|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Name of the Program** | **Page No** |
| **1** | Write a Java Program to demonstrate Constructor Overloading and Method overloading. |  |
| **2** | Write a Java Program to implement Inner class and demonstrate its Access Protections. |  |
| **3** | 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. |  |
| **4** | Write a JAVA Program to demonstrate Inheritance. & Simple Program on Java for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle |  |
| **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. |  |
| **6** | Write a JAVA program using Synchronized Threads, which demonstrates Producer-Consumer concept. |  |
| **7** | Complete the following:  1. Create a package named shape.  2. Create some classes in the package representing some common shapes like Square, Triangle, and Circle.  3. Import and compile these classes in other program. |  |
| **8** | Write a JAVA Program  a. Create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method is Workday( ) to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY. For example, the call DayOfWeek.SUNDAY.isWorkDay ( ) returns false. |  |
| **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. |  |
| **10** | Write a JAVA Program which uses FileInputStream / FileOutPutStream Classes. |  |
| **11** | Write a JAVA program which demonstrates utilities of LinkedList Class. |  |

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

**PROGRAM:**

class cube

{

double l,b,h;

cube()

{

System.out.println("Constructor with no arguments\n");

l=0;

b=0;

h=0;

}

cube(double i)

{

System.out.println("Constructor with one arguments\n");

l=b=h=i;

}

cube(double x,double y,double z)

{

System.out.println("Constructor with three arguments\n");

l=x;

b=y;

h=z;

}

void area()

{

System.out.println("Method with no arguments");

double a=l\*b\*h;

System.out.println("Area of Cube is "+a+"\n");

}

void area(double t)

{

System.out.println("Method with one arguments");

double a=t\*t\*t;

System.out.println("Area of Cube is "+a+"\n");

}

void area(double p,double q,double r)

{

System.out.println("Method with three arguments");

double a=p\*q\*r;

System.out.println("Area of Cube is "+a+"\n");

}

}

class pg\_1a

{

public static void main(String args[])

{

cube A=new cube(5,6,7);

A.area();

cube B=new cube(5);

B.area();

cube C=new cube();

C.area();

C.area(6);

C.area(6,7,8);

}

}

**Output:**

Constructor with three arguments

Method with no arguments

Area of Cube is 210.0

Constructor with one arguments

Method with no arguments

Area of Cube is 125.0

Constructor with no arguments

Method with no arguments

Area of Cube is 0.0

Method with one arguments

Area of Cube is 216.0

Method with three arguments

Area of Cube is 336.0

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

**PROGRAM:**

class outer

{

int a=10;

public int b=20;

private int c=30;

protected int d=40;

class inner

{

int p=5;

public int q=15;

private int r=25;

protected int s=35;

void diplay()

{

System.out.println("Inner class");

System.out.println("Value of a="+a);

System.out.println("Value of public variable"+b);

System.out.println("Value of private variable"+c);

System.out.println("Value of protected variable"+d);

}

}

void outermet()

{

inner inn=new inner();

inn.diplay();

System.out.println("Outer class");

System.out.println("Value of p="+inn.p);

System.out.println("Value of public variable"+inn.q);

System.out.println("Value of private variable"+inn.r);

System.out.println("Value of protected variable"+inn.s);

}

}

class pg\_1b

{

public static void main(String args[])

{

outer ot=new outer();

ot.outermet();

}

}

**Output:**

Inner class

Value of a=10

Value of public variable20

Value of private variable30

Value of protected variable40

Outer class

Value of p=5

Value of public variable15

Value of private variable25

Value of protected variable35

1. **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.**

**PROGRAM:**

import java.io.\*;

class lab2

{

public static void main(String args[])

{

Console in = System.console();

StringBuffer sb=new StringBuffer("Test");

System.out.println("Capacity of StringBuffer object " + sb.capacity());

System.out.println("Enter name to reverse");

String s=in.readLine();

String r=new String();

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

r+=s.charAt(i);

System.out.println("Reverse of " + s + " is " + r);

r=r.toUpperCase();

System.out.println("Its Uppercase " + r);

System.out.println("Enter String to Append");

String a=in.readLine();

