EXPERIMENT – 1

AIM : Programs on Basic programming constructs like branching and looping

Theory : Branching and looping statements help you control the flow of the active statement within your applications. It is difficult to write even a simple application without using a branching statement, such as *If Then* or *If Then Else* .

Code with Output :

1.Simple print function

```
import java.lang.*;
import java.io.*;
public class Hello
{
   public static void main(String args[])
   {
      System.out.println("Hello!");
      System.out.print("Welcome to the Lab of OOP with JAVA.");
   }
}
```

```
Hello!
Welcome to the Lab of OOP with JAVA.
```

2.Larger of 2 numbers

```
import java.lang.*;
import java.io.*;
public class LargestNumber
{
   public static void main(String args[])
   {
      int a,b;
      a = 77;
      b = 97;
   if(a>b)
```

```
    System.out.println("The number " + a + " is the larger than " + b +".");
    else
    System.out.println("The number " + b + " is the larger than " + a +".");
    }
}
```

```
The number 97 is the larger than 77.
```

3.Largest of 3 numbers

```
import java.lang.*;
import java.io.*;
public class LargestOfThree
{
   public static void main(String[] args)
   {
      int a,b,c;
      a=64;
      b=77;
      c=97;
      if(a>b)
      {
         if(a>c)
         {
            System.out.println(a+ " is Largest.");
         }
         else
         {
            System.out.println(c+ " is Largest.");
         }
      }
      else
      {
         if(b>c)
         {
            System.out.println(b+ " is Largest.");
```

```
        }
        else
        {
            System.out.println(c+ " is Largest.");
        }
    }
  }
}
```

```
97 is Largest.
```

4.Factorial of a number

```
import java.lang.*;
import java.io.*;
public class Factorial
{
   public static void main(String args[])
   {
      int a,factorial,i;
      a=3;
      factorial=1;
      i=a;
      while (i>0)
      {
         factorial=factorial*i;
         i--;
      }
   System.out.println("The Factorial of "+a+" is " +factorial);
   }
}
```

```
The Factorial of 3 is 6
```

EXPERIMENT – 2

AIM : Program on accepting input through keyboard.

Theory : Accepting keyboard input in Java is done using a Scanner object. Consider the following statement

Scanner console = new Scanner(System.in)

This statement declares a reference variable named console.

The Scanner object is associated with standard input device (System.in). To get input from keyboard, you can call methods of Scanner class. For example in following statement nextInt() method of Scanner takes an integer and returns to variable x :

int x = console.nextInt();

Some other useful methods of Scanner class

int nextInt(), Returns the next token as an int.

float nextFloat(), Returns the next token as a float.

double nextDouble(), Returns the next token as a long.

String next(), Finds and returns the next complete token as a string ended by a blank.

String nextLine(), Returns the rest of the current line, excluding any line separator at the end.

Code with Output :

Pattern (Accept input from user)

```
import java.lang.*;
import java.io.*;
import java.util.*;
public class starPattern
{
   public static void main(String args[])
   {
      int i;
      int j;
      int k;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number of rows: ");
      k=sc.nextInt();
      for(i=1;i<=k;i++)
      {
         for(j=1;j<=i;j++)
```

```
            System.out.print("*");
            System.out.println();
        }
    }
}
```

```
Enter the number of rows:
4
*
**
***
****
```

Calculation of Simple Interest (Accept input from user)

```
import java.lang.*;
import java.io.*;
import java.util.Scanner;
public class SimpleInterest
{
    public static void main(String args[])
    {
        float p, r, t;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the Principal : ");
        p = sc.nextFloat();
        System.out.print("Enter the Rate of interest : ");
        r = sc.nextFloat();
        System.out.print("Enter the Time period : ");
        t = sc.nextFloat();
        float s;
        s=(r*t*p)/100;
        System.out.print("The Simple Interest is : " + s);
    }
}
```

```
Enter the Principal : 1
Enter the Rate of interest : 2
Enter the Time period : 3
The Simple Interest is : 0.06
```

EXPERIMENT – 3

AIM : Programs on class and objects

Theory : Concept of objects, class. Syntax, example , Multiple objects. Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities.

Class:- A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

In general, class declarations can include these components, in order:

1. Modifiers: A class can be public or has default access (Refer this for details).
2. class keyword: class keyword is used to create a class.
3. Class name: The name should begin with an initial letter (capitalized by convention).
4. Superclass(if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. Interfaces(if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. Body: The class body surrounded by braces, { }.

Object:-It is a basic unit of Object-Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

1. State: It is represented by attributes of an object. It also reflects the properties of an object.
2. Behavior: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. Identity: It gives a unique name to an object and enables one object to interact with other objects.

Declaring Objects (Also called instantiating a class):

When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any

number of instances.

Initializing an object :

The new operator instantiates a class by allocating memory for a new object and returning a
reference to that memory. The new operator also invokes the class constructor.

Ways to create object of a class:

- Using new keywords: It is the most common and general way to create objects in java. Example:
- Using Class.forName(String className) method: There is a predefined class in the java.lang package with the name Class. The forName(String className) method returns the Class object
- Using clone() method: clone() method is present in Object class. It creates and returns a copy of the object.

Code with Output :

Constructor overloading for Rectangle Class

```
import java.lang.*;
import java.io.*;
import java.util.Scanner;
public class Rectangle
{
   int length;
   int breadth;
   Rectangle()
   {
      System.out.println("Testing the execution of constructor:");
      length=breadth=0;
   }
   Rectangle(int x)
   {
      System.out.println("Testing the execution of parametrized constructor with :
argument");
      length=x;
      breadth=x;
   }
```

```
    Rectangle(int x, int y)
    {
        System.out.println("Testing the execution of parametrized constructor");
        length=x;
        breadth=y;
    }
    void area()
    {
        int a=length*breadth;
        System.out.println("The area of Rectangle is "+a);
    }
    public static void main(String args[])
    {
        Rectangle r1=new Rectangle();
        Rectangle r2=new Rectangle();
        Rectangle r3=new Rectangle(1,2);
        r3.area();
        Rectangle r4=new Rectangle(3);
    }
}
```

```
Testing the execution of constructor:
Testing the execution of constructor:
Testing the execution of parametrized constructor
The area of Rectangle is 2
Testing the execution of parametrized constructor with : argument
```

EXPERIMENT – 4

AIM : Program on method and constructor overloading.

Theory : If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

There are two ways to overload the method in java

By changing number of arguments

By changing the data type

In Java, we can overload constructors like methods.

The **constructor overloading** can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

Code with Output :

Find the area and perimeter of a circle.

Also use multiple constructors for the circle class.

```
import java.lang.*;
import java.io.*;
import java.util.Scanner;
import java.lang.Math;
public class circle
{
	float r;
	circle()
	{
		r=0;
		System.out.println("Testing the default constructor");
	}
	circle(int x)
	{
		r=x;
```

```
            System.out.println("Testing the parameterize constructor with
integer value");
        }
        circle(float x)
        {
            r=x;
            System.out.println("Testing the parameterize constructor with
float value");
        }
        void area()
        {
            float a=(float)(Math.PI*r*r);
            System.out.println("The area of circle is " + a);
        }
        void perimeter()
        {
            float a=(float)(Math.PI*2*r);
            System.out.println("The perimeter of circle is " + a);
        }
        public static void main(String[] args)
        {
            circle c1=new circle(1);
            circle c2=new circle();
            circle c3=new circle(2.3f);
            c1.area();
            c1.perimeter();
            c3.area();
            c3.perimeter();
        }
}
```

```
Testing the parameterize constructor with integer value
Testing the default constructor
Testing the parameterize constructor with float value
The area of circle is 3.1415927
The perimeter of circle is 6.2831855
The area of circle is 16.619024
The perimeter of circle is 14.451326
```

Declare a Student class with different data members.
Also put the constructor(default and parameterized), getdata() and showdata() function.

```
import java.lang.*;
import java.util.Scanner;
public class student
{
   String name;
   int roll_no;
   String branch;
   student()
   {
      name="Prathamesh Chikankar";
      roll_no=8;
      branch="CSE-(AI&ML)";
   }
   void getData()
   {
      Scanner in=new Scanner(System.in);
      System.out.println("Enter the name: ");
      name=in.nextLine();
      System.out.println("Enter the Branch: ");
      branch=in.nextLine();
      System.out.println("Enter the Roll No.: ");
      roll_no=in.nextInt();
      in.close();
```

```
    }
    void showData()
    {
        System.out.println("Name: " +name);
        System.out.println("Roll no.: " +roll_no);
        System.out.println("Branch: " +branch);
    }
    public static void main(String[] args)
    {
        student s1=new student();
        s1.getData();
        s1.showData();
    }
}
```

```
Enter the name:
Dhruv
Enter the Branch:
AI
Enter the Roll No.:
08
Name: Dhruv
Roll no.: 8
Branch: AI
```

EXPERIMENT – 5

AIM : Program on 2D array, strings functions

Theory : Multidimensional arrays- concept, declarations, processing arrays.

