R N S Institute of Technology

(*AICTE Approved, VTU Affiliated, NAAC Accredited with 'A' Grade)*

Dr. Vishnuvardhan Road, R.R Nagar Post, Channasandra, Beganluru-560098.

ESTD: 2001

An Institute with a Difference

LAB MANUAL

JAVA PROGRAMMING LABORATORY

(20MCA28)

College

VISION

" Building RNSIT into a world-class institution "

MISSION

" To impart high quality education in Engineering and Technology and Management with a difference, enabling students to excel in their career "

Department

VISION

"Synergizing Computer Applications for real world"

MISSION

Produce technologists of highest caliber to engage in design research and development, so as to enable the nation to be self-reliant

Give conceptual orientation in basic computer applications and mathematics, motivate the students for lifelong learning

Integrate project environment experiences at every level of the post graduate curriculum to give a firm practical foundation.

Prepared by

Mrs. SUMA M G

Assistant Professor, Dept. of MCA

RNIST, Bengaluru

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

Overload.java

```
class Rectangle{
     int length, breadth;
     Rectangle(){
          length=0;
          breadth=0;
     }

     Rectangle(int x, int y){
          length=x;
          breadth=y;
     }

     int rectarea(){
          return length*breadth;
     }

     int rectarea(int x, int y){
          return x*y;
     }
}

class Overload
{
     public static void main(String args[])
     {
          Rectangle r = new Rectangle(30, 50);
          System.out.println("Area of rectangle " + r.rectarea());
          System.out.println("Area or rectangle " + r.rectarea(10,10));
     }
}
```

2. Write a JAVA Program to implement Inner class and demonstrate its Access Protection.

Outer.java

```
class Outer{
     int outer_X=10;
     void test()
     {
          Inner in=new Inner();
          in.display();
     }
     class Inner
     {
          int y=20;
          void display()
          {
               System.out.println("The value of outer_X: "+ outer_X);
          }
          void  showy()
          {
               System.out.println("inner class variable y : "+ y);
          }
     }
     public static void main(String a[])
     {
          Outer outobj=new Outer();
          outobj.test();
          Outer.Inner inobj=outobj.new Inner();
          inobj.showy();
     }
}
```

3. 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).

StringHandling.java

```
import java.io.*;
class StringHandling{
   public static void main(String args[])throws java.io.IOException
   {
     StringBuffer str= new StringBuffer("Java Lab");
     BufferedReader br=new BufferedReader(new
                              InputStreamReader(System.in));
     int ch;

     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:");
          ch=Integer.parseInt(br.readLine());
          switch(ch){
               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 string to concate:");
                    String str2=br.readLine();
                    System.out.println("String after appended:" +
                                        str.append(str2));
                    System.out.println("String =" + str);
               break;
               case 4: return;
               default: return;
          }
     }
  }
}
```

4. a) Write a JAVA Program to demonstrate Inheritance.

SimpleInher.java

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

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

MulInherDemo.java

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

interface Rectangle{
     public void areaRect(int width, int height);
}

interface Triangle{
     public void areaTriangle(double base, double height);
}

class MulInher implements Rectangle, Triangle{
     public void areaRect(int width, int height )
     {
          int 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 MulInherDemo{
     public static void main(String args[])throws java.io.IOException
     {
        System.out.println("Demonstrate Multiple Inheritance");
        MulInher obj = new MulInher();
        BufferedReader br=new BufferedReader(new
                                    InputStreamReader(System.in));
        int w,h;
        double b,hgt;
        System.out.println("Enter Width and Height of the Rectangle");
        w=Integer.parseInt(br.readLine());
        h=Integer.parseInt(br.readLine());
        obj.areaRect(w,h);

        System.out.println("\nEnter base and height of the Triangle:");
        b=Double.parseDouble(br.readLine());
        hgt=Double.parseDouble(br.readLine());
        obj.areaTriangle(b,hgt);
     }
}
```

5. 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 LessBalanceException** 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.

Bank.java

```
import java.io.*;
import java.util.Scanner;
class LessBalanceException extends Throwable{
     public String toString(){
          return("Re-Enter");
     }
}
class Account{
     private double balance;
     private int acc_no;
     Account(int anum){
          balance = 500;
          acc_no=anum;
     }
     void deposite(double dep_amt){
          balance=balance+dep_amt;
     }
     void withDraw(double w_amt){
          try{
               if(w_amt>balance)
                    throw new LessBalanceException();
          }
          catch(LessBalanceException e){
               System.out.println("Withdraw amount is greater than
                                                    balance "+ e);
               return;
          }
          balance=balance-w_amt;
          try{
               if(balance<500)
                    throw new LessBalanceException();
          }
```