System.out.println("Appending String " + a + " with " + r + " is " + r.concat(a));

sb=new StringBuffer(s);

System.out.println("Append using StringBuffer "+sb.append(a));

System.out.println("Reverse using StringBuffer "+ sb.reverse());

}

}

**Output:**

C:\mj>javac lab2.java

C:\mj>java lab2

Capacity of StringBuffer object 20

Enter name to reverse

malar

Reverse of malar is ralam

Its Uppercase RALAM

Enter String to Append

vizhi

Appending String vizhi with RALAM is RALAMvizhi

Append using StringBuffer malarvizhi

Reverse using StringBuffer ihzivralam

1. **a. Write a JAVA Program to demonstrate Inheritance.**

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

**PROGRAM**

class Box

{

double width;

double height;

double depth;

Box(Box ob) { width = ob.width; height = ob.height; depth = ob.depth; }

Box(double w, double h, double d) { width = w; height = h; depth = d; }

Box() { width = -1; height = -1; depth = -1; }

Box(double len) { width = height = depth = len; }

double volume() { return width \* height \* depth; }

}

class BoxWeight extends Box

{

double weight;

BoxWeight(double w, double h, double d, double m) { super(w,h,d); weight = m; }

}

class lab3a {

public static void main(String args[]) {

BoxWeight mybox1 = new BoxWeight(10, 20, 15, 34.3);

BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);

double vol;

vol = mybox1.volume();

System.out.println("Volume of mybox1 is " + vol);

System.out.println("Weight of mybox1 is " + mybox1.weight);

System.out.println();

vol = mybox2.volume();

System.out.println("Volume of mybox2 is " + vol);

System.out.println("Weight of mybox2 is " + mybox2.weight);

}

}

**Output:**

C:\mj>javac lab3a.java

C:\mj>java lab3a

Volume of mybox1 is 3000.0

Weight of mybox1 is 34.3

Volume of mybox2 is 24.0

Weight of mybox2 is 0.076

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

**PROGRAM**

import java.io.\*;  
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 InterfaceMain  
  {  
    public static void main(String args[])  
      {  
        result rect = new result();  
        result1 tri = new result1();  
        area a;  
        a = rect;  
        System.out.println("\nArea of rectangle = " + a.compute(10,20));  
        a = tri;  
        System.out.println("\nArea of triangle  = " +a.compute(10,2));  
      }  
  }

1. **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.**

**PROGRAM**

import java.io.\*;

import java.util.\*;

class account

{

private int bal,acno;

account(int a)

{

acno=a;

bal = 500;

System.out.println("Account " + a + " Created");

}

void deposit(int d)

{

bal+=d;

System.out.println("Amount Deposited " + d);

System.out.println("Current Balance " + bal);

}

void withdraw(int d) throws LessBalanceException

{

if(bal - d < 500)

throw new LessBalanceException(d);

else

{

bal-=d;

System.out.println("Amount Withdrawn " + d);

System.out.println("Current Balance " + bal);

}

}

void disp()

{

System.out.println("Account no " + acno + " Balance " + bal);

}

}

class LessBalanceException extends Exception

{

int a;

LessBalanceException(int a)

{

this.a=a;

}

public String toString()

{

return ("Withdrawal Amount " + a + " is not Valid");

}

}

class pg3

{

public static void main(String args[])

{

account a[]=new account[10];

Scanner s = new Scanner(System.in);

int i=1,j,ch,amt;

do

{

System.out.println("1.Creation");

System.out.println("2. Deposit");

System.out.println("3. Withdrawal");

System.out.println("4. Display");

ch=s.nextInt();

try

{

switch(ch)

{

case 1:

a[i]=new account(i);

i++;

break;

case 2:

System.out.println("Enter account no ");

j=s.nextInt();

System.out.println("Enter amount");

amt=s.nextInt();

a[j].deposit(amt);

break;

case 3:

System.out.println("Enter account no ");

j=s.nextInt();

System.out.println("Enter amount");

amt=s.nextInt();

a[j].withdraw(amt);

break;

case 4:

System.out.println("Enter Account no:");

j=s.nextInt();

a[j].disp();

break;