Multidimensional **Arrays** can be defined in simple words as an array of arrays. Data in multidimensional arrays are stored in tabular form
(in row major order).

Syntax:

data_type[1st dimension][2nd dimension][]..[Nth dimension]
array_name=new data_type[size1][size2]....[sizeN];

where:

- data_type: Type of data to be stored in the array.
  For example:int, char, etc.
- dimension: The dimension of the array created.
  For example: 1D, 2D, etc.
- array_name: Name of the array
- size1, size2, ..., sizeN: Sizes of the dimensions respectively

A number of methods provided in Java to perform operations in Strings are called **String functions**. The methods are compare(), concat(), equals(), split(), length(), replace(), compareTo() and so on. Strings in Java are constant, and it is created either using a literal or using a keyword.

Code with Output :

Input array from user & sum of all elements of array

```
import java.util.Scanner;
public class array
{
   public static void main(String[] args)
   {
      Scanner in=new Scanner(System.in);
      System.out.println("Enter the size of array : ");
      int n=in.nextInt();
      int array[]=new int[n];
```

```
        System.out.println("Enter the elements of array : ");
        for(int i=0;i<n;i++)
        {
            array[i]=in.nextInt();
        }
        for(int i=0;i<n;i++)
        {
            System.out.print(array[i] + " ");
        }
        System.out.println();
        int sum=0;
        for(int i=0;i<n;i++)
        {
            sum+=array[i];
        }
        System.out.println("Sum of array is :  " + sum);
    }
}
```

```
Enter the size of array :
4
Enter the elements of array :
1
2
3
4
1 2 3 4
Sum of array is :  10
```

Smallest & largest element in an array

```
import java.util.Scanner;
public class MinAndMax
{
    public static void main(String[] args)
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter the size of array: ");
```

```
        int n=in.nextInt();
        int array[]=new int[n];
        System.out.println("Enter the elements of array: ");
        for(int i=0;i<n;i++)
        {
           array[i]=in.nextInt();
        }
        int minimum_element=array[0],maximum_element=array[0];
        for(int i=1;i<n;i++)
        {
           if(array[i]<minimum_element)
           {
              minimum_element=array[i];
           }
        }
        for(int i=1;i<n;i++)
        {
           if(array[i]>maximum_element)
           {
              maximum_element=array[i];
           }
        }
        System.out.println("Minimum Element: " + minimum_element);
        System.out.println("Maximum Element: " + maximum_element);
    }
}
```

```
Enter the size of array:
4
Enter the elements of array:
1
9
3
7
Minimum Element: 1
Maximum Element: 9
```

Simple 2D matrix

```
import java.util.Scanner;
public class elementsOfMatrix
{
   public static void main(String[] args)
   {
      int i,j,r,c;
      int a[][]=new int[5][5];
      Scanner in=new Scanner(System.in);
      System.out.println("Enter the number of rows in the matrix");
      r=in.nextInt();
      System.out.println("Enter the number of columns in the matrix");
      c=in.nextInt();
      System.out.println("Enter the elements of matrix");
      for(i=0;i<r;i++)
      {
         for(j=0;j<c;j++)
         {
            a[i][j]=in.nextInt();
         }
      }
      System.out.println("The elements of matrix are ");
      for(i=0;i<r;i++)
      {
         for(j=0;j<c;j++)
```

```
        {
            System.out.print(a[i][j]+ " ");
        }
        System.out.println();
    }
  }
}
```

```
Enter the number of rows in the matrix
2
Enter the number of columns in the matrix
2
Enter the elements of matrix
1
2
3
4
The elements of matrix are
1 2
3 4
```

Transpose

```
import java.util.Scanner;
public class transposeOfMatrix
{
   public static void main(String[] args)
   {
      int i, j, r, c;
      int a[][] = new int[5][5];
      Scanner in = new Scanner(System.in);
      System.out.println("Enter the number of rows in the matrix");
      r = in.nextInt();
      System.out.println("Enter the number of columns in the matrix");
      c = in.nextInt();
      System.out.println("Enter the elements of matrix");
      for(i=0;i<r;i++)
      {
```

```
            for(j=0;j<c;j++)
            {
                a[i][j]=in.nextInt();
            }
        }
        System.out.println("The elements of matrix are ");
        for(i=0;i<r;i++)
        {
            for(j=0;j<c;j++)
            {
                System.out.print(a[i][j]+ " ");
            }
            System.out.println();
        }
        System.out.println("The transpose of matrix are ");
        for(i=0;i<r;i++)
        {
            for(j=0;j<c;j++)
            {
                System.out.print(a[j][i]+ " ");
            }
            System.out.println();
        }
    }
}
```

```
Enter the number of rows in the matrix
2
Enter the number of columns in the matrix
2
Enter the elements of matrix
1
2
3
4
The elements of matrix are
1 2
3 4
The transpose of matrix are
1 3
2 4
```

Addition of 2 matrix

```
import java.util.Scanner;
public class sumOfTwoMatrix
{
   public static void main(String[] args)
   {
      int i,j,r1,c1,r2,c2;
      int a[][]=new int[5][5];
      int b[][]=new int[5][5];
      Scanner in=new Scanner(System.in);
      System.out.println("Enter the number of rows of 1st matrix");
      r1=in.nextInt();
      System.out.println("Enter the number of columns of 1st matrix");
      c1=in.nextInt();
      System.out.println("Enter the elements of 1st matrix");
      for(i=0;i<r1;i++)
      {
         for(j=0;j<c1;j++)
         {
            a[i][j]=in.nextInt();
         }
      }
      System.out.println("The elements of matrix are ");
```

```
for(i=0;i<r1;i++)
{
   for(j=0;j<c1;j++)
   {
      System.out.print(a[i][j]+ " ");
   }
   System.out.println();
}
System.out.println("Enter the number of rows of 2st matrix");
r2=in.nextInt();
System.out.println("Enter the number of columns of 2st matrix");
c2=in.nextInt();
System.out.println("Enter the elements of 2st matrix");
for(i=0;i<r2;i++)
{
   for(j=0;j<c2;j++)
   {
      b[i][j]=in.nextInt();
   }
}
System.out.println("The elements of matrix are ");
for(i=0;i<r2;i++)
{
   for(j=0;j<c2;j++)
   {
      System.out.print(b[i][j]+ " ");
   }
   System.out.println();
}
if(r1==r2&&c1==c2)
{
   System.out.println("Sum of two matrix is: ");
```

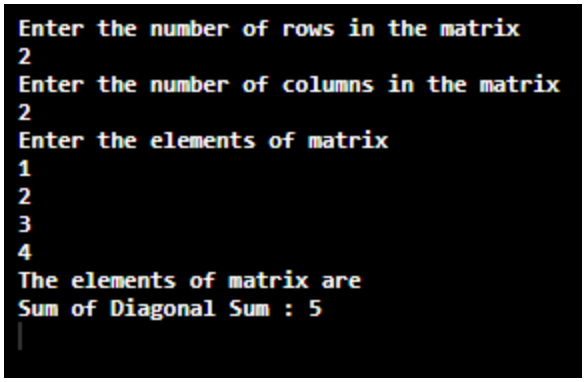
```
        for(i=0;i<r1;i++)
        {
            for(j=0;j<c1;j++)
            {
                System.out.print(a[i][j]+b[i][j]+ " ");
            }
            System.out.println();
        }
    }
    else
    {
        System.out.println("Sum of the matrix is not possible");
    }
  }
}
```

```
Enter the number of rows of 1st matrix
2
Enter the number of columns of 1st matrix
2
Enter the elements of 1st matrix
1
2
3
4
The elements of matrix are
1 2
3 4
Enter the number of rows of 2st matrix
2
Enter the number of columns of 2st matrix
2
Enter the elements of 2st matrix
1
2
3
4
The elements of matrix are
1 2 3 4
Sum of two matrix is:
2 4
6 8
```

