THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONTENTS

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ASSIGNMENT NO. 1

AIM: To implement java operators.

LO MAPPED

LO 1

THEORY

Features of Java are as follows:

Simple: Java was designed to be easy for the professional programmer to learn and use effectively. Java inherits C/C++ syntax and many object oriented features of C++.

Object-Oriented: Although influenced by its predecessor, java was not designed to be source-code compatible with any other language This allowed the freedom to design with a blank state.

Robust: The ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in few key areas, to force you to find your mistakes early in program development. At the same time Java frees you from having to worry about many of the most common causes of programming error.

Multithreaded: Java was designed to meet the real world requirement of creating interactive networked programs. To accomplish this, Java supports multithreaded programming which

PROGRAM

```
import java.util.*;
public class pattern
{
    public static void main (String[] args)
    {
        String emp_name = "Employee";
        int emp_id = 90, basic = 50000;
        double da, hra, gross;
        da = basic * (0.2);
        hra = basic * (0.05);
        gross = basic + da + hra;
        System.out.println ("Employee Name: "+emp_name + " Empl
        +emp_id+"\nGross Salary: "+gross);
    }
}
```

Logical Operator:
AND &&
OR ||

Eg: a>b && d==10
Above statement is TRUE if both a>b and d=10

Conditional Operator
?:

CONCLUSION
In this program we learnt how to implement java operators in the program and get gross salary of employee.

24/8/22

allows you to write programs that do many things simultaneously.

5. Interpreted & High performance: Java enables the creation of cross platform programs by compiling in an immediate representation called Java byte code. This code can be interpreted on any system that provides Java virtual machine. Java was designed so that to perform well on lower CPU's.

6. Distributed: Java is designed for the distributed environment of the internet because it handles TCP/IP protocols. In fact accessing a resource using URL is not much different from accessing a file.

7. Dynamic: Java programs carry with them substantial amount of runtime type information that is used to verify and resolve access to object at runtime. This makes it possible to dynamically link code in a safe and expident manner.

OPERATORS IN JAVA

1. Arithmetic Operators [+, -, *, /, %,]

Integer arithmetic

14+4=18 14-4=10 14*4=56

14/4=3 14%4=2 -14%2=-2

. Relational Operators: [<, >, =, !, >=, <=]

4.5<=5 True/False 10+2!=12 True/False

20>=20 True/False

Relational operator evaluates to a true/false

ASSIGNMENT NO. 2

AIM: To implement control structures

LO MAPPED:

LO1 , LO2

THEORY

Java Virtual Machine (JVM)

A java virtual machine enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java byte code. JVM is detailed by a specification that formally describes what is required.

Java Runtime Environment (JRE)

It is the most common environment available on devices to run java programs. The source java code gets compiled and converted to Java byte code on any platform and to run on platform you need JRE. JRE consists of development technologies, user interface toolkits, Integration library, other base libraries, lang and util base libraries, JVM.

PROGRAM

```
import java.util.*;
public class pattern
{
    public static void main(String[] args)
    {
        int i,j,k,m,n;
        System.out.println("Enter the number of rows: ");
        Scanner sc=new Scanner(System.in);
        n=sc.nextInt();
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=(n-i);j++)
            {
                System.out.print(" ");
            }
            for(k=1;k<=i;k++)
            {
                System.out.print(k);
            }
            for(m=i-1;m>0;m--)
            {
                System.out.print(m);
            }
            System.out.println();
        }
    }
}
```

Java Development Kit (JDK)

It is a distribution of Java technology. It implements the Java language specifications and JVM and provides standard edition of Java application programming interface (API). It allows developers to create Java programs that can be executed and run by JVM and JRE. It is a syntax package you download inorder to create Java based applications

Control Structure Syntax

1. if (conditional)

Syntax:

```
if(condition)
{
    statements
}
else
{
    statements
}
```

2. for loop

Syntax:

```
for(initialization; condition; increment/decrement)
{
    Statements
}
```

```
import java.util.*;
public class mathfunc
{
    public static void main (String[] args)
    {
        int choice;
        System.out.println("1.Factorial\n2.Fibonacci\n3.Armstrong\n4.Sum of
          \nEnter your choice: ");
        Scanner sc=new Scanner(System.in);
        choice = sc.nextInt();
        Switch(choice)
        {
            case 1: int i,n,fact=1;
            System.out.println("Enter a number: ");
            Scanner a=new Scanner(System.in);
            n=a.nextInt();
            for(i=1;i<=n;i++)
            {
                fact=fact*i;
            }
            System.out.println("The factorial of "+n+" is "+fact);
            break;

            case 2: int j,n1=0,n2=1,n3,num;
                System.out.println("Enter number of terms: ");
                Scanner b=new Scanner(System.in);
                num=b.nextInt();
                System.out.println(n1+"\n"+n2);
                for(j=1;j<=num-2;j++)
                {
                    n3=n1+n2;
                    System.out.println(n3);
                    n1=n2;
                    n2=n3;
                }
                break;
```

3. Entry controlled loop

while loop

Syntax:

```
while (condition)
{
    Statements
}
```

4. Exit controlled loop

do-while loop

Syntax:

```
do
{
    Statements
} while (condition);
```

5. Switch case

Syntax:

```
Switch (expression)
{
    case value 1 : Statement 1;
                   break;
    case value 2 : Statement 2;
                   break;
    :
    :
    case value N : Statement N;
                   break;
    default : default statement;
}
```

```
case 3: int sum, temp, rem, k;
        System.out.println("The Armstrong numbers are\n");
        for(k=100; k<=500; k++)
        {
            temp=k;
            sum=0;
            while(temp>0)
            {
                rem=temp%10;
                sum+=(rem*rem*rem);
                temp=temp/10;
            }
            if(sum==k)
            {
                System.out.println(k+"\n");
            }
        }
        break;
case 4: int p, result=0, r;
        System.out.println("Enter a number: ");
        Scanner c=new Scanner(System.in);
        p=c.nextInt();
        while(p>0)
        {
            r=p%10;
            result+=r;
            p=p/10;
        }
        System.out.println("The Sum of digits is: "+result);
        break;
default : System.out.println("Invalid choice");
    }
}
}
```

6. Break

Syntax:

```
for(intialization; condition; increment)
{
    if(condition)
    {
        break;
    }
}
```

7. Continue

Syntax:

```
for(intialization; condition; increment)
{
    if(condition)
    {
        continue;
    }
    //Statements
}
```

CONCLUSION

In this program we learnt how to implement control structures like if-else statement, switch case, for loop and while loop and break statement.

