RNS INSTITUTE OF TECHNOLOGY

Java Lab Manual

16MCA37

Department of MCA

1	a) Demonstrate Constructor and Method Overloading b)Demonstrate access protections of Inner class.
2	Demonstrate String handling which performs the following:i)Checks the capacity of String Buffer objects. ii)Reverses with upper case string given on console. iii)Append String read from console to resultant string of ii.
3	a)Demonstrate***Single Inheritance.*** b) Implement***Multiple inheritance***using interfaces to calculate the area of a rectangle and triangle.
4	Create Account class with 500Rs minimum balance, adeposit() and withdraw() method. withdraw() throws***LessBalanceException*** if withdraw money makes the balanceless than 500Rs. Class called ***LessBalanceException*** which returns thestatement that says withdraw amount (Rs) is not valid. .A Class which creates 2 accounts, one account tries towithdraw more money which generates a***LessBalanceException***take appropriate action for the same.
5	Demonstrates Producer Consumer concept using ***Synchronized***Threads
6	Implement a Queue using user defined Exception Handling (also make use of throw, throws.).
7	Complete the following:1. Create a package named shape. 2. Create classes in the package: Square, Triangle, and Circle. 3. Import and compile these classes in other program.
8	Demonstrate ***enumeration*** Day of Week . Add a method ***isWorkday( )*** that returns true if the value on which it is called isMONDAY through FRIDAY.
9	a)Create***Interface***class for***Stack***Operations b)Create 2 classes: F***ixedLengthStack*** and ***DynamicLengthStack***. c)Create Class that uses both Stacks through Interface reference.
10	Write a JAVA program to print a ***chessboard*** pattern.
11	Write a JAVA program which uses File Input Stream / File OutPutStream Classes.
12	Demonstrates utilities of ***LinkedList*** Class.
13	Demonstrates ***Datagram*** Socket for Client Server Communication.
14	Write a JAVA ***applet*** program, which handles keyboard event.

1a. Write a JAVA Program to implement Inner class and demonstrate its Access protection.

```
class Box
{
        double length, breadth, height;
        Box()
        {
                length = breadth = height = 0;
        }
        Box(double ln,double bh, double hh)
        {
                length = ln; breadth = bh; height = hh;
        }
        void volume(double side)
        {
                System.out.print("\nvolume is = " + side * side * side);
        }
        void volume()
        {
                System.out.print("\nvolume is = "+ length * breadth * height);
        }
    void show()
    {
        System.out.print("\n length = " + length + " breadth = " + breadth + " height = " +
height);
    }
}
class Overload
{
        public static void main(String args[])
        {
                Box b1 = new Box();
                Box b2 = new Box(2,3,4);
                b1.show();
                b2.show();
                b1.volume(3);
                b2.volume();
        }
}
```

1b. Write a JAVA Program to implement Inner class and demonstrate its Access protection.

```
class Outer
{
        void test()
```

```
        {
                Inner innerOb = new Inner();
                innerOb.setData(10);
                innerOb.display();
        }

        class Inner
        {
                private int innerVar;

                void setData(int x)
                {
                        innerVar = x;
                }

                void display()
                {
                  System.out.println("Display of inner: innerVar =" + innerVar);
                }
        }
}

class  InnerClassDemo {
        public static void main (String args[] )
        {
                Outer outOb = new Outer();
                outOb.test();
                Outer.Inner  inOb = outOb.new Inner();
                inOb.setData(30);
                inOb.display();
        }
}
```