}

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

}while(ch!=5);

}

}

**Output:**

C:\mj>java pg3

1.Creation

2. Deposit

3. Withdrawal

4. Display

1

Account 1 Created

1.Creation

2. Deposit

3. Withdrawal

4. Display

4

Enter Account no:

1

Account no 1 Balance 500

1.Creation

2. Deposit

3. Withdrawal

4. Display

1

Account 2 Created

1.Creation

2. Deposit

3. Withdrawal

4. Display

2

Enter account no

1

Enter amount

1000

Amount Deposited 1000

Current Balance 1500

1.Creation

2. Deposit

3. Withdrawal

4. Display

4

Enter Account no:

1

Account no 1 Balance 1500

1.Creation

2. Deposit

3. Withdrawal

4. Display

3

Enter account no

1

Enter amount

1100

Withdrawal Amount 1100 is not Valid

1.Creation

2. Deposit

3. Withdrawal

4. Display

4

Enter Account no:

1

Account no 1 Balance 1500

1.Creation

2. Deposit

3. Withdrawal

4. Display

5

1. **Write a JAVA program using Synchronized Threads, which demonstrates Producer-Consumer concept.**

class Q {

int n;

boolean valueSet = false;

synchronized int get() {

while(!valueSet)

try {

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException caught");

}

System.out.println("Got: " + n);

valueSet = false;

notify();

return n;

}

synchronized void put(int n) {

while(valueSet)

try {

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException caught");

}

this.n = n;

valueSet = true;

System.out.println("Put: " + n);

notify();

}

}

class Producer implements Runnable {

Q q;

Producer(Q q) {

this.q = q;

new Thread(this, "Producer").start();

}

public void run() {

int i = 0;

while(i<=5) {

q.put(i++);

}

}

}

class Consumer implements Runnable {

Q q;

Consumer(Q q) {

this.q = q;

new Thread(this, "Consumer").start();

}

public void run() {

while(true) {

q.get();

}

}

}

class pg\_4 {

public static void main(String args[]) {

Q q = new Q();

System.out.println("Press Control-C to stop.");

new Producer(q);

new Consumer(q);

}

}

**Output:**

Press Control-C to stop.

Put: 0

Got: 0

Put: 1

Got: 1

Put: 2

Got: 2

Put: 3

Got: 3

Put: 4

Got: 4

Put: 5

Got: 5

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

**PROGRAM:**

import java.util.Scanner;

class oe extends Exception

{

int a;

oe(int a) { this.a=a;}

public String toString()

{

return "Overflow inserting " + a;

}

}

class ue extends Exception

{

int a;

ue(int a) { this.a=a;}

public String toString()

{

if(a==1)

return "Underflow";

else

return "Stack Empty";

}

}

class queue

{

int q[],r,f;

queue(int s)

{

q=new int[s];

r=0;f=0;

}

void insert(int a) throws oe

{

if(r==q.length)

throw new oe(a);

else

{

q[r++]=a;

System.out.println(a + " pushed into Stack");

System.out.println("Remaining Stack Capacity : "+(q.length - r ));

}

}

int delete() throws ue

{

if(f==r)

throw new ue(1);

else

{

System.out.println("Queue contains " + (r - f) + " elements before Deletion");

return q[f++];

}

}

void display() throws ue

{

int i;

if(f==r)

throw new ue(2);

else

for(i=f;i<r;i++)

System.out.print(q[i] + "\t");

}

}

class lab6

{

public static void main(String args[])

{

Scanner in=new Scanner(System.in);

int i,s,a,ch;

queue q;

System.out.println("Enter Queue Size :");

s=in.nextInt();

q=new queue(s);

do

{

System.out.println("1. Insert");

System.out.println("2. Delete");

System.out.println("3. Display");

System.out.println("4. Exit");

ch=in.nextInt();

try

{

switch(ch)

{

case 1:

System.out.println("Enter element to insert");

a=in.nextInt();

q.insert(a);

break;

case 2:

System.out.println(q.delete() + " deleted ");

break;

case 3:

q.display();

break;

