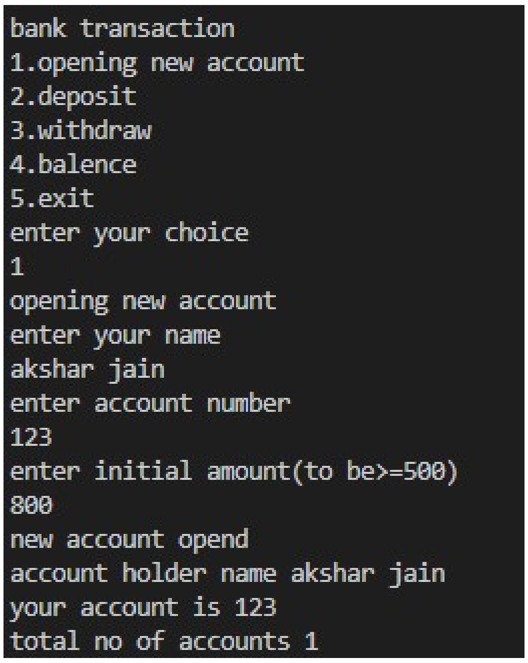
iii) A Class which creates 2 accounts, both account deposit money and one account tries to withdraw more money which generates a LessBalanceException take appropriate action for the same.

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class LessBalanceException extends Exception {
    LessBalanceException(double amt) {
        System.out.println("withdrawing " + amt + "is invalid");
    }
}

class Account {
    static int count = 0;
    int accno;
    double bal;
    String name;

    Account(double bal, String n, int accno) {
        System.out.println("new account opend");
        this.bal = bal;
        count++;
        System.out.println("account holder name " + n);
        name = n;
        System.out.println("your account is " + accno);
        this.accno = accno;
        System.out.println("total no of accounts " + count);

    }

    void deposit(double amt) {
        System.out.println("available balence " + bal);
        bal = bal + amt;
        System.out.println("rs. " + amt + " /- created");
        System.out.println("balance : " + bal);

    }

    void withdraw(double amt) throws LessBalanceException {
        System.out.println(" available balence : " + bal);
        bal -= amt;
        if (bal < 500) {
            bal += amt;
            throw new LessBalanceException(amt);

        }
        System.out.println("rs. " + amt + "/-debited");
        System.out.println("balance " + bal);

    }

    void balance() {
        System.out.println("customer information");
        System.out.println("customer name " + name);
        System.out.println("account number " + accno);
        System.out.println("balance " + bal);

    }
}
```

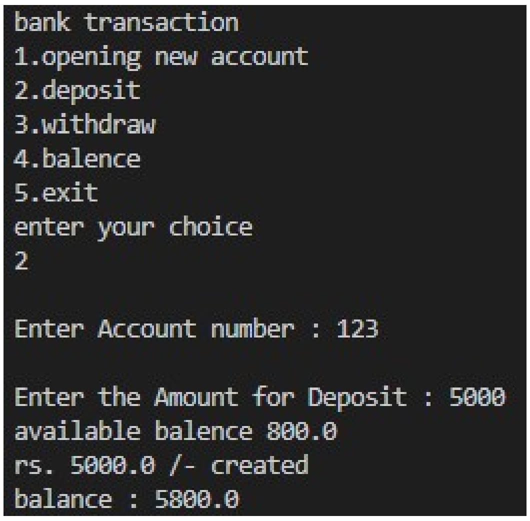
```java
class p4 {
    static int i = 0;

    public static void main(String args[]) throws IOException {
        Account ob[] = new Account[10];
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        double amt;
        String name;
        int ch, accno, k;
        boolean t = false;
        while (true) {
            System.out.println("bank transaction");
            System.out.println("1.opening new account \n2.deposit \n3.withdraw
\n4.balence \n5.exit");
            System.out.println("enter your choice ");
            ch = Integer.parseInt(br.readLine());
            switch (ch) {
                case 1:
                    System.out.println("opening new account ");
                    System.out.println("enter your name ");
                    name = br.readLine();
                    System.out.println("enter account number ");
                    accno = Integer.parseInt(br.readLine());
                    System.out.println("enter initial amount(to be>=500) ");
                    amt = Double.parseDouble(br.readLine());
                    if (amt < 500)
                        System.out.println("you cannot create an account with
less than rs.500/-");
                    else {
                        ob[i] = new Account(amt, name, accno);
                        i++;