```
            catch(LessBalanceException e){
                System.out.println("Balance becoming less than 500 "+e);
                balance+=w_amt;
                return;
            }
            System.out.println("Withdraw Successfully");
        }
        void getBalance(){
            System.out.println("Current Balance of " + this.acc_no + "
                                        account=" +balance);
        }
        int getaccno(){
            return(acc_no);
        }
}

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

        Account[] c = new Account[2];

        int acc_no,i,flag;
        double amnt;

        Scanner inp= new Scanner(System.in);

        System.out.println("Creating Account for 2 people.....");
        System.out.println("Enter 1st Account Number: ");
        acc_no=(inp.nextInt());
        c[0] = new Account(acc_no);

        System.out.println("Enter 2nd Account Number: ");
        acc_no=(inp.nextInt());
        c[1] = new Account(acc_no);

        while(true){
            System.out.println("1. Deposite\n2. Withdraw\n 3.Balance\n
                                4.Exit\n ");
            System.out.println("Enter your choice: ");
            int ch=(inp.nextInt());

            switch(ch){
                case 1:
                    System.out.println("Enter deposit amount: ");
                    amnt=(inp.nextDouble());
                    System.out.println("Enter the account number:");
                    acc_no=(inp.nextInt());
```

```
                    if(c[0].getaccno()==acc_no)
                         c[0].deposite(amnt);
                    else if(c[1].getaccno()==acc_no)
                         c[1].deposite(amnt);
                    else
                       System.out.println("Invalid Account number..s");
               break;
               case 2:
                    System.out.println("Enter the Withdraw amount:");
                    amnt=(inp.nextInt());
                    System.out.println("Enter the account number:");
                    acc_no=(inp.nextInt());

                    if(c[0].getaccno()==acc_no)
                         c[0].withDraw(amnt);
                    else if(c[1].getaccno()==acc_no)
                         c[1].withDraw(amnt);
                    else
                       System.out.println("Invalid Account number..s");
               break;
               case 3:
                    for(i=0;i<2;i++)
                         c[i].getBalance();
               break;
               default: return;
          }//end switch
     }//end while
  }//end main()
}//end class
```

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

```
import java.io.*;

class Producer extends Thread
{
     private Product Chocolate=new Product();
     Producer(Product cadberry)
     {
          Chocolate=cadberry;
     }
     public void run()
     {
          for (int pid = 1; pid <= 10; pid++)
          {
               Chocolate.put(pid);
               try {
                       sleep((int)(Math.random() * 15000));

                    }
               catch (InterruptedException e) {   }
          }
     }
}

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

```
class Product
{
    private int prod_id=0;
     private boolean available = false;
     public synchronized int get()
     {
          while (available == false)
          {

               try
               {
                    wait();
               }
               catch (InterruptedException e){    }
          }
          available = false;
          return prod_id;
     }

     public synchronized void put(int value)
     {
          prod_id = value;
          System.out.println("Producer produces product" + prod_id);
          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();
     }
}
```

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

QueueDemoException.java

```
import java.io.*;

class Queue
{
     private int[] q;
     private int max,front,rear;
     public Queue(int n)
     {
          front = rear = -1;
          max = n;
          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("Underflow exception: Trying to remove from empty
                                                        Queue");
     }
}
class QueueDemoException
{
     public static void main(String[] args)throws java.io.IOException
     {
          int ele,n,ch;
          BufferedReader br=new BufferedReader(new
                                   InputStreamReader(System.in));
          System.out.println("Enter the Queue Size:");
          n=Integer.parseInt(br.readLine());
          Queue myq=new Queue(n);
          while(true)
          {
               System.out.println("\n1.Insert\n2.Delete\n
                                   3.Display\n4.Exit");
               System.out.println("Enter your Choice:");
               ch=Integer.parseInt(br.readLine());
               switch(ch)
               {
                    case 1:
                         System.out.println("Enter element:");
                         ele=Integer.parseInt(br.readLine());
                         myq.insert(ele);
                    break;
```

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

                    case 3:
                         myq.display();
                    break;

                    default:return;
               }
          }
     }
}
```

8. Complete the following:

i. Create a package named shape.

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

iii. Import and compile these classes in other program.

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);
     }
}
```

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);
     }
}
```

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) );
     }
}
```

PackageDemo.java

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

public class PackageDemo {
     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 for creating Package and running Demo Program

Method 1: *Creating package in current directory*

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

E:\javaLab> *javac –d . Circle.java*

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

E:\javaLab> *javac PackageDemo.java*

E:\javaLab> *java PackageDemo*

Method 2: *Creating package at other directory*

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

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

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

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

E:\javaLab> *javac PackageDemo.java*

E:\javaLab> *java PackageDemo*

9. Write a JAVA Program to create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method isWorkday( ) to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY.