Sum of diagonal elements of matrix

```
import java.util.Scanner;
```

```
public class diagonalSum
{
	public static void main(String[] args)
	{
		int i,j,r,c;
		int a[][]=new int[5][5];
		Scanner in=new Scanner(System.in);
		System.out.println("Enter the number of rows in the matrix");
		r=in.nextInt();
		System.out.println("Enter the number of columns in the matrix");
		c=in.nextInt();
		System.out.println("Enter the elements of matrix");
		for(i=0;i<r;i++)
		{
			for(j=0;j<c;j++)
			{
				a[i][j]=in.nextInt();
			}
		}
		int sum=0;
		System.out.println("The elements of matrix are ");
		for(i=0;i<r;i++)
		{
			sum +=a[i][i];
		}
		System.out.println("Sum of Diagonal Sum : " + sum);
	}
}
```



Sorting of elements in descending order

```
import java.util.Scanner;
public class sorting
{
   public static void main(String[] args)
   {
      Scanner in=new Scanner(System.in);
      System.out.println("Enter the size of array: ");
      int n=in.nextInt();
      int array[]=new int[n];
      System.out.println("Enter the elements of array: ");
      for(int i=0;i<n;i++)
      {
         array[i]=in.nextInt();
      }
      for(int i=0;i<n;i++)
      {
         for(int j=i+1;j<n;j++)
         {
            if(array[i]<array[j])
            {
               array[i]^=array[j];
               array[j]^=array[i];
               array[i]^=array[j];
            }
```

```
            }
        }
        System.out.print("Sorted array in descending order: ");
        for(int i=0;i<n;i++)
        {
            System.out.print(array[i] + " ");
        }
        in.close();
    }
}
```

```
Enter the size of array:
4
Enter the elements of array:
1
2
3
4
Sorted array in descending order: 4 3 2 1
```

EXPERIMENT – 6

AIM : Program on StringBuffer and Vectors

THEORY : StringBuffer and Vectors – concept, syntax, examples Important Constructors and methods of StringBuffer class and Vectors .

String Class in Java:

StringBuffer is a peer class of String that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences. StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed, to allow room for growth.

StringBuffer Constructors

1. StringBuffer() :creates an empty string buffer with the initial capacity of 16.
2. StringBuffer(String str): creates a string buffer with the specified string.
3. StringBuffer(int capacity): creates an empty string buffer with the specified capacity as length.

Vector Class in Java:

Vector is like the dynamic array which can grow or shrink its size. Unlike array, we can store n-number of elements in it as there is no size limit. It is a part of Java Collection framework since Java 1.2. It is found in the java.util package and implements the List interface, so we can use all the methods of List interface here.

Methods:

1) add() It is used to append the specified element in the given vector.
2) addAll() It is used to append all of the elements in the specified collection to the end of this Vector.
3) addElement() It is used to append the specified component to the end of this vector. It increases the vector size by one.
4) capacity() It is used to get the current capacity of this vector.
5) clear() It is used to delete all of the elements from this vector.
6) clone() It returns a clone of this vector.
7) contains() It returns true if the vector contains the specified element.

8) containsAll() It returns true if the vector contains all of the elements in the specified collection.

Code with Output :

Vowels & consonants(string)

```
import java.util.Scanner;
public class numOfVowelsAndConsonant
{
   public static boolean vowel(char a)
   {
      if(a=='a'||a=='e'||a=='i'||a=='o'||a=='u')
      {
         return true;
      }
      return false;
   }
   public static void main(String[] args)
   {
      String s="Prathamesh";
      s=s.toLowerCase();
      int n=s.length();
      int consonant=0;
      for(int i=0;i<n;i++)
      {
         if(vowel(s.charAt(i)))
         consonant++;
      }
      System.out.println("Number of vowels : "+consonant);
      System.out.println("Number of consonants : "+(n-consonant));
   }
}
```

```
Number of vowels : 3
Number of consonants : 7
```

EXPERIMENT – 7

AIM : Program on types of inheritance

THEORY : Inheritance is the most powerful feature of object-oriented programming. It allows us to inherit the properties of one class into another class.

Inheritance is a mechanism of driving a new class from an existing class. The existing (old) class is known as base class or super class or parent class. The new class is known as a derived class or sub class or child class. It allows us to use the properties and behavior of one class (parent) in another class (child). A class whose properties are inherited is known as parent class and a class that inherits the properties of the parent class is known as child class. Thus, it establishes a relationship between parent and child class that is known as parent-child or Is-a relationship.

Java supports the following four types of inheritance:

Single Inheritance

Multi-level Inheritance

Hierarchical Inheritance

Hybrid Inheritance

Code with Output :

Default constructor inheritance.

```
public class Rectangle
{
	Rectangle()
	{
		System.out.println("Testing the default constructor.");
	}
	public static void main(String args[])
	{
```

```
        Rectangle r1=new Rectangle();
        Rectangle r2=new Rectangle();
    }
}
```

```
Testing the default constructor.
Testing the default constructor.
```

Parameterized constructor inheritance.

```
public class Rectangle
{
    int length;
    int breadth;
    Rectangle()
    {
        System.out.println("Testing the execution of constructor:");
        length=breadth=0;
    }
    Rectangle(int x)
    {
        System.out.println("Testing the execution of parametrized
constructor with : argument");
        length=x;
        breadth=x;
    }
    Rectangle(int x,int y)
    {
        System.out.println("Testing the execution of parametrized
constructor");
        length=x;
        breadth=y;
    }
    void area()
    {
```

```
        int a=length*breadth;
        System.out.println("The area of rectangle is " + a);
    }
    public static void main(String args[])
    {
        Rectangle r1=new Rectangle();
        Rectangle r2=new Rectangle();
        Rectangle r3=new Rectangle(6,8);
        r3.area();
        Rectangle r4=new Rectangle(4);
    }
}
```

```
Testing the execution of constructor:
Testing the execution of constructor:
Testing the execution of parametrized constructor
The area of rectangle is 48
Testing the execution of parametrized constructor with : argument
```

EXPERIMENT – 8

AIM : Program on abstract class and abstract methods.

THEORY :

Abstract class in java: A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

Before learning the Java abstract class, let's understand the abstraction in Java first.

Abstract method in java : A method without body (no implementation) is known as abstract method. A method must always be declared in an abstract class, or in other words you can say that if a class has an abstract method, it should be declared abstract as well.

Code with Output :

Define an abstract shape class and have abstract method area.

Extend this class into circle and rectangle.

```
import java.lang.*;
import java.util.*;
import java.io.*;
abstract class shape
{
        abstract void area();
}
class rectangle extends shape
{
        void area()
        {
                Scanner a=new Scanner(System.in);
                System.out.println("Enter a the length : ");
                int l=a.nextInt();
                Scanner c=new Scanner(System.in);
                System.out.println("Enter a breadth : ");
                int b=c.nextInt();
                int area=l*b;
```

```
                System.out.println("Area of rectangle : " + area);
        }
}
class circle extends shape
{
        void area()
        {
                Scanner d=new Scanner(System.in);
                System.out.println("Enter a the radius : ");
                int r=d.nextInt();
                int area=(22/7)*(r*r);
                System.out.println("Area of circle is : " + area);
        }
}
public class abstractshape
{
        public static void main(String args[])
        {
                System.out.println("RECTANGLE");
                rectangle r1=new rectangle();
                r1.area();
                System.out.println("CIRCLE");
                circle c1=new circle();
                c1.area();
        }
}
```

```
RECTANCLE
Enter a the length :
2
Enter a breadth :
4
Area of rectangle : 8
CIRCLE
Enter a the radius :
3
Area of circle is : 27
```

EXPERIMENT – 9

AIM : Program using super and final keyword

THEORY : super and final keyword

Super keyword in java :

The super keyword in Java is a reference variable which is used to refer to an immediate
parent class object.
Whenever you create an instance of a subclass, an instance of the parent class is
created implicitly which is referred to by a super reference variable.
Usage of Java super Keyword
1. super can be used to refer to the immediate parent class instance variable.
2. super can be used to invoke immediate parent class methods.
3. super() can be used to invoke immediate parent class constructor.