                    }
                    break;
                case 2:
                    System.out.print("\nEnter Account number : ");
                    accno = Integer.parseInt(br.readLine());
                    for (k = 0; k < i; k++)
                        if (accno == ob[k].accno) {
                            t = true;
                            break;
                        }

                    if (t) {
                        System.out.print("\nEnter the Amount for Deposit : ");
                        amt = Double.parseDouble(br.readLine());
                        ob[k].deposit(amt);
                    } else
                        System.out.println("Invalid Account Number...!!!");
                    t = false;
                    break;

                case 3:
                    System.out.print("\nEnter Account number : ");
                    accno = Integer.parseInt(br.readLine());
                    for (k = 0; k < i; k++)
                        if (accno == ob[k].accno) {
                            t = true;
                            break;
                        }

                    if (t) {
                        System.out.print("\nEnter the Amount for Withdraw : ");
                        amt = Double.parseDouble(br.readLine());
                        try {
                            ob[k].withdraw(amt);
                        } catch (LessBalanceException e) {
                        }
                    } else
                        System.out.println("Invalid Account Number...!!!");
                    t = false;
                    break;

                case 4:
                    System.out.print("\nEnter Account number : ");
                    accno = Integer.parseInt(br.readLine());
                    for (k = 0; k < i; k++)
                        if (accno == ob[k].accno) {
                            t = true;
                            break;
                        }

                    if (t) {
                        // System.out.println(accno +" asdfsdf " +ob[k].accno);
                        ob[k].balance();
                    } else
                        System.out.println("Invalid Account Number...!!!");
                    t = false;
                    break;

                case 5:
                    System.exit(1);
                default:
                    System.out.println("Invalid Choice !!!");

            }
        }
```

```
    }
}
```

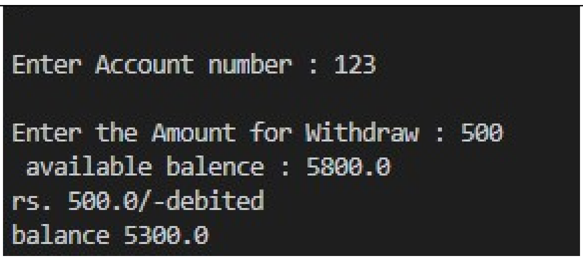
Out put



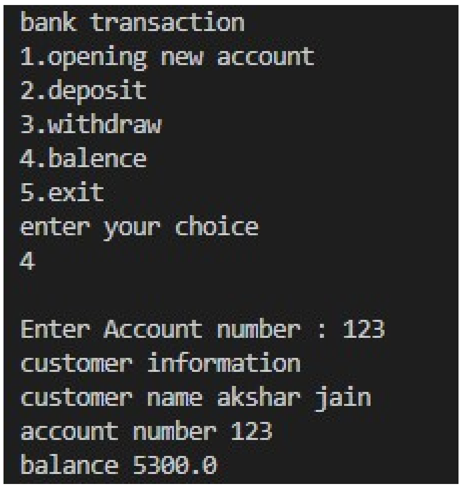
```
bank transaction
1.opening new account
2.deposit
3.withdraw
4.balence
5.exit
enter your choice
1
opening new account
enter your name
akshar jain
enter account number
123
enter initial amount(to be>=500)
800
new account opend
account holder name akshar jain
your account is 123
total no of accounts 1
```



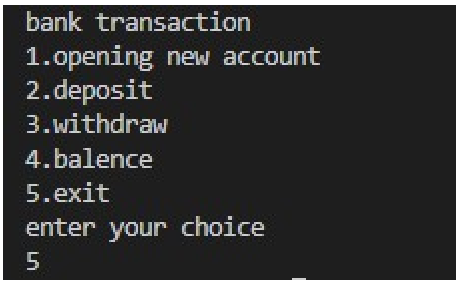
```
bank transaction
1.opening new account
2.deposit
3.withdraw
4.balence
5.exit
enter your choice
2