2. Write a program in Java for String handling which performs the following:

i) Checks the capacity of StringBuffer 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.*;
class StringHandling
{
    public static void main (String args[])  throws java.io.IOException
    {
       StringBuffer str = new StringBuffer("rns");
       BufferedReader br = new BufferedReader ( new InputStreamReader (System.in) );
       int choice;
       System.out.println ("Default String: " + str );
       while(true)
       {
            System.out.println ("1: Capacity\n 2: Reverse and Converts to Upper case\n
                                        3. Append to resultant\n 4. Exit" );
            System.out.println ( "Enter your choice: ");
            choice = Integer.parseInt ( br.readLine() );
            switch (choice)
            {
                 case 1:
                    System.out.println ("The Capacity of the String=" + str.capacity() );
                    break;
                 case 2:
                    System.out.println ("The Reverse of the String=" + str.reverse() );
                    String temp = new String(str);
                    System.out.println ("String in Upper case=" + temp.toUpperCase() );
                    break;
                 case 3:
                    System.out.println ("Enter the string to concatenate: ");
                    String str2 = br.readLine();
                    System.out.println ("String after appended:" + str.append(str2) );
                    System.out.println ("String =" + str);
                    break;
              default: return;              }
       }
    } }
```

3a. Write a JAVA Program to demonstrate Inheritance.

```
import java.io.*;
class Shape
{
        protected       int width;
```

```
        protected       int height;
        public void setValue(int w,int h)
        {
                width = w;
                height = h;
        }
}

class Square extends Shape
{
        public int getArea()
        {
                return (width*height);
        }
}

public class SimpleInheritance
{
        public static void main(String args[])
        {
                Square s = new Square();
                s.setValue(10,20);
                System.out.println("Area of square is:" + s.getArea());
        }
}
```

3b. Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle .

```
import java.io.*;
import java.util.*;
interface Rectangle
{
        public void areaRectangle (double width, double height);
}
interface Triangle
{
        public void areaTriangle (double height, double breadth);
}
class Shape implements Rectangle, Triangle
{
        public void areaRectangle (double width, double height)
```

```
        {
                double area =  width * height ;
                System.out.println ("The Area of Rectangle is " + area);
        }

        public void areaTriangle (double base, double height)
        {
                double area = 0.5 * base * height ;
                System.out.println ("The Area of Triangle is " + area);
        }
}
public class MultiInheritance
{
    public static void main (String args[])  throws java.io.IOException
    {
        BufferedReader br = new BufferedReader (
                                    new InputStreamReader (System.in) );
        double width, height, base;

        System.out.println ("Enter the Width and Height of the Rectangle");
        width = Integer.parseInt (br.readLine());
        height = Integer.parseInt (br.readLine());

        Shape sh = new Shape();
        sh.areaRectangle(width, height);

        System.out.println ("\nEnter a Base and Height of Tiangle:");
        base = Integer.parseInt (br.readLine());
        height = Integer.parseInt (br.readLine());
        sh.areaTriangle(base, height);
         }
 }
```

4. Write a JAVA program which has

1. A Class called Account that creates account with 500Rs minimum balance, a deposit() method to deposit amount, a withdraw() method to withdraw amount .
2. withdraw() method throws LessBalanceException if an account holder tries to withdraw money which makes the balance become less than 500Rs.

3. A Class called LessBalanceException which returns the statement that says withdraw amount (_____ Rs) is not valid.
4. 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..

```
class Account
{
	private double balance;
	private int accNo;
	Account (int anum) {
		balance = 500;
		accNo = anum;
	}
	void deposite (double amt) {
		balance = balance + amt;
	}
	void withDraw (double amt)
	{
		try {
			if ( amt>balance )
				throw new LessBalanceException(amt);
		}
		catch ( LessBalanceException e) {
		   System.out.println ("Withdrawing More than balance"+ e);
		}
		balance = balance - amt;
		try {
			if ( balance<500 )
				throw new LessBalanceException(amt);
		}
		catch ( LessBalanceException e) {
		   System.out.println ("Balance becoming less than 500 "+ e);
		   balance += amt;
		}
		System.out.println ("Withdraw Successfully");
	}
```

```
        void getBalance()  {
                System.out.println("Current Balance= " + balance);
        }
        int getAccNo()  {
                return accNo;
        }
} //end of class Account

class LessBalanceException extends Exception
{
        double amount ;
        LessBalanceException(double amt) {
                amount = amt ;
        }
        public String toString() {
            return "withdraw amount Rs"  + amount + " is not valid..";
        }
}

class Bank
{
    public static void main(String args[]) {
        Account [] cust = new Account[2];
        int accountNo, flag;
        double amount;
                Scanner input= new Scanner(System.in);
        System.out.println("Creating Account for 2 people.....");
        System.out.println("Enter 1st Account Number: ");
        accountNo = input.nextInt() ;
                cust[0] = new Account (accountNo);