24/8/22

PROGRAM

```
import java.util.*;
import java.util.io*;
class Bank{
    public String nameOfDepositor;
    public String accNumber;
    public String accType;
    public double balanceAmount;
    public void assignValues(String nameOfDepositor,String accType,double balance
    {
        this.nameOfDepositor= nameOfDepositor;
        this.accType = accType;
        this.balanceAmount = balanceAmount;
        Random random = new Random();
        this.accNumber = random.nextInt(1000000);
        System.out.println("Your new account number:"+accNumber);
    }
    public void depositAmount(double amount){
      if(accNumber==0)
        System.out.println("You need to create an account");
      else{
        balanceAmount+=amount;
        System.out.println("Amount deposited successfully");
      }
    }
    public void withdrawAmount(double amount){
      if(accNumber==0)
        System.out.println("Create an account first");
      else if(balanceAmount>amount){
        balanceAmount+=amount;
        System.out.println("Amount credited successfully");
      }
      else
        System.out.println("Balance amount:"+balanceAmount);
    } public void getInput(){System.out.println("1.Open Account\n2.Deposit\n3.Withdr
    class Main{
        public static void main(String[] args){
            System.out.println("Welcome to TSEC Bank");
            Bank newAccount = new Bank();
```

ASSIGNMENT. NO. 3

AIM

Write a program to demonstrate classes and object concept in Java.

LO MAPPING : LO1, LO2

THEORY

Everything in java is associated with classes and objects, along with its attributes and methods. For example, in real life, a car is an object. The car has attributes, such as weight and colour and the methods such as drive and brake. A class is like an object ~~constructor~~ created constructor for creating objects. Class is not real world entity. It is just a template from which objects are created.

In object oriented programming, objects are things you think about first in designing a program and they are also the units of code that are eventually derived from the process. Object is an instance of class.

Syntax of class

```
<access specifiers> class class_name
{
    // member variable
    // class methods
}
```

```
Scanner sc=new Scanner(System.in);
boolean process =true;
int continueState=0;
while (continueState==0){
  newAccount.getInput();
  int currentProcess=scan.nextInt();
  if (currentProcess==1){
     System.out.println("Enter name: ");
     String nameOfDepositor=sc.next();
     System.out.print("Enter acc Type ");
     String accType=sc.nextInt();
     System.out.println("Enter opening balance:");
     double balanceAmount=sc.nextDouble();
     newAccount.assignValues(nameOfDepositor, accType, balanceAmount);
  }
  else if(currentProcess==2){
     System.out.println("Enter amount to be deposited");
     newAccount.depositAmount(sc.nextDouble());
  }
  else if(currentProcess==3){
     System.out.print("Enter amount to be withdrawn");
     newAccount.withdrawAmount(sc.nextDouble()); {
  else if(currentProcess==4){
     newAccount.displayDetails()
  }
  else if(currentProcess==5){
     newAccount continueState=1;
     System.out.print("Press 0 to continue...");
     continueState=scanner.Int();
  }
}
}
```

OUTPUT

1. Open Account
2. Deposit
3. Withdraw
4. Display

Enter your choice:

Syntax of object

<class-name>.<object-name>

CONCLUSION

The concept of classes and object is observed and implemented in a banking application program.

7/9/22

PROGRAM

```
class Rectangle{
  private int l,b;
  public Rectangle(){
     l=b=10;
  }
  public Rectangle(int x){
     l=b=x;
  }
  public Rectangle(int x,int y){
     l=x;
     b=y;
  }    public void area(){System.out.println("Area="+l*b);
class main{
  public static void main(String[] args){
     Rectangle r1=new Rectangle();
     Rectangle r2=new Rectangle(5);
     Rectangle r3=new Rectangle(3,3);
     r1.area();
     r2.area();
     r3.area();
  }
}
```

OUTPUT

Area = 100

Area = 25

Area = 9

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ASSIGNMENT NO 4

AIM:
Write a program to demonstrate the usage of constructor in javer
(a) Constructor overloading
(b) Method overloading

LO MAPPED : LO 2

THEORY

1. Construction Overloading
Constructor in java are used to initialise an object in the class of constructor. They have the same name as class they are in. We can have more than one constructor in a class, this is called constructor overloading.

Overloaded constructors have the same name. They differ by number and type of arguements. This way compiler knows ~~which~~ which constructor to call.

2. Method Overloading
Methods are functions of a class which perform operations on the data members of class. Multiple declaration of methods with same name is known as method overloading. The overloading is also similar to constructor overloading, they differs by number & type of arguments.

PROGRAM

```
import java.util.*;
class Volume{
  private int x;
  public float volume(float l);
    return(l*l*l);
  }
  public float volume(float r, float h){
    return(3.142*r*r*h);
  }
  public float volume(float l, float b, float h){
    return(l*b*h);
  }
  class Main{
     public static void main(String[] args){
        Volume a=new Volume();
        System.out.println("Volume of cube="+a.volume(10));
        System.out.println("Volume of cylinder="+a.volume(10,10));
        System.out.println("Volume of cuboid="+a.volume(10,10,10));
     }
  }
```

OUTPUT

Volume of cube = 1000.0
Volume of cylinder = 3140.0
Volume of cuboid = 1000.0

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONCLUSION

Method and constructor overloading is implemented in the program using default and parameterized constructors and methods

7/9/22

PROGRAM

```
import java.util.*;
class Temp {
   Temp() {
      this(5);
      System.out.println("The Default Constructor");
   }
   Temp(int x) {
      this(5,15);
      System.out.println(x);
   }
   Temp(int x, int y) {
      System.out.println(x*y);
   }
   public static void main(String[] args) {
      new Temp();
   }
}
```

OUTPUT

```
75
5
The Default Constructor.
```

ASSIGNMENT NO 5

AIM : To implement constructor chaining.