Enter Account number : 123

Enter the Amount for Deposit : 5000
available balence 800.0
rs. 5000.0 /- created
balance : 5800.0
```



```
Enter Account number : 123

Enter the Amount for Withdraw : 500
 available balence : 5800.0
rs. 500.0/-debited
balance 5300.0
```

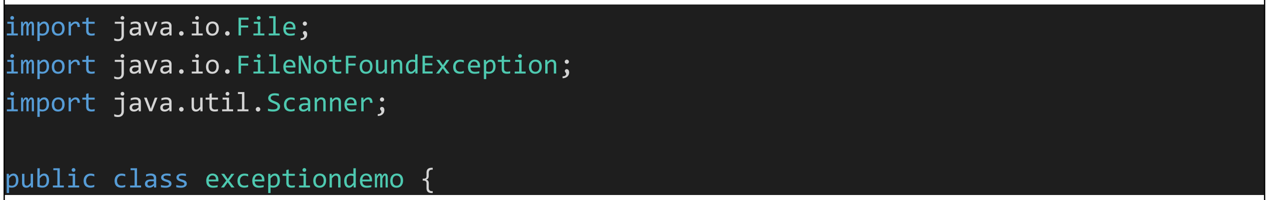


```
bank transaction
1.opening new account
2.deposit
3.withdraw
4.balence
5.exit
enter your choice
4

Enter Account number : 123
customer information
customer name akshar jain
account number 123
balance 5300.0
```

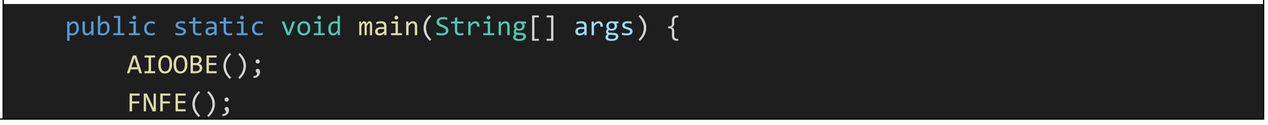


```
bank transaction
1.opening new account
2.deposit
3.withdraw
4.balence
5.exit
enter your choice
5
```

5 Write a java program to handle the following system exceptions
ArrayIndexOutOfBoundException
FileNotFoundException
NumberFormatException



```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class exceptiondemo {
```



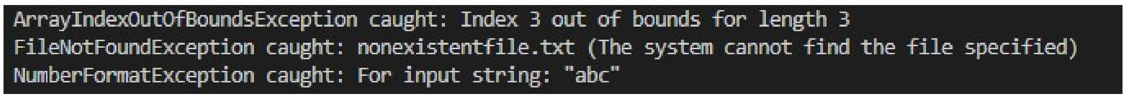
```java
    public static void main(String[] args) {
        AIOOBE();
        FNFE();
        NFE();
    }
```

```
    public static void AIOOBE() {
        try {
            int[] arr = { 1, 2, 3 };
            System.out.println(arr[3]); // Trying to access index 3 which is out
of bounds
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException caught: " +
e.getMessage());
        }
    }
```

```
    public static void FNFE() {
        try {
            File file = new File("nonexistentfile.txt");
            Scanner scanner = new Scanner(file); // Attempting to read from a
non-existent file
        } catch (FileNotFoundException e) {
            System.out.println("FileNotFoundException caught: " +
e.getMessage());
        }
    }
```

```
    public static void NFE() {
        try {
            String str = "abc";
            int num = Integer.parseInt(str); // Trying to parse a non-numeric
string
        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException caught: " +
e.getMessage());
        }
    }
}
```

Out put