        System.out.println("Enter 2nd Account Number: ");
        accountNo = input.nextInt();
                cust[1] = new Account (accountNo);

        while(true)
        {
            System.out.println ("1. Deposite\n2. Withdraw\n3.Balance\n4.Exit\n ");
            System.out.println("Enter your choice: ");
```

```
        int ch = input.nextInt();


        switch(ch)
        {
           case 1:
                  System.out.println("Enter deposit amount: ");
                  amount = input.nextDouble();
                  System.out.println("Enter the account number:");
                  accountNo = input.nextInt();

                  if ( cust[0].getAccNo() == accountNo )
                         cust[0].deposite(amount);
                  else if ( cust[1].getAccNo() == accountNo )
                         cust[1].deposite(amount);
                  else
                      System.out.println("Invalid Account number ..s");
                  break;
           case 2:
                  System.out.println("Enter the Withdraw amount:");
                  amount = input.nextInt();
                  System.out.println("Enter the account number:");
                  accountNo = input.nextInt();

                  if(cust[0].getAccNo()==accountNo)
                         cust[0].withDraw(amount);

                  else if(cust[1].getAccNo()==accountNo)
                         cust[1].withDraw(amount);
                  else
                         System.out.println("Invalid Account number ..s");
                  break;
           case 3:
                  for ( int i=0; i<2; i++)
                         cust[i].getBalance();
                  break;
           default: return;

        } //end switch
    } //end while
```

```
    } //end main
} //end class
```

5.Write a JAVA program using Synchronized Threads, which demonstrates Producer Consumer.

```
import java.io.*;
class Producer extends Thread
{
        private Product chocolate = new Product();
        Producer (Product storeRef ) {
                chocolate = storeRef;
        }
        public void run()
        {
                for ( int prodNo = 1; prodNo <= 10; prodNo++)
                {
                        chocolate.put (prodNo);
                        try {
                                sleep(1000);
                        }
                        catch ( InterruptedException e) {
                                System.out.println("Cannot Sleep");
                        }
                }
        }
}

class Consumer extends Thread
{
        private Product chocolate = new Product();
        private int prodNo;
        Consumer (Product storeRef)  {
                chocolate = storeRef;
        }
        public void run()  {
                for ( int i = 1; i <= 10; i++) {
                        prodNo = chocolate.get();
                        System.out.println ("Consumer consumes product: " + prodNo);
                }
```

```
        }
}

class Product
{
        private int prodNo = 0;
        private boolean available = false;

        public synchronized int get()
        {
                while (available == false)
                {
                        try {
                                wait();
                        }
                        catch (InterruptedException e) {
                                //display error
                        }
                }
                available = false;
                return prodNo;
        }

        public synchronized void put ( int value )
        {
                prodNo = value;
                System.out.println ("Producer produces product" + prodNo);
                available = true;
                notify();
        }
}

class ProducerConsumer
{
        public static void main(String args[])          {
                Product store = new Product();
                Producer p1 = new Producer(store);
                Consumer c1 = new Consumer(store);
                p1.start();
                c1.start();
        }
}
```

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

```
import java.io.*;
class Queue
{
        private int[] q;
        private int max, front, rear;
        public Queue (int size)
        {
                front = rear = -1;
                max = size;
                q = new int[max];
        }

        public void insert (int item)
        {
             try {
                        if(rear == max-1)
                                throw new QueOverflowException();
                        else
                        {
                             if(front == -1)
                                  front = 0;