LO MAPPED: LO1, LO2

THEORY:

In Java, constructor chaining is a sequence of invoking constructors upon initializing an object. It is used when we want to invoke a number of constructors one after another by using only an instance.

Constructor chaining can be done in two ways, Within same class and From base class.

→ Within same class: It can be done using this() keyword for constructors in same class.

→ From base class: It can be done using super() keyword to call the constructor in same class.

Constructor chaining occurs through inheritence. A sub class constructor's task is to call super class constructor first. This ensures that creation of subclass' object starts with the initialization of the data members of the superclass. There could be any number of classes in the inheritence chain. Every constructor calls up the chain till the class at the top is reached.

```
import java.util.*;
class Base {
    String name;
    Base() {
        this(" ");
        System.out.println("No arguement constructor of "+"base class");
    }
    Base(String name) {
        this.name = name;
        System.out.println("Calling parameterized constructor "+"of base");
    }
}
class Derived extends Base {
    Derived() {
        System.out.println("Calling parameterized constructor "+" of derived");
    }
    Derived(String name){
        super(name);
        System.out.println("Calling parameterized "+"constructor of derived");
    }
    public static void main(String[] args) {
        Derived obj = new Derived("test");
    }
}
```

OUTPUT

Calling parameterized constructor of base.
Calling parameterized constructor of derived.

Rules of Constructor Chaining

1. The this() keyword should always be the first line of the constructor.
2. There should be atleast one constructor without the this() keyword.
3. Constructor chaining can be achieved in any order.

Note: Similar to constructor chaining in same class super() should be the first line of the constructor as per super class constructor are invoked before the sub-class constructor.

CONCLUSION

This experiment helps us to understand constructor chaining and use of keywords this and super in java programs.

12/10/22

ASSIGNMENT NO. 6

AIM

Write a program to demonstrate
(a) Matrix multiplication & addition
(b) Array sorting technique using 1 dimensional array.

LO MAPPED : LO1, LO2

THEORY

An array is a collection of items of same data type; storage at contiguous memory locations. When declaring an array needs to be specified. After sorting data in array each element is indexed. The first elements index is 0 & 1 is index of next elements. If there are n elements, last one is n-1.

→ 1-Dimensional Array

A list of items can be given one variable name using only one subscript & such a variable is called an on dimensional array

→ 2-Dimensional Array

It can be defined as an array of arrays. When declaring this type of array the number of rows & columns needs to be specified. These type of arrays are also indexed according to rows and columns.

PROGRAM

```
import java.util.*;
public class Matrices{
public static void main(String[] args){
  int choice
  int matrix[][]={{0,0,0},{0,0,0},{0,0,0}};
  matrix1 [][] = {{1,2,3},{1,2,3},{1,2,3}},
  matrix2 [][] = {{1,2,3},{1,2,3},{1,2,3}};
  System.out.println("Enter a choice\n1. Addition
   of matrix\n2. Multiplication of matrix");
  choice = sc.nextInt();
  switch(choice){
    case 1: matAdd(matrix1,matrix2,matrix);
    System.out.println("Sum is");
    display(matrix);
    break;
    case 2: matMul(matrix1,matrix2,matrix);
    System.out.println("Multiplication is");
    display(matrix);
    break;
    default: System.out.println("Invalid");
  }
}
public static void display(int mat[][]){
  for(int i=0; i<3; i++){
    for(int j=0;j<3;j++)
      System.out.print(mat[i][j]+" ");
    System.out.println();
  }
}
public static void matAdd(int mat1[][],
                  int mat2[][],int mat[][])
{
   for(int i=0; i<3; i++){
     for(int j=0;j<3;j++)
       mat[i][j]=mat1[i][j]+mat2[i][j];
   }
}
public static void matMul(int mat1[][],
                  int mat2[][],int mat[][])
{
  for(int i=0; i<3; i++){
    for(int j=0; j<3; j++){
      for(int k=0; k<3; k++)
        mat[i][j]+=mat1[i][j]*mat2[i][j];
    }
  }
}
```

OUTPUT

```
Enter a choice:
1. Addition of matrix
2. Multiplication of matrix
1
Sum is
2 4 6
2 4 6
2 4 6
```

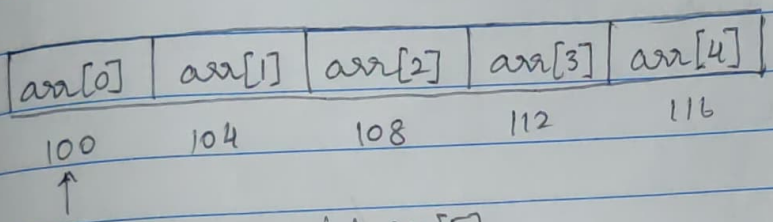
DECLARATION OF ARRAY

Arrays can be declared in various ways in different languages.
In C, array is declared as follows:

```
        name
         ↓
int array[10] = {35,33,42,10,14,19,27,44,26,31}
 ↑        ↑                    Elements
datatype  size
```

→ The array's length is 10, which means we can store 10 elements
→ The index of array starts with 0.
→ Each element in the array can be accessed via its index.