Final keyword in java:

In Java, the Final keyword can be used while declaring an entity. Using the final keyword means that the value can't be modified in the future. This entity can be - but is not limited to - a variable, a class or a method.

Code with Output :

Use Final keyword with function, class, variable and study the difference when you try to change the value.

```
import java.util.*;
class Person
{
   String Name;
   String dateOfBirth;

   Person()
   {
      Name="Dhruv";
```

```
        dateOfBirth="31/07/2001";
    }

    Person(String a,String b)
    {
        Name=a;
        dateOfBirth=b;
    }
    void getdata()
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter name: ");
        Name=in.nextLine();
        System.out.println("Enter DOB: ");
        dateOfBirth=in.nextLine();
    }
    void showdata()
    {
        System.out.println("Name: " + Name);
        System.out.println("DOB: " + dateOfBirth);
    }
}
class Student extends Person
{
    int roll_no;
    Student()
    {
        super();
        roll_no=97;
    }

    Student(String a,String b,int c)
    {
```

```
        super(a,b);
        roll_no=c;
    }
    void getdata()
    {
        Scanner in=new Scanner(System.in);
        super.getdata();
        System.out.println("Enter Roll No.: ");
        roll_no=in.nextInt();
    }
    void showdata()
    {
        super.showdata();
        System.out.println("Roll No.: " + roll_no);
    }
}
class Faculty extends Person
{
    int faculty_ID;
    Faculty()
    {
        super();
        faculty_ID=77;
    }
    Faculty(String a,String b,int c)
    {
        super(a,b);
        faculty_ID=c;
    }
    void getdata()
    {
        Scanner in=new Scanner(System.in);
        super.getdata();
```

```
        System.out.println("Enter Faculty ID: ");
        faculty_ID=in.nextInt();
    }
    void showdata()
    {
        super.showdata();
        System.out.println("Faculty ID: " + faculty_ID);
    }
}
public class Main
{
    public static void main(String[] args)
    {
        Person p1=new Person();
        Person p2=new Person("Dhruv", "31/07/2001");
        p1.showdata();
        p2.showdata();
        p1.getdata();
        p1.showdata();

        Student s1=new Student();
        Student s2=new Student("DhruvDP", "31/07/2002", 123);
        s1.showdata();
        s2.showdata();
        s1.getdata();
        s1.showdata();

        Faculty f1=new Faculty();
        Faculty f2=new Faculty("Aayan", "31/07/2003", 12345);
        f1.showdata();
        f2.showdata();
        f1.getdata();
        f1.showdata();
```

```
    }
}
```

```
Name: Dhruv
DOB: 31/07/2001
Name: Dhruv
DOB: 31/07/2001
Enter name:
Prathamesh
Enter DOB:
18/09/2001
Name: Prathamesh
DOB: 18/089/2001
Name: Dhruv
DOB: 31/07/2001
Roll No.: 97
Name: DhruvDP
DOB: 31/07/2002
Roll No.: 123
Enter name:
PrathameshPC
Enter DOB:
18/09/2002
Enter Roll No.:
08
Name: PrathameshPC
DOB: 18/09/2002
Roll No.: 808
Name: Dhruv
DOB: 31/07/2001
Faculty ID: 77
Name: Aayan
DOB: 31/07/2003
Faculty ID: 12345
Enter name:
PrathameshPC77
Enter DOB:
18/09/2003
Enter Faculty ID:
1
Name: PrathameshPC77
DOB: 18/09/20023
Faculty ID: 1
```

EXPERIMENT – 10

AIM : Program on Exception handling

THEORY : The **Exception Handling** in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained. Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. There are mainly two types of exceptions: checked and unchecked.

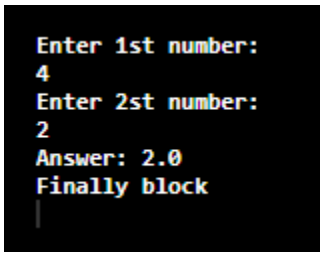
Code with Output :

Write a simple division program using try and catch block to study different types of exceptions(Accept number input from user)

Use finally block in same program

```
import java.util.*;
import java.lang.*;
public class division
{
	public static void main(String[]args)
	{
		Scanner in=new Scanner(System.in);
		try
		{
			System.out.println("Enter 1st number: ");
			int a=in.nextInt();
			System.out.println("Enter 2st number: ");
			int b=in.nextInt();
			float c=0;
			c=(float)a/b;
			System.out.println("Answer: " + c);
```

```
            }
            catch(ArithmeticException e)
            {
                  System.out.println("Number cannot divide by zero\nTry some
other value for Denominator.");
            }
            catch (InputMismatchException e)
            {
                  System.out.println("Please enter only Integer values");
            }
            finally
            {
                  System.out.println("Finally block");
            }
      }
}
```



EXPERIMENT – 11

AIM : Program on user defined exception

THEORY : Java provides us facility to create our own exceptions which are basically derived classes of Exception. **User Defined Exception** or custom exception is creating your own exception class and throws that exception using 'throw' keyword. This can be done by extending the class Exception.

Code with Output :

Throwing a user defined exception

```
import java.lang.*;
public class userdefinedexception
{
   public static void main (String args[])
   {
      try
      {
         int a=5,b=2;
         if (b==0)
         throw new Exception("Denominator is zero");
         else
         {
            int c=a/b;
            System.out.println("The result is :  "+c);
         }
      }
      catch(Exception e)
      {
         System.out.println(e);
      }
   }
}
```

```
The result is :  2
```

EXPERIMENT – 12

AIM : Program on Multithreading

THEORY : **Multithreading** in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

Code with Output :

```
class counter
{
	int count;
	public synchronized void increment()
	{
		count++;
	}
}
public class multithreading
{
	public static void main(String[] args) throws InterruptedException
	{
		counter c=new counter();
		Thread t1=new Thread(new Runnable()
		{
			public void run()
			{
				for(int i=1;i<=100000; i++)
				{
					c.increment();
				}
				System.out.println("Hello world! ");
			}
```

```
        });
        Thread t2=new Thread(new Runnable()
        {
            public void run()
            {
                for(int i=1;i<=100000;i++)
                {
                    c.increment();
                }
                System.out.println("Hello world! ");
             }
        });
        t1.start();
        t2.start();
        t1.join();
        t2.join();
        System.out.println(c.count + " is the count ");
    }
}
```

```
Hello world!
Hello world!
200000 is the count
```

EXPERIMENT – 13

AIM : Program on applet class

THEORY : **Applet** is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

*Advantage of Applet

There are many advantages of applet. They are as follows:

-It works at client side so less response time.

-Secured

-It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

*Drawback of Applet

Plugin is required at client browser to execute applet.