                             q[++rear] = item;
                             System.out.println ("Inserted Successfully");
                        }
              }
               catch (QueOverflowException  e)
               {
                  System.out.println ("Error from Exception:" + e);
               }
        }


        public void remove() throws QueUnderflowException
        {
             if(front == -1)
                  throw new QueUnderflowException();
```

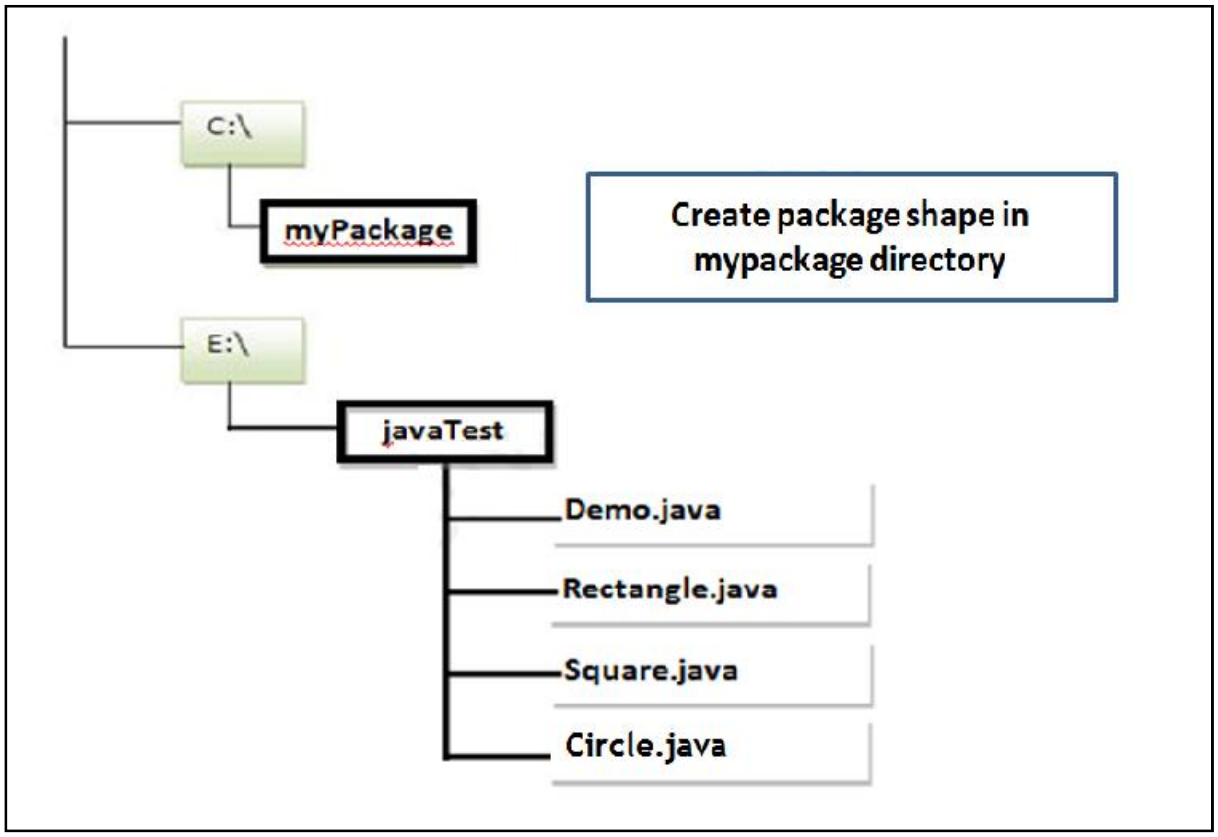
```
            else if(front == rear)
            {
                System.out.println ("Item Deleted: " + q[front]);
                front = rear = -1;
            }
            else
                System.out.println ("Item Deleted: " + q[front++]);
        }

        public void display()
        {
                if(front == -1)
                {
                        System.out.println ("Queue is Empty\n");
                        return;
                }
                else
                {
                        System.out.println ("\nThe content of Queue:");
                        for(int i = front; i<=rear ;i++)
                                System.out.print( q[i] + "   ");
                }
        }
}
class QueOverflowException extends Throwable
{
        public String toString()
        {
                return ("Overflow exception: No space to add item");
        }
}
class QueUnderflowException extends Throwable
{
        public String toString()
        {
                return ("Stack Underflow exception: ");
        }
}


class QueueDemoException
{
    public static void main (String[] args) throws IOException
    {
```