MEMORY ALLOCATION OF ARRAY



DISPLAYING AN ARRAY

An array can be displayed using for loop() having count equal to length of the array

For example:

```
#include<stdio.h>
void main(){
    int i, int arr[5]={18,30,15,70,12};
    printf("Elements of the array are:\n");
    for(i=0;i<5;i++)
        printf("Arr[%d] = %d", i, arr[i]);
}
```

PROGRAM

```
import java.util.*;
public class Arr {
public static void main(String[] args){
 int len;
 Scanner sc= new Scanner (System.in);
 System.out.println("Enter length of the array: ");
 len= sc.nextInt();
 int arr[] = new int [len];
 System.out.println("Enter elements in array: ");
 inputArr (arr , len);
 System.out.println("Array entered is ");
 display(arr);
 bubbleSort(arr);
 System.out.println("Array after sorting is ");
 display(arr);
}
public static void bubbleSort(int arr1[]){
  int n = arr1.length;
  for(int i=0; i<n-1 ; i++){
    for(int j=0;j<n-i-1 ;j++) {
      if(arr1[j] > arr1[j+1]){
         int temp = arr1[j];
         arr1[j]=arr1[j+1];
         arr[j+1] =temp;
    }
  }
}
public static void display(int arr1[]){
  for(int i : arr1){
    System.out.print(i+" "); } }
public static void inputArr(int arr1[], int l){
   Scanner sc=new Scanner(System.in);
   for (int i=0; i<l; i++)
      arr1[i]=sc.nextInt();
}
```

CONCLUSION

Multiplication and addition of matrices was implemented using two dimensional arrays. Sorting of one dimensional array was performed using bubble sorting.

21/9/22

PROGRAM:

```
import java.util.*;
public class FriendList {
    public static void main(String[] args){
        Vector<String> in= new Vector<>();
        in.add("Mike");
        in.add("Harvey");
        in.add("Patrick");
        in.add("Conrad");
        in.add("Rick");
        System.out.println("List of friends:" +in);
        System.out.println("Remove friend at index 4." +in.remove(4));
        System.out.println("Remove friend list after removing at index 4"+in);
        in.insertElementAt("Daryl",2);
        System.out.println("Element in vector after insertion at index 2="+in);
        in.add("Cho");
        System.out.println("Friend List after adding at end=" +in);
    }
}
```

OUTPUT

List of Friends: [Mike, Harvey, Patrick, Conrad, Rick]

Remove friend at index 4: Rick

Remove friend list after removing at index 4: [Mike, Harvey, Patrick, Conrad]

Element in vector after insertion at index 2 = [Mike, Harvey, Daryl, Patrick, Conrad]

Friend List after adding at end = [Mike, Harvey, Daryl, Patrick, Conrad, Cho]

ASSIGNMENT NO. 7

AIM: To write a program to demonstrate (a) Vector (b) Strings

LO MAPPED: LO1, LO2

THEORY:

Vector is like a dynamic array which can grow or shrink in size. Unlike array, we can store n-number of elements in it as there is no size limit. It is found in the java.util package and implements the list interface.

There are various Java Vector Methods. Some of them are mentioned below

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 all of the elements in the specified component to the end of this vector.
4) Capacity() : It is used to get current capacity of the vector.
5) Clear() : It is used to delete all of the elements from the vector.
6) Clone() : It returns a clone of the vector.
7) equals() : It is used to compare the specified object with the vector for equality.
8) get() : It is used to get an element at the specified position in the vector.

```
import java.util.*;
public class String {
    public static void main(String[] args) {
        String[] arr = new String[] {"Mike", "Harvey", "Patrick", "Rick", "Daryl",
                                     "Conrad", "Nick", "Choi", "Jessica", "Louis"};

        System.out.println("Before...");
        System.out.println(Arrays.toString(arr));
        System.out.println();
        for(int i=0; i<arr.length; i++) {
            arr[i] = arr[i].substring(3);
        System.out.println("After removing 3 characters: ");
        System.out.println(Array.toString(arr));
        System.out.println();
        System.out.println("After sorting in alphabetical order");
        sort(arr);
        System.out.println(Arrays.toString(arr));

    }
}
```

OUTPUT

Before...
Mike Harvey Patrick Rick Daryl Conrad Nick Choi Jessica Louis
After removing 3 characters:
e vey rick k yl rad k i sica uis
After sorting in alphabetical order
e i k k rad rick sica uis vey yl

String

In Java, String is an object that represents a sequence of characters. The java.lang.String class is used to create a string object. Java String class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length, replace(), compareTo(), intern(), substring(), etc.

There are various java string methods

1. int length() : It returns string length
2. String concat(String str) : It concatenates the specified string.
3. String Intern() : It returns interned string.
4. int indexOf(Int ch) : It returns the specified char value index.
5. boolean isEmpty() : It checks if string is empty.
6. bodean equals(Object another) : It checks the equality of string with given object.

CONCLUSION

In this program, we understood and implemented the concept of vectors and strings.

13/10/22

PROGRAM:

```
import java.util.*;
class Book {
    String name;
    String author;
    void display() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Book name : ");
        name = sc.nextLine();
        System.out.println("Enter author name: ");
        author = sc.nextLine();
        System.out.println("\nBook Name: "+name);
        System.out.println("Book author : "+author);
    }
}
class ReferenceBook extends Book {
    void display() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Reference Book name: ");
        name = sc.nextLine();
        System.out.print("Enter Reference Book author name : ");
        author = sc.nextLine();
        System.out.println("\nReference Book name: "+name);
        System.out.println("Reference Book Author : "+author);
    }
}
class Magazine extends Book {
    void display() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Magazine name: ");
        name = sc.nextLine();
        System.out.print("Enter Magazine author : ");
        author = sc.nextLine();
        System.out.println("\nMagazine name : "+name);
        System.out.println("Magazine author : "+author);
    }
}
public class BookStore {
    public static void main(String[] args) {
        Book b = new Book();
        b.display
        ReferenceBook ref = new ReferenceBook();
        ref.display();
        Magazine mag = new Magazine();
        mag.display();
    }
}
```

OUTPUT:

```
Enter Book name: abc
Enter Author name : xyz

Book name : abc
Author name : xyz

Enter Reference Book name: def
Enter Reference Book author : xyz

Reference Book name: def
Reference Book Author : xyz

Enter Magazine name: nasd
Enter Magazine author : abc

Magazine name : nasd
Magazine author : abc
```

ASSIGNMENT NO. 8

AIM: (a) Create a class Book and difine a display method to display book information.