}

}catch(oe e) {System.out.println(e);}

catch(ue e) {System.out.println(e);}

}while(ch!=4);

}

}

**Output:**

C:\mj>java lab6

Enter Queue Size :

4

1. Insert

2. Delete

3. Display

4. Exit

1

Enter element to insert

10

10 pushed into Q

Remaining Q Capacity : 3

1. Insert

2. Delete

3. Display

4. Exit

1

Enter element to insert

20

20 pushed into Q

Remaining Q Capacity : 2

1. Insert

2. Delete

3. Display

4. Exit

1

Enter element to insert

30

30 pushed into Q

Remaining Q Capacity : 1

1. Insert

2. Delete

3. Display

4. Exit

1

Enter element to insert

40

40 pushed into Q

Remaining Q Capacity : 0

1. Insert

2. Delete

3. Display

4. Exit

1

Enter element to insert

50

Overflow inserting 50

1. Insert

2. Delete

3. Display

4. Exit

3

10 20 30 40

2. Delete

3. Display

4. Exit

2

Queue contains 4 elements before

10 deleted

1. Insert

2. Delete

3. Display

4. Exit

2

Queue contains 3 elements before

20 deleted

1. Insert

2. Delete

3. Display

4. Exit

2

Queue contains 2 elements before

30 deleted

1. Insert

2. Delete

3. Display

4. Exit

2

Queue contains 1 elements before

40 deleted

1. Insert

2. Delete

3. Display

4. Exit

2

Underflow

1. Insert

2. Delete

3. Display

4. Exit

3

Queue Empty

1. Insert

2. Delete

3. Display

4. Exit

1

Enter element to insert

15

Overflow inserting 15

1. Insert

2. Delete

3. Display

4. Exit

4

1. **Write a JAVA Program**

**a. Create an enumeration Day of Week with seven values SUNDAY through SATURDAY. Add a method is Workday( ) to the DayofWeek class that returns true if the value on which it is called is MONDAY through FRIDAY. For example, the call DayOfWeek.SUNDAY.isWorkDay ( ) returns false.**

**PROGRAM**

enum dayofweek

{

SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY

Boolean isworkday()

{

if(this.ordinal() == 0 || this.ordinal() == 6)

return false;

else

return true;

}

};

Class program8

{

Public static void main(String args[])

{

Dayofweek v;

for(dayofweek i: v.values())

if(i.isworkday())

System.out.println(i + “ is a working day”);

Else

System.out.println(i + “ is not a working day”);

}

}

**Output:**

SUNDAY is not a working day.

MONDAY is a working day.

TUESDAY is a working day.

WEDNESDAY is a working day.

THURSDAY is a working day.

FRIDAY is a working day.

SATURDAY is not a working day.