```
        int element, size, ch;
        BufferedReader br = new BufferedReader (new InputStreamReader (System.in) );
        System.out.println("Enter the Queue Size:");
        size = Integer.parseInt(br.readLine());
        Queue myQueue = new Queue(size);

        while (true)
        {
            System.out.println ("1.Insert\n2.Delete\n3.Display\n4.Exit");
            System.out.println("Enter your Choice:");
            ch = Integer.parseInt( br.readLine());

            switch (ch)
            {
               case 1:
                        System.out.println("Enter element:");
                        element = Integer.parseInt(br.readLine());
                        myQueue.insert(element);
                        break;

               case 2:
                        try {
                                myQueue.remove();
                        }
                        catch(QueUnderflowException e) {
                            System.out.println("Error from class:" + e);
                        }
                        break;

               case 3:
                        myQueue.display();
                        break;

               default: return;
            }
         }
        }
}
```

7. Complete the following:
1. Create a package named shape.
2. Create some classes in the package shape like Square, Triangle, & Circle.

3. Import and compile these classes in other program..

Create following directory structure



1. Create '*myPackage*' directory in C:\
2. Create '*javaTest*' directory in other directory like E:\
3. Create classes Rectangle.java, Square.java, circle.java and Demo.java in javaTest Directory.
4. Compile Rectangle.java, Square.java, circle.java to create package shape in myPackage diretory
5. Use this package in Demo class

Rectangle.java

```
package  Shape;
public class Rectangle {
        private double length, breadth;
        public void setRectangle ( double len, double br ) {
                length = len;
```

```
            breadth = br;
        }
        public void area(){
            double area = length * breadth;
            System.out.println ("Area of Rectangle =" + area);
        } }
```

Square.java

```
package  Shape;
public class Square {
        private double side;
        public void setSquare ( double val ) {
            side = val;
        }
        public void area() {
            System.out.println ("Area of Square=" + (side*side) );
        }
}
```

Circle.java

```
package  Shape;
public class Circle {
        private double rad;
        public void setCircle ( double radius ) {
            rad = radius;
        }
        public void area() {
            double area = (o.5)* 3.14 * rad * rad;
            System.out.println ("Area of Rectangle =" + area);
        }
}
```

Demo.java

```
import  Shape.Rectangle;
import  Shape.Square;
import  Shape.Circle;

public class Demo {
```

```
        public static void main (String [] args) {
                Rectangle rect = new Rectangle();
                rect.setRectangle (5.6, 6.4);
                rect.area();

                Square sq = new Square();
                sq.setSquare (10.5);
                sq.area();

                Circle round = new Circle();
                round.setCircle (5.6);
                round.area();
        }
}
```

Compilation steps to for creating package running Demo program

Method1: *Creating package at other directory*

E:\javaTest> *javac –d c:\mypackage Rectangle.java*

E:\javaTest> *javac –d c:\mypackage Circle.java*

E:\javaTest> *javac –d c:\mypackage Rectangle.java*

E:\javaTest> *set CLASSPATH= c:\mypackage*

E:\javaTest> *javac Demo.java*

E:\javaTest> *java Demo*

Method2: *Creating package in current directory*

E:\javaTest> *javac –d . Rectangle.java*

E:\javaTest> *javac –d . Circle.java*

E:\javaTest> *javac –d . Rectangle.java*

E:\javaTest> *javac Demo.java*

E:\javaTest> *java Demo*

8. Write a JAVA Program to Create an enumeration Day of Week with seven values SUNDAY through SATURDAY.

Add a method isWorkday( ) to the DaysOfWeek class that returns true if the value on which it is called is MONDAY through FRIDAY.

For example, the call DaysOfWeek.SUNDAY.isWorkDay ( ) returns false..