(b) Create an interface vehicle and classes bicycle, car, bike, etc having common functionalities.

LO MAPPED: LO1, LO3

THEORY:

1. Inheritence in Java

Inheritence in Java is a mechanism in which one object acquires all the properties and behaviours of a parent object. It is an important part of OOPS (Object Oriented programming system).

The idea behind inheritence in Java is that one can create new classes that are built upon existing classes. When one inherits from an existing class, you can reuse methods and fields of the parent class. Moreover, new methods and fields can be added in the current class.

Inheritence represents IS-A relationship which is also known as parent-child relationship.

Syntax of Java Inheritence

```
class Subclass extends Superclass
{
    // methods and fields
}
```

The extends keyword indicates that the new class is derived from an existing class.

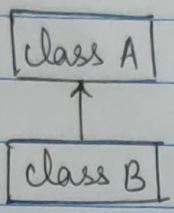
PROGRAM:

```
import java.io.*;
interface Vehicle {
    void changeGear(int a);
    Void speedUP (int a);
    void applyBrakes(int a);
}
class Bicycle implements Vehicle{
    int speed, gear;
    public void changeGear(int newGear) {
        gear=newGear;
    }
    public void speedUP(int increment){
        speed=speed+increment;
    }
    public void applyBrakes(int decrement){
        speed=speed-decrement;
    }
    public void printStates (){
        System.out.println("Speed : "+speed+"km/h Gear: "+gear);
    }
}
class Bike implements Vehicle {
    int speed, gear;
    public void changeGear(int newGear){ gear=newGear; }
    public void speedUp (int increment){ speed+= increment; }
    public void applyBrakes(int decrement){ speed-= decrement; }
    public void printStates(){
        System.out.println("Speed: "+speed+"km/h Gear: "+gear);
    }
}
class Car implements Vehicle {
    int speed, gear;
    public void changeGear(int newGear){ gear=newGear; }
    public void speedUp (int increment){ speed+= increment; }
    public void applyBrakes (int decrement){ speed-= decrement; }
    public void printStates() {
        System.out.println("Speed: "+speed+"km/h Gear : "+gear);
    }
}
```

Types of Inheritence

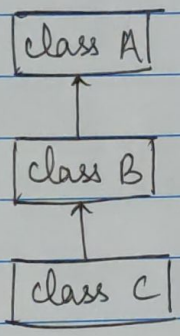
Single Inheritence

When a class inherits another class, it is known as a single inheritence. In the following structure class B inherits class A.



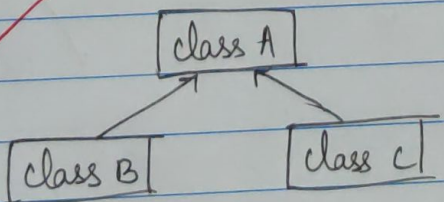
Multilevel Inheritence

When there is a chain of inheritence, it is known as multilevel inheritence. In the following structure class C inherits class B which again inherits from class A.



3. Hierarchical Inheritence

When two or more classes inherit a single class, it is known as hierarchical inheritence. In the following structure both classes B and C inherit from class A



```
class VehicleDemo{
    public static void main(String[] args){
        Bicycle bicycle = new Bicycle();
        bicycle.changeGear(2);
        bicycle.speedUp(25);
        bicycle.applyBrakes(10);
        System.out.println("Bicycle present State: ");
        bicycle.printStates();

        Bike bike= new Bike();
        bike.changeGear(2);
        bike.speedUp(45);
        bike.applyBrakes(3);
        System.out.println("Bike present State: ");
        bike.printStates();

        Car car = new Car();
        car.changeGear(3);
        car.speedUp(60);
        car.applyBrakes(5);
        System.out.println("Car present State: ");
        car.printStates();
    }
}
```

OUTPUT:

```
Bicycle present State:
Speed: 15km/h Gear: 2
Bike present State:
Speed: 42km/h Gear: 2
Car present State:
Speed: 55km/h Gear: 3
```

Interface in Java

An interface in java programming language is defined as an abstract type used to specify the behaviour of a class and contains static constants and abstract methods. A Java interface is a blueprint of a class

The interface in Java is a mechanism to achieve abstraction. There can only be abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritence. In other words, interfaces can have abstract methods and variables Java interface also represents the IS-A relationship.

Syntax of Interface

```
interface {
    // declare constant fields
    // declare methods that are abstract
    // by default.
}
```

CONCLUSION:

In this experiment we understood and implemented the concept of inheritence and interfaces in Java. We also learnt about the types of inheritence in java.