For example: call DayOfWeek.SUNDAY.isWorkDay ( ) returns false.

EnumPgm.java

```
import java.io.*;
enum DaysOfWeek
{
     Sunday(1),Monday(2),Tuesday(3),Wednesday(4),
     Thursday(5),Friday(6),Saturday(7);

     private int day_num;
     DaysOfWeek(int d)
     {
          day_num=d;
     }
     int getDayNum()
     {
          return(day_num);
     }
     boolean isWorkDay()  {
     if ( day_num == DaysOfWeek.Sunday.getDayNum())
               return false;
     if (day_num == DaysOfWeek.Saturday.getDayNum())
               return false;

          return true;
     }

}
class EnumPgm
{
     public static void main(String[] args)
     {
          boolean flag = DaysOfWeek.Sunday.isWorkDay();
          System.out.println ("WeekDay = " + flag);
          if(flag)
               System.out.println ("Its Working Day:");
          else
               System.out.println ("Its Holiday:");

          System.out.println("\n****List of all days****");
          DaysOfWeek allw[]=DaysOfWeek.values();
          for(DaysOfWeek d : allw)
               System.out.println(d);
     }
}
```

10. 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.

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

interface StackOperations {
	int MAX = 3;
	void push(int item);
	void pop();
	void display();
}

class FixedStack implements StackOperations{
	int[] st;
	int top;

	public FixedStack(){
		st = new int[MAX];
		top = -1;
	}

	public void push(int item){
		if(top==MAX-1){
			System.out.println("--- Stack Overflow ---");
			return;
		}
		else{
			st[++top]=item;
			System.out.println("Inserted Successfully");
		}
	}

	public void pop(){
		if(top==-1){
			System.out.println("--- Stack Underflow ---");
			return;
		}
		System.out.println("Item deleted: " + st[top--]);
	}

	public void display(){
		if(top==-1){
			System.out.println("Stack is Empty");
			return;
```

```
            }
            else{
                  System.out.println("***Contents***");
                  for(int i=top;i>=0;i--)
                        System.out.print(st[i]+ "   ");
                  System.out.println("\n---------------");
            }
      }
}

class DynamicStack implements StackOperations{
      int[] stk;
      int top;

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

      public void push(int item){
            if(top==stk.length-1){
                  System.out.println("--- Stack Overflow ---");
                  System.out.println("***** Stack is Resized ****");
                  int[] temp=new int[stk.length*2];
                  System.out.println("..NEW STACK SIZE:"+temp.length);
                  for(int i=0;i<stk.length;i++)
                        temp[i]=stk[i];

                  stk=temp;
                  stk[++top]=item;
            }
            else{
                  stk[++top]=item;
                  System.out.println("Inserted Successfully");
            }
      }

      public void pop(){
            if(top==-1){
                  System.out.println("--- Stack Underflow ---");
                  return;
            }
            System.out.println("Item deleted: " + stk[top--]);
      }

      public void display(){
            if(top==-1){
                  System.out.println("Stack is Empty");
                  return;
            }
            else{
                  System.out.println("***Contents***");
                  for(int i=top;i>=0;i--)
                        System.out.print(stk[i]+ "   ");
```

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

class Stack{
        public static void main(String[] args)throws IOException{
                int ch,opt,item;
                StackOperations stopr;
                Scanner sc=new Scanner(System.in);

                System.out.println("1. Fixed Stack\n2. Dynamic Stack");
                System.out.println("Enter your choice:");
                ch=sc.nextInt();
                switch(ch){
                        case 1: System.out.println("*****Fixed Stack*****");
                                        stopr = new FixedStack();
                                break;
                        case 2: System.out.println("****Dynamic Stack****");
                                        stopr = new DynamicStack();
                                break;
                        default: System.out.println("Invalid Choice");
                                        return;
                }
                while(true){
                        System.out.println("1.Push\n2.Pop\n3.Display");
                        System.out.println("Enter your option:");
                        opt=sc.nextInt();
                        switch(opt){
                                case 1: System.out.println("Enter element: ");
                                                item=sc.nextInt();
                                                stopr.push(item);
                                        break;
                                case 2: stopr.pop();
                                        break;
                                case 3: stopr.display();
                                        break;
                                default: System.out.println("Invalid Choice");
                                                System.exit(0);
                        }
                }
        }
}
```

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

```
import java.io.*;

class FileHandling {
  public static void main(String[] args) {
     try{
          File fileIn  = new File("E:\\JAVALAB\\data.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  beginning");
              System.out.println("4. Remove element at End..");
              System.out.println("5. Display List..");
       }
}
```