Code with Output :

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class appp extends Applet implements ActionListener
{
	Button b;
	TextField tf;
	public void init()
	{
		tf=new TextField();
		tf.setBounds(30,40,150,20);
		b=new Button("Click");
		b.setBounds(80,150,60,50);
		add(b);
		add(tf);
		b.addActionListener(this);
		setLayout(null);
```

```
        }
        public void actionPerformed(ActionEvent e)
        {
                tf.setText("Welcome");
        }
}
// EventAppletEx.html<html><body><applet code="EventApplet.class"
width="300"
// height="300"></applet></body></html>
```

MULTIPLE CHOICE QUIZ

Submitted in partial fulfillment of the requirements of the degree
**BACHELOR OF ENGINEERING **
in
Computer Science and Engineering
(Artificial Intelligence & Machine Learning)

Skill base Lab course: Object Oriented Programming with Java

Sem - III
by
AIML08_Chikankar Prathamesh Shivaji
AIML03_Ansari Mohd Hanzala Mohd Imamuddin

Lokmanya Tilak College of Engineering
Koparkhairne, Navi Mumbai - 400 709
University of Mumbai
(AY 2021-22)

INDEX

Introduction:

In this project, we are creating a **MUITIPLE CHOICE QUIZ** with a collection of questions.

ABSTRACT:

User can enter / play a quiz.
There will be a limited number of questions and the user will get a credit score for each correct answer. The correct answer for each wrong answer will be shown by the program immediately after attempting a particular question.

The user will also get to know their score after attempting all the questions.

CODE:

```
import java.util.Scanner;

class Question
{
    String promt;
    String[] options;
    int answer;
    public Question(String promt, String[] options, int
answer)
    {
        this.promt = promt;
        this.options = options;
        this.answer = answer;
    }
}

public class QUIZ
{
    public static void main(String[] args)
    {
        String q1 = "What colour are apples?\n";
        String a1[] = { "(a)Red/Green", "(b)Orange",
"(c)Magenta" };
        String q2 = "What colour are bananas?\n";
        String a2[] = { "(a)Red/Green", "(b)Yellow",
"(c)Blue" };
        String q3 = "What colour are mangoes?\n";
        String a3[] = { "(a)Red/Green", "(b)Black",
"(c)Yellow/Green" };

        Question[] questions = { new Question(q1, a1, 1), new
Question(q2, a2, 2), new Question(q3, a3, 3) };
```

```
        takeTest(questions);
    }

    public static void takeTest(Question[] questions)
    {
        int score = 0;
        Scanner in = new Scanner(System.in);

        for (int i = 0; i < questions.length; i++)
        {
            Question x = questions[i];
            System.out.println(x.promt);
            for (int j = 0; j < x.options.length; j++)
            {
                System.out.println(x.options[j]);
            }
            System.out.println("Enter your answer: ");
            char ans = in.next().charAt(0);

            if (ans - 96 == x.answer)
            {
                score++;
            }
            else
            {
                System.out.println("Wrong answer!!");
                System.out.println("Correct Options = " +
x.options[x.answer - 1] + "\n");
            }
        }
        System.out.println("You got " + score + "/" +
questions.length);
    }
}
```

OUTPUT:

```
What colour are apples?

(a)red/green
(b)Orange
(c)Magenta
Enter your answer:
a
What colour are bananas?

(a)red/green
(b)Yellow
(c)Blue
Enter your answer:
a
Wrong answer!!
Correct Option = (b)Yellow

What colour are mangoes?

(a)red/green
(b)Black
(c)yellow/green
Enter your answer:
c
You got 2/3
```

Name : Prathamesh Chikankar

Roll No. And Year : AIMLD08 And SE

Branch And Div. : CSE (AI & ML) And D

Subject : Object oriented programming with JAVA [OOPs - J]

Course Code : CSL304

Year / Semester : SE / III

Sign : Prathu

Assignment No - 1

20/10/2021

AJMLD08

PAGE: 01
DATE: / /

LO1 To apply fundamental programming constructs

1. What is object oriented programming?

Ans:- Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).

Elaborate different OOPs concepts with suitable examples. [objects and class]

Ans:- a. Encapsulation:

Encapsulation is the process of wrapping up of data (properties) and behaviour (methods) of an object into a single unit; and the unit here is a class (or interface). It enables data hiding and expose only the relevant details required by the user. Consider the example of a linked list's getsize method. We might be now using a variable named size that is updated on every insert/delete operation. Later we might be decide to traverse the list and find size every time someone ask for size. But if some code was directly accessing the size variable, we would have to change all those code for this change. However if we were accessing the size variable through a getsize method, other code can still call that method and we can do our changes in that method.

b. Inheritance:

Inheritance describes the parent child relationship

between two classes. A class can get some of its characteristics from a parent class and then add more unique features of its own. For example, consider a vehicle parent class and a child class car. Vehicle class will have properties and functionalities common for all vehicles. Car will inherite those common properties from the vehicle class and then add properties which are specific to a car. Vehicle parent class is known as base class or super class. Car is known as derived class, child class or subclass.

c. polymorphism:

It refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities. polymorphism in Java are mainly of two types as overloading and overriding. Example- Consider a class shape with a draw() method. It can have subclass circle and square. An object of circle and square can be assigned to a shape reference type variable at runtime (e.g. shape S = new circle();). While executing draw() on the shape reference, it will

d. Abstraction:

Abstraction in object oriented programming refers to the ability to make a class abstract. Abstraction captures only those details about an object that are relevant to the current perspective, so that the programmer can focus on a few concepts at a time. Consider a real life example of a man driving

a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing the accelerator brackets will the speed is actually increasing, he does not know that the inner mechanism of the car or the implementation of accelerator, brakes, etc.. in the car. This is what abstraction is.

2. What is Java Virtual Machine?

Ans:- Java virtual machine also known as JVM; is an abstract machine. JVM
It is a specialization that provides suitable environment in which java bytecode can be executed with execution. JVMs are available for many hardware and software platforms. JVM is a platform dependent. It is a specialization whose working of JVM is specified but implementation provides is independent to choose algorithm. Its implementation has been provided by oracle and other companies.
Explain its significance
i. Ensure security
when java was initially develop, security was the priority. That is the reason why Java programms run separated in an enclosed area of the Java virtual machine which act as a protecting shield
ii. Cross platform
A program that is cross platform has the capability

to run succesfully on different types of hardware. Java is also a cross-platform language. i.e. it is possible to run a single piece of code written on a particular hardware that has JVM installed on it.

iii. Just-in time compiler

The java virtual machine comes with a Just-in time compiler which converts the java code into a low level machine language that can run as fast as the sequency applications. This compiled code goes into the cached memory of the browser, this means that one can use the code again without downloading it again and again compiling it.

* JVM performs various operations like:
  - → loadding the code
  - → verification of the code
  - → providing runtime environment
  - → Execution

3. Explain various features of Java.

The primary objective of Java programming language creation was to make it possible portable, simple and secure programming language. Apart from this, there are also some excellent features which plays an important role in the popularity of this language. The features of Java also known as Java buzz words.

The most important features of the Java language are as follow:

a. simple

Java is very easy to learn, and its syntax is simple,

clean and easy to understand. According to Sun microsystem, Java language is a simple programming language.

b. Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behaviour. Basic concepts of OOPs are: object, class, inheritance, polymorphism, Abstraction and Encapsulation.

c. Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while java is a write once, run anywhere language. A platform is the hardware or software environment in which programs runs. Java provides a software-based platform. It has two components Runtime Environment and API (Application programming Interface)

d. Secured

Java is well-known for its security, with java, we can develop virus-free system. Java is secured because
- No explicit pointer
- Java programs run inside a virtual machine sandbox
- classloader
- Bytecode verifier
- Security Manager

e. Robust

Robust mean strong. Java is Robust because

- It uses strong memory management
- There is a lack of pointers that avoids security problems.

f. Architecture-neutral

Java is architecture-neutral because there are no implementation dependent features, for example the size of primitive types is fixed.

g. portable

Java is portable because it facilitates you to carry the java bytecode to any platform. It don't require any implementation.

h. High performance:

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. Java is interpreted language that is why it is slower than compiled languages eg., C, C++, etc..

i. Distributed

Java is distributed because it facilitates users to create distributed applications in Java.

j. Multi-threaded

A thread like a separate program executing concurrently, we can write java programs that deal with many tasks at once by defining multiple threads.

k. Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are located on demand. It also supports functions from its native languages i.e. C, C++, etc..

AIML108

PAGE: 07
DATE: / /

LO2 To illustrate the concept of packages, classes & objects

1. Explain the concept of Java package with suitable example

It is a mechanism to encapsulate a group of classes, sub-packages and interfaces

It is used for

a) preventing name conflicts

b) providing control access

c) making searching, locating and usage of classes, interfaces, enumerations and annotations

d) They are considered as data-encapsulation or data-hiding.

when we reuse the existing classes from the packages as many times as we need.