28/9/22

PROGRAM

```
class Banking {
    void rateOfInterest() {
        System.out.println("RBI Rate of Interest = 5.90%.");
    }
}
class PNB extends Banking {
    void rateOfInterest() {
        System.out.println("PNB Rate of Interest = 3.45%.");
    }
}
class SBI extends Banking {
    void rateOfInterest() {
        System.out.println("SBI Rate of Interest = 8.55%");
    }
}
class BOI extends Banking {
    void rateOfInterest() {
        System.out.println("BOI Rate of Interest = 4.30%.");
    }
}
public class MethodOverriding {
    public static void main(String[] args) {
        Banking bank = new BOI();
        bank.rateOfInterest();
    }
}
```

OUTPUT

BOI Rate of Interest = 4.30%

ASSIGNMENT NO. 9

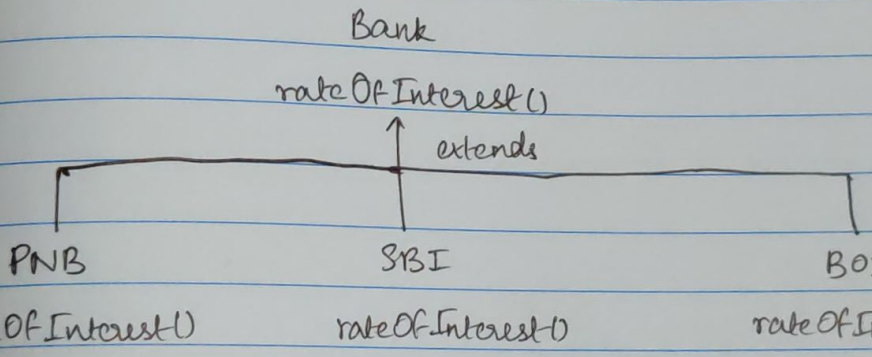
AIM: Write a program to demonstrate Polymorphism.

LO MAPPED: LO1, LO3

THEORY:

If subclass (child class) has the same method a declared in the parent class, it is known as method overriding in Java.

In other words, if a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.



CONCLUSION

In this experiment, we understood and implemented the concept of method overiding. The method of base class is overriden in the derived classes.

12/10/22

PROGRAM

```
class Thread1 extends Thread {
    public void run() {
        System.out.print("a");
    }
}
class Thread2 extends Thread {
    public void run() {
        System.out.print("b ");
    }
}
class Thread3 extends Thread {
    public void run() {
        for(int i=0; i<100; i++)
            System.out.print(i+" ");
    }
}
public class MultiThreading {
    public static void main(String[] args) {
        Thread1 th1 = new Thread1();
        Thread2 th2 = new Thread2();
        Thread th3 = new Thread3();
        th1.start();
        th2.start();
        th3.start();
    }
}
```

OUTPUT

a a a .... b b b ... a a a a .... 1 2 3 4 5 ..... 98 99

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO.10

AIM: Write a program to demonstrate Multithreading in Java

LO MAPPED: LO1, LO3, LO4

THEORY:

In Java, Multithreading refers to a process of executing two or more threads simultaneously for maximum utilization of the CPU. A ~~thread~~ thread in Java is a lightweight process requiring fewer resources to create and share the process resources.

LIFE CYCLE OF A THREAD

There are five states a thread has to go through in its life cycle This life cycle is controlled by JVM. These states are

1. New

In this state, a new thread begins its life cycle. This is also called a born thread. The thread is in the new state if you create an instance of Thread class but before the invocation of the start() method

2. Runnable

A thread becomes runnable after a newly born thread is started. In this state, a thread would be executing its task.

3. Running

When the thread scheduler selects the thread then, that thread would be in a running state.

4. Non-Runnable (Blocked)

The thread is still alive in this state, but currently, it is not eligible to run

5. Terminated

A thread is terminated due to the following reasons:

→ Either its run() method exists normally, i.e., the thread's code has executed the program.

→ Due to some unusual errors like segmentation fault or an unhandled exception.

CONCLUSION

In this experiment, we understood and implemented the concept of multithreading. We used run() method to display the required output and used start() method to use multithreading to ~~run~~call the required method.

12/10/22

PROGRAM

```
public class JavaException {
    public static void main(String[] args) {
        try {
            int data = 100/0;
        }
        catch(ArithmeticException e) {
            System.out.println(e);
        }
        // rest of the code
        System.out.println("rest of the code....");
    }
}
```

OUTPUT

```
Exception in thread main java.lang.ArithmeticException: / by zero
rest of the code....
```

ASSIGNMENT NO. 11

AIM: Write a program to demonstrate Exception handling in Java.

LO MAPPED: LO1, LO3, LO4

THEORY:

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. An exception is an event that disrupts the normal flow of the program. It is an object which is thrown at run time. Some of the run time exceptions are

(a) ClassNotFoundException
(b) IOException
(c) SQLException
(d) Remote Exception

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

CONCLUSION

In this program, we understood and implemented the concept of exception handling.

PROGRAM:

```
import java.io.File;
import java.io.IOException;
class CreateFile {
    public static void main(String[] args) {
    try {
        File fo = new File("D:FileOperationExample.txt");
        if (fo.createNewFile()) {
            System.out.println("File" + fo.getName() + " is created
                                                    successfully");
        }
        else {
            System.out.println("File is already existing in directory.");
        }
    }
    catch (IOException exception) {
        System.out.println("An unexpected error occured.");
        exception.printStackTrace();
    }
  }
}
```

OUTPUT

File FileOperationExample.txt is created successfully.

ASSIGNMENT NO. 12

AIM: Write a program to demonstrate File Handling in Java.

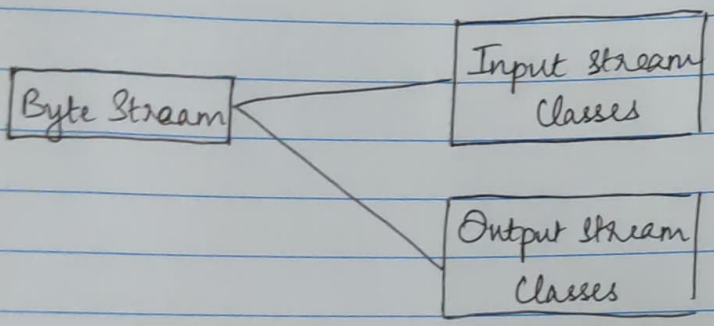
LO MAPPED: LO1, LO3, LO4

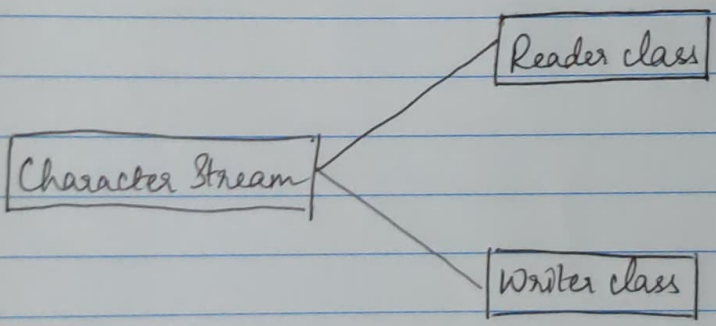
THEORY:

In Java, a file is an abstract data type. A named location used to store related information is known as a File. There are several File operations like creating a new File, getting information about File, writing into a file, reading from a file.