```
ArrayIndexOutOfBoundsException caught: Index 3 out of bounds for length 3
FileNotFoundException caught: nonexistentfile.txt (The system cannot find the file specified)
NumberFormatException caught: For input string: "abc"
```

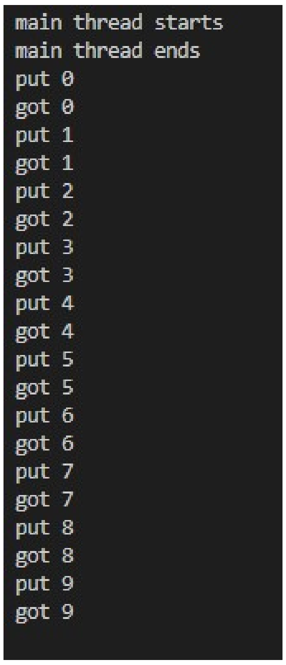
6 a) Write a JAVA program using Synchronized Threads, which demonstrates Producer Consumer concept.

```
class Q {
    int n = 0;
    boolean valset = false;
```

```java
    synchronized int get() {
        while (!valset)
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("caught");
            }
        valset = false;
        System.out.println("got " + n);
        notify();
        return n;
    }

    synchronized void put(int n) {
        while (valset)
            try {
                wait();

            } catch (Exception e) {
                System.out.println("caught");

            }
        this.n = n;
        System.out.println("put " + n);
        valset = true;
        notify();

    }
}

class Producer implements Runnable {
    Q q;

    Producer(Q q) {
        this.q = q;
        Thread t1 = new Thread(this);
        t1.start();

    }

    public void run() {
        int i = 0;
        while (i < 10) {
            q.put(i++);

        }
    }
}
```

```java
class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        Thread t2 = new Thread(this);
        t2.start();

    }

    public void run() {
        while (true) {
            q.get();
        }
    }
}

class p5 {
    public static void main(String[] args) {
        System.out.println("main thread starts");
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("main thread ends");

    }
}
```

Out put



```
main thread starts
main thread ends
put 0
got 0
put 1
got 1
put 2
got 2
put 3
got 3
put 4
got 4
put 5
got 5
put 6
got 6
put 7
got 7
put 8
got 8
put 9
got 9
```

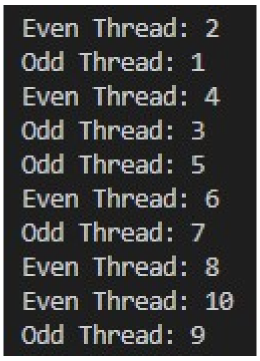
b) Design a program to create two threads, one thread will print odd numbers and second thread will print even numbers between 1 to 10 numbers

```java
class OddThread extends Thread {
    public void run() {
        for (int i = 1; i <= 10; i += 2) {
            System.out.println("Odd Thread: " + i);
            try {
                Thread.sleep(100); // Sleep for a short while to allow the other
thread to run
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class EvenThread extends Thread {
    public void run() {
        for (int i = 2; i <= 10; i += 2) {
            System.out.println("Even Thread: " + i);
            try {
                Thread.sleep(100); // Sleep for a short while to allow the other
thread to run
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```java
public class p6b {
    public static void main(String[] args) {
        OddThread oddThread = new OddThread();
        EvenThread evenThread = new EvenThread();
```

```java
        oddThread.start();
        evenThread.start();
    }
}
```

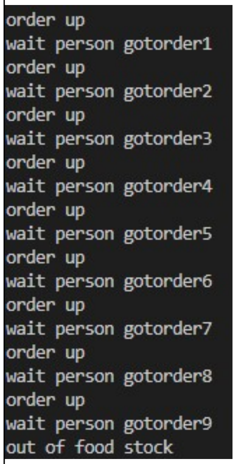
Out put



```
Even Thread: 2
Odd Thread: 1
Even Thread: 4
Odd Thread: 3
Odd Thread: 5
Even Thread: 6
Odd Thread: 7
Even Thread: 8
Even Thread: 10
Odd Thread: 9
```

7 **Write a JAVA program to implement a Queue using user defined Exception Handling (also make use of throw, throws).**

```java
class Order {
    private static int i = 0;
    private int count = i++;

    public Order() {
        if (count == 10) {
            System.out.println("out of food stock");
            System.exit(0);