```
enum DaysOfWeek {
Sunday(1), Monday(2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6), Saturday (7);
        private int dayNo;
        DaysOfWeek ( int day ) {
                dayNo = day;
        }
        int  getDayNo() {
                                return  dayNo;
        }
        boolean isWorkDay()  {
                if ( dayNo == DaysOfWeek.Sunday.getDayNo() )
                        return false;
                if ( dayNo == DaysOfWeek.Satday. getDayNo() )
                        return false;

                return true;
        }


    public static void main(String args[]) {
        // Display price of Banana.
        boolean flag = DaysOfWeek.Monday.isWorkDay ();
        System.out.println ("WeekDay = " + flag);

        if(  flag )
            System.out.println ("Its Working Day:");
        else
            System.out.println ("Its Holiday:");

        System.out.println ("List of all days:");
        for (DaysOfWeek day : DaysOfWeek.values() )
            System.out.println ( day );
        }
}
```

9. Write a JAVA program which has

i. A Interface 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.

```
import java.io.*;

interface Stack {
        int MAX = 3;
        void push( int item);
         int pop();
        void showStack();
}

class FixedStack implements Stack
{
        int[]  stackList;
        int top;

        FixedStack () {
                stackList = new int[MAX] ;
                 top = -1;
        }
        public void push(int item)
        {
                System.out.print("Inside Push");
                if (top == MAX-1) {
                        System.out.println(".....Stack Overflow....");
                        return;
                }
                stackList [++top] = item;
                if( top == MAX-1)
                        System.out.println("..Stack is full after insert..");
                else
                        System.out.println("..Item inserted successfully..");
        }


        public int pop()
        {
                System.out.println("Inside Pop....");
                if (top == -1) {
                        System.out.println("..Stack Underflow..");
                        return -1;
```

```
                }
                else
                        return stackList [top--];
        }

        public void showStack()
        {
                System.out.print("Fixed Stack elements are:");
                for ( int i = 0; i<= top ; i++)
                        System.out.print("[" + stackList[i] + "] -");
                System.out.println("\n--------------------------");
        }
}

class DynamicStack implements Stack
{
        int[]  stackList;
        int top, SIZE;

        public DynamicStack()  {
                stackList = new int[MAX] ;
                 top = -1;
                 SIZE = MAX;
        }

        public int pop()
        {
                System.out.println("Inside Pop....");
                if (top == -1)  {
                     System.out.println("..Stack Underflow..");
                     return -1;
                }
                else
                     return   stackList [top--];
        }
        public void push (int item)
        {
                System.out.print("Inside Push");

                if (top == SIZE-1)      {
                        System.out.println(".....Reconstructing Stack....");
                        SIZE = SIZE *2;        // double the size
```

```
                int [] tempStack = new int[SIZE];
                int i = 0;
                for (int element: stackList)
                        tempStack [ i++ ] = element;
                stackList = tempStack;
                System.out.println(".....New Stack Size : " + SIZE);
            }

            stackList [++top] = item;

            if ( top == SIZE-1)
                System.out.println("..Stack is full after insert..");
            else
                System.out.println("..Item inserted successfully..");
        }

        public void showStack()
        {
            System.out.print("Fixed Stack elements are:");
            for ( int i = 0; i<= top ; i++)
                System.out.print("[" + stackList[i] + "] -");

            System.out.println("\n---------------------------");
        }
}

public class  StackOperations  {

    public static void main (String[] args)  throws IOException  {
        BufferedReader input = new BufferedReader( new InputStreamReader (System.in) );
        int choice, item;
        Stack myStack ;

        System.out.println("Menu :");
        System.out.println("1. Fixed Stack:  \n2. Dynamic Stack ");
        System.out.print("Enter Choice :");
        choice = Integer.parseInt(input.readLine());

        if (choice == 1)
        {
            System.out.print("Creating Fixed Stack :");
            myStack = new FixedStack();
        }
```

```
        else if (choice == 2)
        {
                System.out.print("Creating Dynamic Stack :");
                myStack = new DynamicStack();
        }
        else
        {
                System.out.print("Invalid choice...:Default to Fixed Stack ");
                myStack = new FixedStack();
        }

        while(true)
        {
                System.out.println("Menu :");
                System.out.println("1. Push:  \n2. Pop \n3. Display \n4. Exit");
                System.out.print("Enter Choice :");
                choice = Integer.parseInt( input.readLine() );

                switch(choice)
                {
                        case 1: System.out.println("....Push to Fixed Stack :");
                                System.out.print("Enter Element :");
                                item = Integer.parseInt(input.readLine());
                                myStack.push (item);
                                break;