Stream

A series of data is referred to as a stream. In Java, Stream is classified into two types i.e Byte Stream and Character Stream.





Byte Stream

Byte stream is mainly involved with byte data. A file handling process with a byte stream is a process in which an input is provided and executed with the byte data.

Character Stream

Character stream is mainly involved with character data. A file handling process with a character stream is a process in which an input is provided with the character data.

CONCLUSION

In this program, we understood and implemented ~~with~~ the concept of file handling in Java.

13/10/22

PROGRAM

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class myinfo {
    Static JTextfield name_txt;
    Static JTextfield sname_txt;
    Static JTextfield cname_txt;
    Static JTextfield pincode_txt)
    Static JButton Submit_btn;
    Static JTextArea output_txtArea;
    public static void main(String[] args){
        JFrame frame=new JFrame("MY INFORMATION");
        frame.setVisible(true);
        frame.setBounds(700,700,709700);
        frame.SetDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c=frame.getContentPane();
        c.SetLayout(null);
        c.SetBackground(Color.white);
        Font f=new Font("Arial",Font.BOLD,32);
        JLabel heading_lbl=new JLabel();
        heading_lbl.setBounds(250,70,400,40);
        heading_lbl.setText("MY INFORMATION");
        heading_lbl.setFont(f);
        font f1=new Font("Arial",Font.Bold,20);
        JLabel name_lbl=new JLabel("Name:");
        name_lbl.setBounds(50,150,200,30);
        name_txt=new JTextfield();
        name_txt.setBounds(180,150,250,30);
        JLabel sname_lbl=new JLabel("Street:");
        sname_lbl.setBounds(50,230,200,30);
        Sname_txt=new JTextfield();
        Sname_txt.SetBounds(180,230,250,30);
        JLabel cname_lbl=new JLabel("City:");
        cname_lbl.SetBounds(50,310,200,30);
        Cname_txt=new JTextfield();
        Cname_txt.SetBounds(180,310,250,30);
        Cursor cur=new Cursor(Cursor.HAND_CURSOR);
        JLabel pincode_lbl=new JLabel("Pincode:");
        pincode_lbl.SetBounds(50,390,200,30);
        pincode_txt=new JTextfield();
        pincode_txt.SetBounds(180,390,250,30);
        Submit_btn=new JButton("My_Info");
        Submit_btn.SetBounds(300,450,160,40);
        submit_btn.SetCursor(cur);
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO. 13

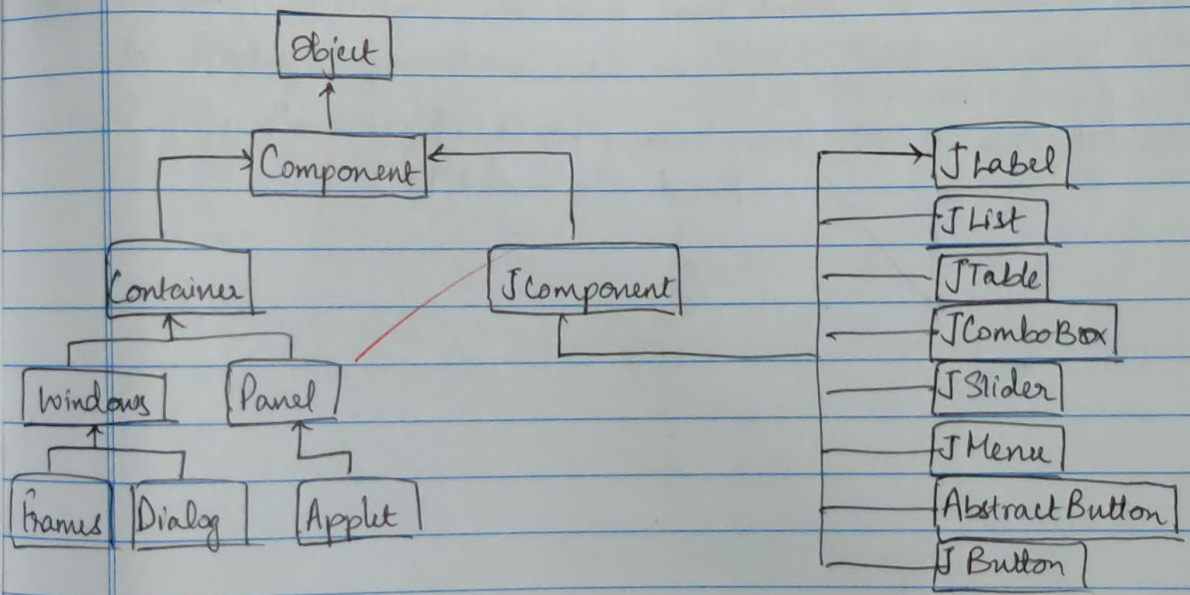
AIM: Write a program to create a window with four text fields for the name, street, city and pincode with suitable labels. Also windows contains a button MyInfo. When the user types the name, street, city and pincode and then clicks the button, then types details must appear in Arial font with size 32, Italics.