        }
    }

    public String toString() {
        return "order" + count;
    }
}

class waitperson extends Thread {
    private Restaurant rest;

    public waitperson(Restaurant r) {
        rest = r;
        start();

    }

    public void run() {
        while (rest.order == null)
            synchronized (this) {
                try {
                    wait();

                } catch (InterruptedException e) {
                    throw new RuntimeException(e);

                }
                System.out.println("wait person got" + rest.order);
                rest.order = null;
            }
    }
}

class chef extends Thread {
    private Restaurant rest;
    private waitperson wp;

    public chef(Restaurant r, waitperson w) {
        rest = r;
        wp = w;
        start();
    }

    public void run() {
        while (true) {
            if (rest.order == null) {
                rest.order = new Order();
                System.out.println("order up");
                synchronized (wp) {
                    wp.notify();

                }
            }
            try {
                sleep(1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}

class Restaurant {
    Order order;

    public static void main(String[] args) {
        Restaurant rest = new Restaurant();
        waitperson wp = new waitperson(rest);
        chef ch = new chef(rest, wp);

    }
}
```
Out put



```
order up
wait person gotorder1
order up
wait person gotorder2
order up
wait person gotorder3
order up
wait person gotorder4
order up
wait person gotorder5
order up
wait person gotorder6
order up
wait person gotorder7
order up
wait person gotorder8
order up
wait person gotorder9
out of food stock
```

8 Complete the following:

i. Create a package named shape.

ii. Create some classes in the package representing some common shapes likeSquare, Triangle, and Circle.

Import and compile these classes in other program.

```
// Create a file named 'Main.java' outside of the 'shape' package
import shape.Square;
import shape.Triangle;
import shape.Circle;

public class MainClass {
    public static void main(String[] args) {
        // Creating objects of Square, Triangle, and Circle
        Square s = new Square(5);
        Triangle t = new Triangle(3, 4);
        Circle c = new Circle(2.5);

        // Printing areas and perimeters of the shapes
        System.out.println("Square - Area: " + s.getArea() + ", Perimeter: " +
s.getPerimeter());
        System.out.println("Triangle - Area: " + t.getArea() + ", Perimeter: " +
t.getPerimeter());
        System.out.println("Circle - Area: " + c.getArea() + ", Circumference: "
+ c.getCircumference());
    }
}
```

Create folder shape and insert packages
Circle.java

```
// Inside the same 'shape' folder, create a file named 'Circle.java'
package shape;

public class Circle {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double getArea() {
        return Math.PI * radius * radius;
    }

    public double getCircumference() {
        return 2 * Math.PI * radius;
    }
}
```

Square.java

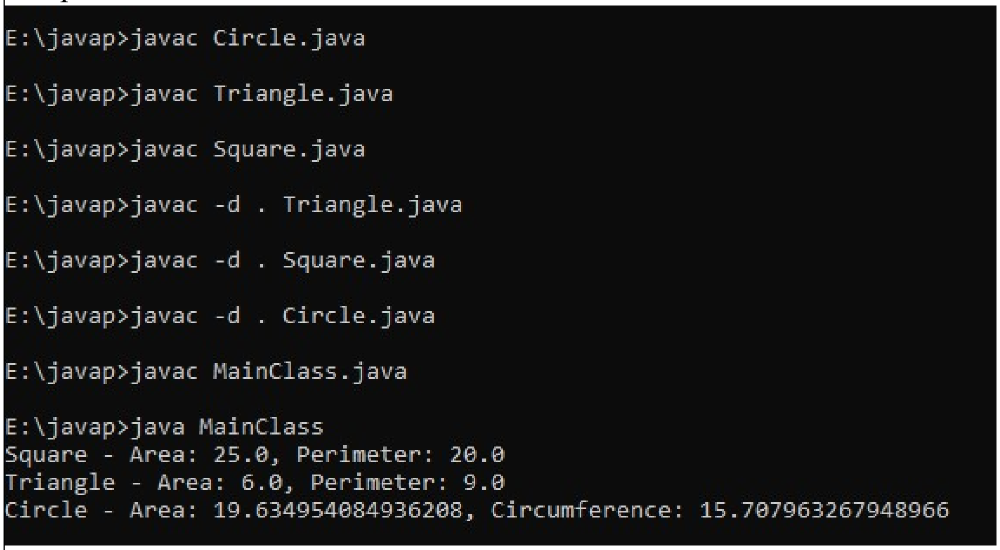
```
// Inside a folder named 'shape', create a file named 'Square.java'
package shape;