```
package student;
public class myc
{
    public void getRollNo(int roll)
    {
        System.out.println(roll);
    }
}
```

Now we can use the myc by importing the package student.

packages are of two types basically

Uses defined packages and In Built packages

import pkg1 (.pkg2).(classname | *);

```
import studout.myc;
public class printRoll()
{
    public static void main (string args[])
    {
        int Roll 8:
        myc obj = new myc();
        obj.getRoll No. (Roll);
    }
}
```

Built-in packages:

1) java.lang: contains language support classes (eg class which defines primitive data types, math operations). This package is automatically imported
2) java.io: contains classed for supporting i/o operath.
3) java.util: contains utility classes which implement data structures like linked list, Dictionary and support; for Date/Time operations.
4) java.applet: contains classes for creating Applets.
5) java.awt: for implementing the components for graphical user interfaces (like button, ; menus etc.)
6) java.net: Contain classes for supporting networking opert

User-defined packages:

These are the packages that are defined by the user. first we create a directory myPackage(name should be same as the name of the package). Then creat the My class inside the directory with the first statment being the package name.

2. What are different types of Java Input/output functions?

Ans:-

JAVA output

In java you can simply use
System.out.println(); or
System.out.print(); or
System.out.printf();
to send output to standard outputs (screen)
Here,
- system is a class
- out is a public static field: It accepts output data.

Example to output a line.

```
class AssignmentOpesator
{
    public static void main(String[] args)
    {
        System.out.println("Java is interesting.");
    }
}
```

output

Java is intersting.

Here we have used the println() method to display the string.

i) print(): This method in java is used to display the text on the console. This text is passed as parameter to this method in the form of string. This method prints the text in the parameter

ii) println(): This method in java is also used to display the text on the console. It prints the text

on the console moves to the start of the next line at the console

iii) printf(): This is the easiest of all the methods as this is similar to the print in C. The printf() may take multiple arguments. This is used to format output in Java.

JAVA Input

Java provides different ways to get input from the user. In order to use the object of scanner, we need to import java.util.Scanner package.

Then we need to create an object of the scanner class. we can use the object to make input from the user.

```
// create an object of scanner
Scanner input = new Scanner(System.in);
// take input from the user
int number = input.nextInt();
```

Similarly, we can use nextLong(), nextFloat(), nextDouble() and next() methods to get long, float, double and string input respectively from the user. we have used to close the object method is the close()

1) Buffered Reader: It is a java class that reads texts from the input stream. It buffers the character so that if can get the efficient reading of characters, array. It includes inherits the reader class and makes code efficient.

ii) Scanner: Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc.. and strings.

3. what is constructor?

Ans:- A constructor in java is a special method that is used to initialize objects. The constructors is when is called an object of a class is created. It can be used to set initial values for object attributes. Explain constructor overloading with a suitable example.

important point to note while overloading a constructor is: when we don't implement any constructor, the java compiler inserts the default constructor into our code during compilation, however if we implement any constructor then compiler doesn't do it. See the below example

```
public class Demo
{
    private int rollNum;
    //we are not defining a non-arg constructor here
    Demo(int rnum)
    {
        rollNum = rollNum + rnum;
    }
    //Getting and setter methods
    public static void main(String args[])
    {
        //this statement would invoke no-arg constructor
        Demo obj = new Demo();
    }
}
```

output:

Exception in thread "main" java.lang.Error:
Unresolved compilation problem: The constructor Demo()
is undefined

simply, constructor overloading is a concept of having more than one constructor with different parameters list, in such a way so that each constructor performs a different tasks.

Sample:

```
public class Demo
{
    Demo()
    {
      ..
    }
    Demo(String s)
    {
      ..
    }
    Demo(int i)
    {
      ..
    }
    ..
}
```

Three overloaded constructors - They must have different parameters list

4. Compare the static and public methods in java

	static method	public methods
	'static' keyword is use to show class level property. In a class 'static keyword' show common property for the entire class.	'public' is a keyword to show that the class/method is available for entire package or accessed within anywhere.
1.	A static method is a method that belongs to a class, but it does not belong to an instance of that class and this mthd can be called without the Instance or object of that class	Every methods in java defaults to non-static i.e. public mthd without static keyword preceding it. non-static mthds can access any static mthd & static Variable also, without using the object of the class.
2.	the method can only access only static data members & static methods of another class or same class but cannot access public methods and Variables	In public method user the method can access static data members & static methods as well as non-static members and method of another class or same class
3.	static method uses compile time or early binding	public method uses runtime or dynamic binding
4.	static method cannot be overridden because of early binding	public method can be overridden because of runtime binding
5.	In static method, less memory is use for the execution.	In public method, much memory is used for the execution.

LO3 To elaborate the concept of strings, arrays and vectors.

1 Explain various functions used to manipulate strings.

Java programming language comes up with a huge list of string manipulate functions. we can manipulate string in different ways. Given below are the different ways for manipulating a string:

1. String concatenations: suppose we have two strings in separate. we want these two strings to be concatenated in one. for this we can simply use the traditional one (the + operator)