                        case 2: System.out.println(".....Popping from Fixed Stack :");
                                item = myStack.pop ( );
                                if (item != -1)
                                        System.out.println(" Element poped is :" + item);
                                break;

                        case 3: System.out.print("Displaying Fixed Stack :");
                                myStack.showStack ( );
                                break;
                        default:  System.exit(0);
                }
        }  //end while
   } //end main
} //end class
```

10. Write a JAVA program to print a chessboard pattern

```
import java.awt.*;
import javax.swing.*;

public class GridLayoutExample  {

      public static void main(String[] args) {

          JFrame frame = new JFrame ("GridLayout Test");
          frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

          frame.setLayout ( new GridLayout(8, 8) );

          boolean flag = true;
          for (int val =1 ; val <=64; val ++)
          {
                    JTextField tBox = new JTextField ("      ");
                    if (flag)  {
                                        flag = false;
                                        tBox.setBackground (Color.black);
                                        frame.add (tBox);
                    }
                    else     {
                                         flag = true;
                                         frame.add (tBox);
                    }
                    if ( val % 8 == 0)
                              flag = !flag;
                    }
                     frame.pack ();
                     frame.setVisible (true);
          }
}
```

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

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

```
        try
         {
            File fileIn  = new File("d:\\source.txt");
            File fileOut = new File("d:\\target.txt");

                FileInputStream streamIn = new FileInputStream (fileIn);
                FileOutputStream streamOut = new FileOutputStream (fileOut);

                  int ch;
                while ((ch = streamIn.read()) != -1)
                  {
                            streamOut.write(ch);
                  }
                  System.out.println("\nFile Copy succeful ");
                  streamIn.close();
                  streamOut.close();
         }
         catch (FileNotFoundException e)
         {
             System.err.println("FileCopy: " + e);
         }
         catch (IOException e)
         {
             System.err.println("FileCopy: " + e);
         }
   }
}
```

12. Write JAVA programs which demonstrates utilities of LinkedList Class.

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

public class LinkedListDemo {

      public static void main(String[] args)  throws IOException
      {
          LinkedList list = new LinkedList();
          BufferedReader br = new BufferedReader (new InputStreamReader(System.in) );

          int choice;
          String element;
          while(true)
          {
```

```
System.out.println("Menu :");
menu();
System.out.print("Enter your Choice :");
choice = Integer.parseInt(br.readLine());
switch(choice)
{
 case 1:
        System.out.println("\nTo Insert at Begining :");
        System.out.print("Enter Element :");

        element = br.readLine();
        list.addFirst(element);

        System.out.println("Elements after Add :");
        System.out.println(list);
        break;

 case 2:
        System.out.println("\nTo Insert at End :");
        System.out.print("Enter Element :");

        element = br.readLine();
        list.addLast(element);

        System.out.println("Elements after Add :");
        System.out.println(list);
        Break;
case 3:
        if (list.isEmpty() == true)
                System.out.println("\nList is Empty :");
        else
        {
            element = (String) list.removeFirst();
            System.out.println("\nFirst Element removed is :" + element);
         System.out.println("\nList of Elements after removed First :");
            System.out.println(list);
        }
        break;

 case 4:
        if (list.isEmpty() == true)
                System.out.println("\nList is Empty :");
        else
        {
                element = (String) list.removeLast();
                System.out.println ("\nLast Element removed is :" + element);
```

```
                    System.out.println ("\n Elements after removed Last :" );
                        System.out.println(list);
                    }
                    break;

                case 5:
                    System.out.println("\nList of Elements is :" );
                    System.out.println(list);
                    break;

                default: java.lang.System.exit(0);
                }
            }
        }
        static void menu()
        {
            System.out.println("1. Add element at begining..");
            System.out.println("2. Add element at End..");
            System.out.println("3. Remove element at begining..");
            System.out.println("4. Remove element at End..");
            System.out.println("5. Display List..");
        }
}
```

13. Write a JAVA program which uses Datagram Socket for Client Server Communication.

Steps Creating UDP Servers

1. Create a DatagramSocket attached to a port.

   DatagramSocket serverSocket = new DatagramSocket(1234);