public class Square {
    private double side;

    public Square(double side) {
        this.side = side;
    }

    public double getArea() {
        return side * side;
    }

    public double getPerimeter() {
        return 4 * side;
    }
}
```

Triangle.java

```
// Inside the same 'shape' folder, create a file named 'Triangle.java'
package shape;

public class Triangle {
    private double base;
```

```java
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    public double getArea() {
        return 0.5 * base * height;
    }

    public double getPerimeter() {
        // Assuming it's an equilateral triangle for simplicity
        return 3 * base;
    }
}
```

Out put



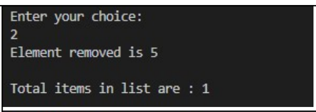
```
E:\javap>javac Circle.java

E:\javap>javac Triangle.java

E:\javap>javac Square.java

E:\javap>javac -d . Triangle.java

E:\javap>javac -d . Square.java

E:\javap>javac -d . Circle.java

E:\javap>javac MainClass.java

E:\javap>java MainClass
Square - Area: 25.0, Perimeter: 20.0
Triangle - Area: 6.0, Perimeter: 9.0
Circle - Area: 19.634954084936208, Circumference: 15.707963267948966
```

9 Write a JAVA program which has
i). A Interface class for Stack Operations
ii). A Class that implements the Stack Interface and creates a fixed length Stack. iii).A Class that implements the Stack Interface and creates a Dynamic length Stack. iv). A Class that uses both the above Stacks through Interface reference and does the
Stack operations that demonstrates the runtime binding.

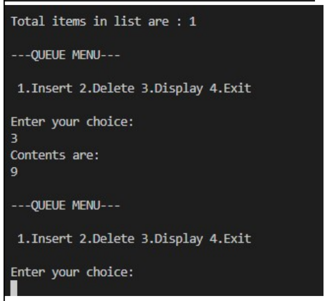
```java
//listexp.java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;

interface Stackoperations
```

```
{
    public void insert();

    public void delete();

    public void display();
}

class llist implements Stackoperations

{

    LinkedList<Integer> list = new LinkedList<Integer>();
    int i;

    int count = 0;

    public void listtest()

    {

        while (true)

        {

            System.out.println("\n---QUEUE MENU---");
            System.out.println("\n 1.Insert 2.Delete 3.Display 4.Exit");
            System.out.println("\nEnter your choice:");
            Scanner s1 = new Scanner(System.in);
            try

            {

                i = s1.nextInt();

            }

            catch (Exception e)

            {

                System.out.println("Invalid choice");

            }

            switch (i)

            {

                case 1:
                    insert();
                    break;
                case 2:
                    delete();
                    break;
                case 3:
                    display();
                    break;
                case 4:
                    System.exit(1);
                    break;
            }

        }

    }

    public void insert()

    {

        System.out.println("Enter the elements");
        Scanner s = new Scanner(System.in);
        try

        {

            list.add(s.nextInt());
            count++;

        }

        catch (Exception e)

        {

            e.printStackTrace();

        }

    }

    public void delete()

    {

        if (list.isEmpty())
            System.out.println("Stack is empty");
        else {

            System.out.println("Element removed is " + list.iterator().next());
            System.out.println("\nTotal items in list are : " + --count);
            list.removeFirst();

        }

    }

    public void display()

    {

        Iterator iterator;
        if (list.isEmpty())
            System.out.println("stack is empty");
        else {

            System.out.println("Contents are:");
            iterator = list.iterator();
            while (iterator.hasNext()) {