**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**

|  |
| --- |
| import java.util.LinkedList; |
| import java.util.Scanner; |
|  |
| interface StackOperation { |
| public void push(); |
|  |
| public void pop(); |
|  |
| public void display(); |
| } |
|  |
| class Dystack implements StackOperation { |
| int i = 0, count = 0; |
| LinkedList<Integer> ll = new LinkedList<Integer>(); |
|  |
| public void push() { |
| int ele = 0; |
| System.out.println("Enter element :"); |
| Scanner s2 = new Scanner(System.in); |
| try { |
| ele = s2.nextInt(); |
| } catch (Exception e) { |
| System.out.println("Exception caught"); |
| } |
| ll.addFirst(ele); |
| count++; |
| } |
|  |
| public void pop() { |
| if (ll.isEmpty()) |
| System.out.println("Stack is Empty"); |
| else { |
| System.out.println("Deleted value :" + ll.removeFirst()); |
| count--; |
| } |
| } |
|  |
| @Override |
| public void display() { |
| if (ll.isEmpty()) { |
| System.out.println("list is empty"); |
|  |
| } |
| System.out.println(ll); |
| } |
| } |
|  |
| class Ststack implements StackOperation { |
| int i = 0, top = -1; |
| int a[] = new int[50]; |
| int size = 5; |
|  |
| public void push() { |
| int ele = 0; |
| if (top == size - 1) |
| System.out.println("full"); |
| else { |
| System.out.println("Enter element :"); |
| Scanner s3 = new Scanner(System.in); |
| try { |
| ele = s3.nextInt(); |
| } catch (Exception e) { |
| System.out.println("Error"); |
| } |
| a[++top] = ele; |
| } |
| } |
|  |
| public void pop() { |
| if (top == -1) |
| System.out.println("Empty"); |
| else { |
| System.out.println("Deleted value :" + a[top--]); |
|  |
| } |
| } |
|  |
| public void display() { |
| System.out.println("List is"); |
| if (top == -1) |
| System.out.println("Empty"); |
| else { |
| for (int i = top; i >= 0; i--) |
| System.out.println(a[i]); |
| } |
|  |
| } |
|  |
| public static class lab5a { |
| public static int fun() { |
| System.out |
| .println("1.push\n2.pop\n3.display\n4.exit\nenter your choice"); |
| Scanner s = new Scanner(System.in); |
| return (s.nextInt()); |
|  |
| } |
|  |
| public static void main(String[] args) { |
| Dystack ds = new Dystack(); |
| Ststack ss = new Ststack(); |
| Scanner s = new Scanner(System.in); |
| while (true) { |
| System.out |
| .println("1.dynamic \n 2.static\n 3.exit\n enter your choice"); |
| switch (s.nextInt()) { |
| case 1: |
| int val = fun(); |
| if (val == 1) { |
| ds.push(); |
| } else if (val == 2) { |
| ds.pop(); |
| } else if (val == 3) { |
| ds.display(); |
| } else { |
| System.out.println("invalid choice"); |
| } |
| break; |
| case 2: |
| int val1 = fun(); |
| if (val1 == 1) { |
| ss.push(); |
| } else if (val1 == 2) { |
| ss.pop(); |
| } else if (val1 == 3) { |
| ss.display(); |
| } else { |
| System.out.println("invalid choice"); |
| } |
| break; |
| case 3: |
| System.exit(0); |
| } |
|  |
| } |
| } |
| } |
| } |

**10. Write a JAVA Program which uses File Input Stream / File OutPut Stream Classes.**

**PROGRAM:**

import java.io.\*;

class pg\_8

{

public static void main(String args[]) throws IOException

{

If(args.length == 1) //copy text input from keyboard to file.

{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

FileOutputStream fout = new FileOutputStream(args[0]);

Char c=’ ‘ ;

System.out.println(“enter text and press ctrl + c”);

While(c != ‘q’ || c != ‘Q’)

{  
 c=br.read();

Fout.write(c);

}

Fout.close();

}

If(args.length == 2) //copy args[0] file to args[1].

{

FileInputStream fin=new FileInputStream(args[0]);

FileOutputStream fout = new FileOutputStream(args[1]);

Int i;

Do

{

i=fin.read();

If(i != -1) fout.write(i);

}

while(i != -1);

fout.close();

fin.close();

}

}

**Output:**

C:\mj>javac pg\_8.java

C:\mj> java pg\_8 f1.txt

Enter the text and press ctrl+c

This is Oxford College of Engineering

Press <ctrl+c>

C:\mj> type f1.txt

This is Oxford College of Engineering

C:\mj> java pg\_8 f1.txt f2.txt

C:\mj> type f2.txt

This is Oxford College of Engineering

**11. Write a JAVA program which demonstrates utilities of LinkedList Class**

import java.util.LinkedList;

public class LinkedListDemo {

public static void main(String[] args) {

LinkedList<String> myLinkedList = new LinkedList<String>();

myLinkedList.addFirst("A");

myLinkedList.add("B");

myLinkedList.add("C");

myLinkedList.add("D");

myLinkedList.add(2, "X");//This will add C at index 2

myLinkedList.addLast("Z");

System.out.println("Original List before deleting elements");

System.out.println(myLinkedList);

myLinkedList.remove();

myLinkedList.removeLast();

myLinkedList.remove("C");

System.out.println("Original List After deleting first and last object");

System.out.println(myLinkedList);

System.out.println("First object in linked list: "+ myLinkedList.getFirst());

System.out.println("Last object in linked list: "+ myLinkedList.peekLast());

}

}