2. Allocate space to hold the incoming packet

   byte[] buffer = new byte[1024];

3. Read data to buffer *using switch*.

   buffer[len++] = (byte) input;

4. Create an instance of DatagramPacket to hold the incoming data.

   DatagramPacket packet = new DatagramPacket(buffer, len, InetAddress.getLocalHost(), clientPort)

5. Send packet.

   serverSocket.send (*packet*);

<!— Lab13: ServerSocket.java -- >

```
import java.net.*;

public class ServerSocket {

    public static DatagramSocket serverSocket;
    public static DatagramPacket packet;
    public static byte buffer[] = new byte[1024];

  public static void main(String args[]) throws Exception  {
        System.out.println("Server ready..\n Please type here");
   // 1. Creating  DatagramSocket attached  to  a  port.
        int serverPort = 888;
        serverSocket = new DatagramSocket (serverPort);
   // 2. Allocate space to hold the incoming packet.
        buffer = new byte[1024];
    myServer();  //calls method
  }

  public static void myServer() throws Exception
  {
    int len = 0;
    while(true)
    {
      int input = System.in.read();    // read data from user
      switch (input)
      {
        case -1:
            System.out.println("Server quits");
            return;

        case '\n':                    // Construct packet for buffered data
          int clientport = 777;

          packet = new DatagramPacket (buffer, len,
            InetAddress.getLocalHost(), clientPort ));

          serverSocket.send ( packet);     // send packet
          len = 0;
```

```
                    break;

                default:
                    buffer [len++] = (byte) input;    // store data to buffer
            }
        } //end while
    } //end myServer()

} //end class
```

Creating UDP Clients

1. Create a DatagramSocket attached to a port.

   DatagramSocket clientSocket = new DatagramSocket(clientPort);

2. Allocate space to hold the incoming packet

   byte[] buffer = new byte[1024];

3. Create an instance of DatagramPacket to hold the incoming data.

   packet = new DatagramPacket (buffer, buffer.length);

4. Block until a packet is received.

   Socket.receive (packet);

5. Convert to string and display.

   str = new String(packet.getData(), 0, packet.getLength())

   System.out.println (str);

<!— Lab13: ClientSocket.java -- >

```
import java.net.*;

public class ClientSocket {

    public static DatagramSocket client;
    public static byte buffer[] = new byte[1024];
```

```
    public static void main (String args[])  throws Exception
    {
        System.out.println ("Client - Press CTRL+C to quit");
                int  clientPort = 777;
                client = new DatagramSocket (clientPort);
        myClient();
    }
    public static void myClient ( )  throws Exception
    {
        while (true)
        {
          DatagramPacket packet = new DatagramPacket (buffer, buffer.length);
          socket.receive (packet);
          System.out.println (new String ( packet.getData(), 0, packet.getLength() ) );
        }
    }
}
```

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

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

/*   <applet code=AppletKeyboard HEIGHT=200 WIDTH=200>
      </applet>
*/

public class AppletKeyboard extends Applet implements KeyListener {
        char ch; String str;
        public void init()       // link the KeyListener with Applet
        {
                addKeyListener(this);
                requestFocus ();
        }

        // override all the 3 abstract methods of KeyListener interface
        public void keyPressed (KeyEvent e) {  //code  }
        public void keyReleased (KeyEvent e) {  //code  }
        public void keyTyped (KeyEvent e)
```

```
        {
                ch = e.getKeyChar();
                if (ch == 'a')     str = "a for apple";
                else if (ch == 'e')        str = "e for elephant";
                else if (ch == 'i')        str = "i for igloo";
                else if( ch ==  'o')       str = "o for ox";
                else if( ch == 'u')        str = "u for umbrella";
                else
                        str = "Type only vowels a, e, i, o, u only";
                repaint ( );
        }
        public void paint(Graphics g) {
                Font font = new Font("Arial",Font.BOLD,30);
                g.setFont(font);
                g.setColor(Color.blue);
                g.drawString(str, 100, 150);
                showStatus( “You typed ” + ch + " character" );
        }
}
```