```
String str1 = "Hello";
String str2 = "India";
String finalResult = str1 + " " + str2;
```

2. Changing the given string's case: we can change the case of the string whenever required by using toLowerCase() and the toUpperCase() Java built-in functions.

3. Sub-string inside a string: we have a contain() method in java to deal with the checking of the string inside a string. we can use this function to check with some sequence of characters in a defined string. we can also use the substring() of java.

4. Cleaning the string with the unwanted character: for this, we can use the Java trim() function. This function can be used to remove the space from the begining and the end of the given string.

5. String reverse: string reverse is again the part of

String manipulation. This is all about getting the string from the last index to the first index. For example, we have "Hello world!", the reverse of this string will be "!dlrow olleH".

Syntax:

1. toLowerCase()

public String toLowerCase();

This function changes the case of the given string to the lower case. We can use this function at the time of string to handle the ignore case with the campareTo() function itself.

2. toUpperCase()

public string toLowerCase();

This function remains the same as toLowerCase(), but it changes the case to uppercase.

3. Substring()

String substring(int StartIndex, int LastIndex)

We can use the position from where we want to string in place of the StartIndex, and we can replace the lastIndex with the end index to where we want the string.

4. trim()

public string trim()

Again, this is a string manipulation function we can use to remove the unwanted space from a given string.

AJM1D08
PAGE: 16
DATE: / /

2. Write note on the following:

i) Arrays

- In Java all the arrays are dynamically allotted.
- A Java variable can also be declared like other variables with [] after data type
- The size of the array must be specified by an int value & not long or short
- Representation of an Array

30	20	18	293	23	24	97
0	1	2	3	4	5	6

The row 1 contains value & row 2 contains indices.

- Array length. 7 with first index 0
  last index 6
- Declaration of an array:
  type arrayname[];
  type[] arrayname;

ii) Vectors

- The vector implements a dynamically array that means the array can grow or shrunk the size of the vector as required.
- Can be accessed using indexes, just like array.
- Vectors are very similar to similar to array but vector is synchronized and has some legacy method that the collection of framework does not contain.
- • the interactions returned by vector class are fails-fast in case of concurrent modifications. it fails & throw the concurrent modification Exception.
- • We can perform multiple operation on vectors class in Java like adding elements, changing elements, removing & inserting vectors.

PAGE: a)
DATE: / /

Name: Prathamesh Chikankar

Roll No. And Year: AIML008 and SE

Branch And Div.: CSE-(AI&ML) And D

Subject: Object Oriented programming with JaVA (OOPs-J)

Course code: CSL304

Year/semester: SE/III

Sign: Prathu

ASSIGNMENT NO.-2

30/11/2021

LO4 To implement the concepts of inheritance and interfaces.

1. Explain the various types of inheritance in Java.

Ans:- Inheritance is a mechanism of driving a new class from an existing class.

> Constructor and private members do not get inherited in Java
> cyclic inheritance is not permitted
> assign parents reference to child objects
> Constructor get executed because of super() present in the constructor.

Java supports the following four types of inheritance:

- Single inheritance
- Multi-level inheritance
- Hierarchical inheritance
- Hybrid inheritance

Single Inheritance:

In single inheritance, a sub-class is derived from only one super class. It inherits the property & behaviour of a single-parent class.

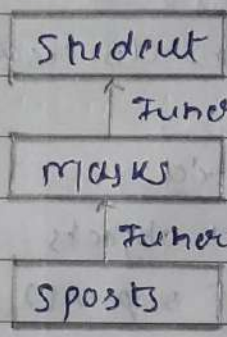
Employee ↑ Inherits Executive	super class sub class
Single Inheritance	

In the figure, Employee is a parent class and Executive is a child class. The executive class inherits all the properties of the Employee class.

Multi-level Inheritance:

In multi-level Inheritance, a class is derived from

a class which is also derived from another class is called multi-level inheritance. In simple words, we can say that a class that has more than one parent class is called multi-level Inheritance.

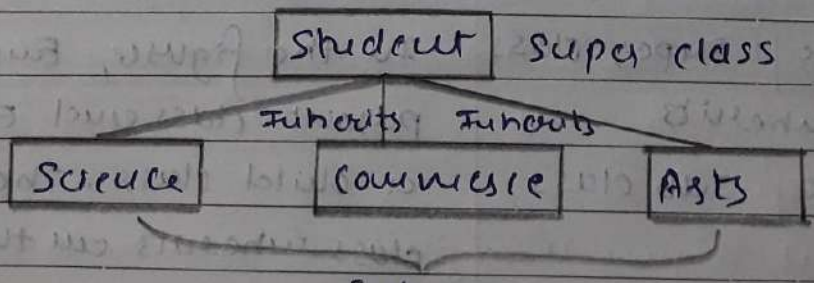


Multi-level Inheritance

In the figure, the class Marks inherit the members or methods of the class student. The class Sports inherits the members of the class Marks. Therefore, the Student class is the parent class of the class Marks and the class Marks is the parent class of the class Sports. Hence, the class Sports implicitly inherits the properties of student along with the class Marks.

Hierarchical Inheritance:

If a number of classes are derived from a single base class; it is called hierarchical inheritance.



Hierarchical Inheritance

In the figure, the classes science, commerce & Arts inherit a single parent class named Student.

2. Explain the super and final keywords in Java.

Ans:- • The Super keyword in Java is a reference variable which is used to refer immediate parent class object. Whenever you create the instance of sub-class, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

1. super can be used to refer immediate parent class instance variable.
   - We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields
2. super can be used to invoke parent class method
   - The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same methods as parent class. In other words it is used if method is overridden.
3. super is used to invoke parent class constructor
   - The super keyword can also be used to invoke the parent class constructor.
   As we know well that default constructor is provided by compiler automatically, if there is no constructor. But it also adds super() as the first statement.

• The final keyword in java is used to restrict the user. The java final keyword can be used in many context. final can be 1. variable
2. method
3. class

The final keyword can be applied with the variables a final variable that have no value it is called blank final variable or uninitialized final variable.
It can be initialized in the constructor only.
The blank final keyword can be static also which will be initialized in the static book only. We will have detailed learning of these. Below the basics final keyword.

1) Java final variable
If you make any variable as final, you cannot change the value of final variable (It'll be constant)

2) Java final method
If you make any methods as final, you cannot override it.

3) Java final class
If you make any class as final, you can't extend it.

Java final keyword
→ stop value change
→ stop method overriding
→ stop Inheritance

LO5 To implement the concept of exception handling and multithreading?

1 What do you mean by exception handling? Explain the use of try catch and finally in handling the exceptions.

Ans:- The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the applicatn can be Maintained.

An exception is an unexpected event that may occurs during the program's compile-time or run-time. It is capable enough to cause a severe potential threat & disrupt the entire flow of the program. library: java.lang.Exception.

Try, catch, finally blocks

To handle exceptions Java provides a try-catch block mechanism.

A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code.

Syntax:

```
try {
    // protected code
} catch (ExceptionName e1) {
    // catch block
}
```

When an exception raised inside a try block, instead of terminating the program JVM stores the excptn details in the exception stack and proceeds to the

catch block. A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in the try block it is passed to the catch block (or blocks) that follow it. If the type of exception that occured is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter. The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.

2. Difference b/w throw and throws

Ans :-

throw	throws
Throw is a keyword which is used to throw an exception in the program inside a function or inside a block / code.	Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.
Functionally throw is implemented as it is allowed to throw only single exception at a time i.e. we cannot throw multiple exception with throw keyboard.	On other hand we can declare multiple exceptions with throws keyword that could get thrown by the function where throws is used
With throw keyword we can propogate only unchecked exception i.e. checked exception cannot be propogated using throw.	On other hand we can declare multiple exception with throws keyword that could get thrown by the function where throws keyword is used.

throw	throws
Syntax wise throw Keyword is followed by the instance variable	On other hand Syntax wise throws Keyword is followed by exception class names.
In order to use throw Keyword we should know that throw Keyword is used within the method	On other hand throws Keyword is used with the method signature.

3. What is the significance of multi-threading. Explain the thread life cycle.

Ans:- Multithreading in java is a process of executing multiple threads simultaneously. A thread is a light-weight sub-process, the smallest unit of processing. multiprocessing and multithreading, both are used to achieve multitasking.

Significance:

i) It doesn't block the user because threads are independent and you can perform multiple operation at the same time.

ii) you can perform many operation together, so its saves time

iii) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

iv) Better use of cache storage by utilization of resources.

v) Decreased cost of maintenance & better use of CPU resource.

PTO

AIML108

PAGE: 08
DATE: / /

thread life cycle

In Java, a thread always exists in any one of the following states. These states are:

1. New 2. Active 3. Blocked/waiting 4. time waiting 5. terminated

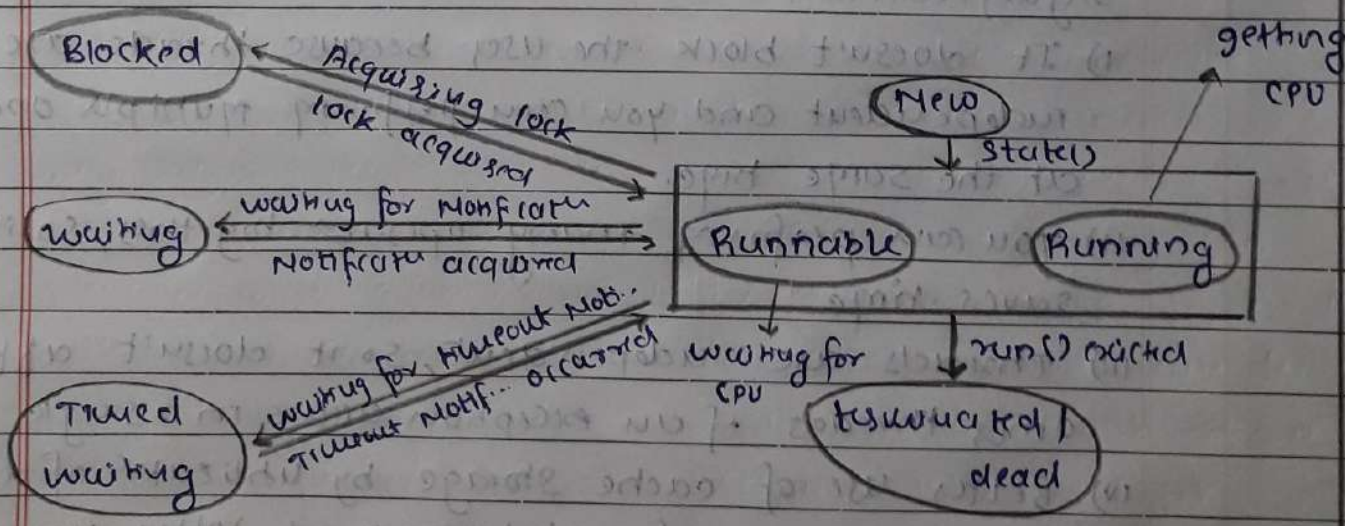
New: whenever a thread is created, it is always in the new state.

Active: when a thread invokes the start() method, it moves from the new state to the active state. The Active state contains two states within it one is runnable and other is running.

Blocked/waiting: whenever a thread is inactive for a span of time (not permanently) then, either the thread is in the blocked state or is in the waiting state.

Time waiting: sometimes, waiting for leads to starvation.

Terminated: thread reaches termination because whenever threads has finished its job or abnormal termination.



Life cycle of a Thread.

LO6 To develop GUI based applicatn.

1. what is Swing class in JAVA?

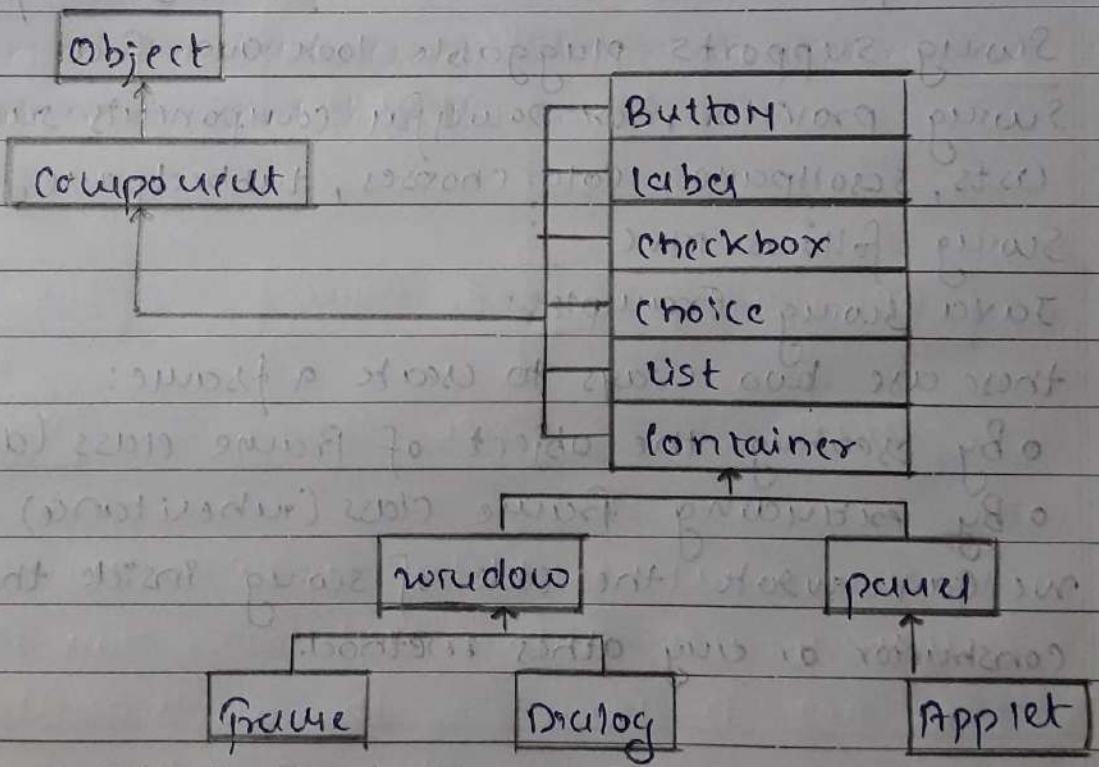
Ans:- Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applicatn. It is a part of the JFC (Java foundation class). It is built on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components.
It becomes easier to built applicatn since we already have GUI components like button, check box etc.. This is helpful because we do not have to start from scratch.
Swing supports pluggable look and feel.
Swing provides more powerful components such as tables, lists, scrollpanes, colorchooses, tabbedpane, etc..
Swing follows MVC.
Java Swing Examples:
there are two ways to create a frame:
- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

we can write the code of swing inside the main(), constructor or any other method.

2. Explain AWT, Draw Java AWT Hierarchy

Ans:- Java AWT (Abstract window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applicatn in Java.
Java AWT components are platform-dependent i.e. components are displayed accordingly to the view of operating system. AWT is heavy weight i.e. its

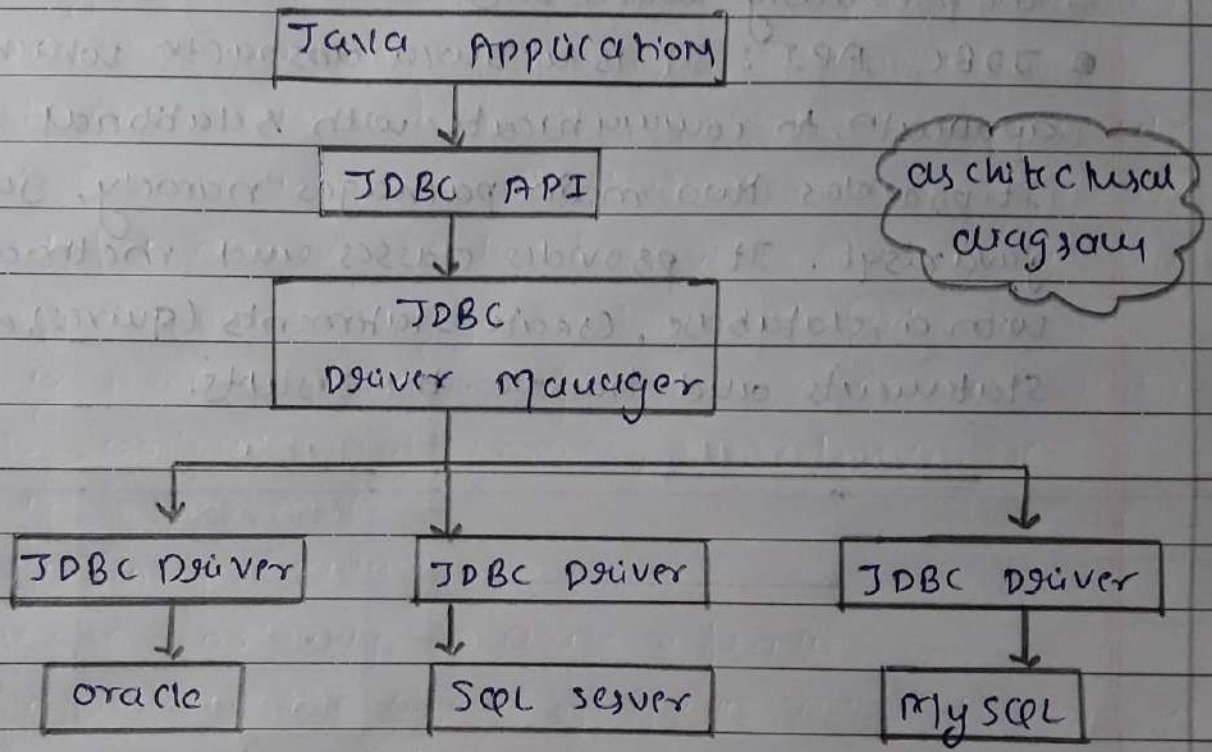
components are using the resources of underlying operating system (OS).
The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List, etc..
The AWT tutorial will help the users to understand Java GUI programming in simple and easy steps.
AWT platform is independent because AWT appearance will look like a windows application in window OS whereas it will look like a mac application in the MAC OS.



Java AWT Hierarchy

3. Explain JDBC with architectural diagram

Ans:- JDBC stands for Java Database connectivity. JDBC is a java API to connect and execute the query with the database. It is a part of Java SE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.



If you want to develop a Java application that communicates with a database, you should use JDBC API. A driver is the implementation of the said API; various vendors provide various drivers you need to use a suitable driver with respect to the database you need to communicate with. The driver manager loads the driver and manages the driver.

following are the componeub of JDBC

- JDBC Driver-Mauager : the driver manager class of the java.sql package meaus manages different types of JDBC drivers. This class loads the driver classes. In addition to this whenever a new connection establishes it choose and loads the suitable driver from the previously loaded ones.
- JDBC API : It is a Java abstractn which enables appucatn to communicate with relational database. It provides two main packages namely, java.sql and javax.sql. It provides classes and methods to connect with a database, create statements (quires), execute statements and handle the results.