                System.out.println(iterator.next() + "");
            }

        }
    }

}

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

    {

        llist l = new llist();
        l.listtest();
    }
}
```

Out put



```
Enter your choice:
2
Element removed is 5

Total items in list are : 1
```

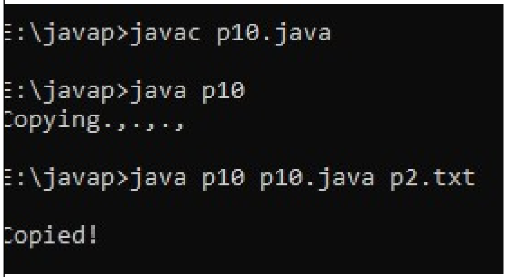


```
Total items in list are : 1

---QUEUE MENU---

 1.Insert 2.Delete 3.Display 4.Exit

Enter your choice:
3
Contents are:
9

---QUEUE MENU---

 1.Insert 2.Delete 3.Display 4.Exit

Enter your choice:
```

10 Write a JAVA Program which uses FileInputStream / FileOutPutStream Classes.

```java
import java.io.*;

class p10

{
    public static void main(String[] args) {
        FileInputStream Fin = null;
        FileOutputStream Fout = null;
        int i, j;
        if (args.length != 2) {
            System.out.println("Copying.,.,.,.,");
            return;
        }
        try {
            Fin = new FileInputStream(args[0]);

            Fout = new FileOutputStream(args[1]);
            do {
                i = Fin.read();
                if (i != -1)
                    Fout.write(i);
            } while (i != -1);
            System.out.println("\nCopied!");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            Fin.close();
            Fout.close();
        } catch (IOException e) {
            System.out.print("\nError while closing.,.,\n");
        }
    }
}
```

Out put



```
:\javap>javac p10.java

:\javap>java p10
opying.,.,.,

:\javap>java p10 p10.java p2.txt

opied!
```

11 Write a JAVA applet program, which handles keyboard event.

```java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class appletex extends Frame implements KeyListener {
    TextArea textArea;

    public void init() {
        textArea = new TextArea(10, 40);
        textArea.addKeyListener(this);
        add(textArea);
        setSize(400, 300);
    }

    public void keyTyped(KeyEvent e) {
        // Display the typed key in the text area
        textArea.append("Key Typed: " + e.getKeyChar() + "\n");
    }

    public void keyPressed(KeyEvent e) {
        // Display the pressed key in the text area
        textArea.append("Key Pressed: " + e.getKeyChar() + "\n");
    }

    public void keyReleased(KeyEvent e) {
        // Display the released key in the text area
        textArea.append("Key Released: " + e.getKeyChar() + "\n");
    }
}
/*
 * <applet code="appletex.class" width=200 height=200>
 * </applet>
 */
```

12 Write a JAVA program which uses Datagram Socket for Client Server Communication for multiple systems

```
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        DatagramSocket clientSocket = null;
        try {
            clientSocket = new DatagramSocket();
            InetAddress IPAddress = InetAddress.getByName("localhost");
            byte[] sendData;
            byte[] receiveData = new byte[1024];
            BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
            String sentence = inFromUser.readLine();
            sendData = sentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 9876);
            clientSocket.send(sendPacket);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            clientSocket.receive(receivePacket);
            String modifiedSentence = new String(receivePacket.getData());
            System.out.println("From Server: " + modifiedSentence);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (clientSocket != null) {
                clientSocket.close();
            }
        }
    }
}
```

Server

```
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] args) {
        DatagramSocket serverSocket = null;
        try {
            serverSocket = new DatagramSocket(9876);
            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];
            while (true) {
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket);
                String sentence = new String(receivePacket.getData());
                InetAddress IPAddress = receivePacket.getAddress();
                int port = receivePacket.getPort();
                System.out.println("From: " + IPAddress + ":" + port);
                System.out.println("Message: " + sentence);
                String capitalizedSentence = sentence.toUpperCase();
                sendData = capitalizedSentence.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, port);
                serverSocket.send(sendPacket);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (serverSocket != null) {
                serverSocket.close();
            }
        }
    }
}
```