LO MAPPED: LO1, LO4 and LO5

THEORY:

Java Swing will be used for this assignment. Java Swing tutorial is a part of Java foundation classes that is used to create window based applications. It is ~~both~~ built on top of AWT API & entirely written in java. The java.swing package provides classes for java Swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser, etc.

Hierarchy of Java Swing



```
submit_btn.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent event){
        submit_action(event);
    }
});
Font f2=new Font("Arial", Font.ITALIC,35);
Font f3=new Font("Arial",Font.PLAIN,20);
output_txtArea=new JTextArea();
output_txtArea.setBounds(500,200,500,500);
name_lbl.setFont(f1);
sname_lbl.setFont(f1);
pincode_lbl.setFont(f1);
cname_lbl.setFont(f1);
name_txt.setFont(f3);
sname_txt.setFont(f3);
cname_txt.setFont(f3);
pincode_txt.setFont(f3);
submit_btn.setFont(f2);
c.add(heading_lbl);
c.add(name_lbl);
c.add(sname_lbl);
c.add(cname_lbl);
c.add(pincode_lbl);
c.add(cname_txt);
c.add(sname_txt);
c.add(name_txt);
c.add(pincode_txt);
c.add(submit_btn);
c.add(output_txtArea);
}
public static void submit_action(ActionEvent event)
{
    String name=name_txt.getText();
    String cname=cname_txt.getText();
    String sname=sname_txt.getText();
    String pincode=pincode_txt.getText();
    output_txtArea.setText("Name: "+name+"\nStreet: "+sname+"\nCity: "+cname+
                           "\nPincode: "+pincode+"\n");

}
}
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Methods of Component class

1. add(Component) - Add a component on another component.
2. SetSize(int width, int height) - Sets size of the component.
3. setLayout(Layout Manager) - Sets the layout manager for the component.
4. setVisible(boolean b) - sets visibility of the component. It is by default false.
5. JLabel - A JLabel is a object component for placing text in a container.
6. JList - A JList component represents the user with the scrolling list of text items.
7. JComboBox - A JComboBox is the part of AWT that presents the user with a show up of menu choices.
8. JSlider - This class lets the user graphically select a value by ~~scrolling~~ sliding a knob within bounded interval.
9. JMenu - The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits JMenuItem.
10. AbstractButton - This is an abstract base class for all buttons like JButton, JToggleButton, JCheckBox, etc.

CONCLUSION

In this experiment, we used Java swing to create a form to display informations such as name, address, etc.

8/12/22

PROGRAM:

```
import java.awt.*;
class MyCalc extends Frame {
    Frame f;
    Label l1;
    Button b1,b2,b3,b4,b5,b6,b7,b8,b9,b0,badd,bsub,bmult,bdiv,bmod,bcalc,bclr,
     bpts,bneg,bback;
    double rel;
    double num1,num2,check;
    MyCalc() { f=new Frame("TSEC Calculator");
               l1=new Label();
    l1.setBounds(50,50,260,60);
    b1=new Button("1");
    b1.setBounds(50,340,50,50);
    b2=new Button("2");
    b2.setBounds(120,340,50,50);
    b3=new Button("3");
    b3.setBounds(190,340,50,50);
    b4=new Button("4");
    b4.setBounds(50,270,50,50);
    b5=new Button("5");
    b5.setBounds(120,270,50,50);
    b6=new Button("6");
    b6.setBounds(190,270,50,50);
    b7=new Button("7");
    b7.setBounds(50,200,50,50);
    b8=new Button("8");
    b8.setBounds(120,200,50,50);
    b9=new Button("9");
    b9.setBounds(190,200,50,50);
    b0=new Button("0");
    b0.setBounds(120,410,50,50);
    bneg=new Button("+/-");
    bneg.setBounds(50,410,50,50);
    bpts=new Button(".");
    bpts.setBounds(190,410,50,50);
    bback=new Button("back");
    bback.setBounds(120,130,50,50);
    badd=new Button("+");
    badd.setBounds(260,340,50,50);
    bsub=new Button("-");
    bsub.setBounds(260,270,50,50);
    bmult=new Button("*");
    bmult.setBounds(260,200,50,50);
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

ASSIGNMENT NO. 14

AIM: Write a program to create Scientific Calculator.

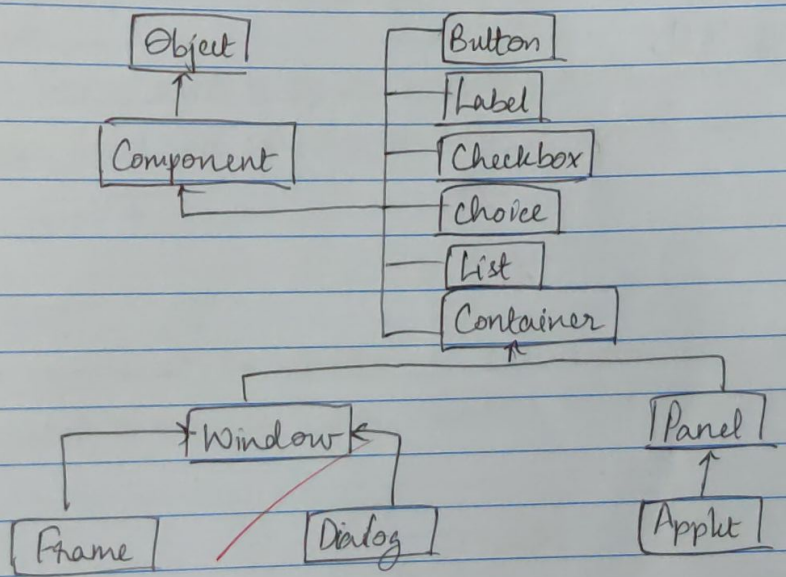
LO MAPPED: LO1, LO4 and LO5.

THEORY:

AWT

AWT Stands for Abstract Window Toolkit, which is an application programming interface (API) for creating Graphical User Interface (GUI) in Java which allows java programmers to develop window-based appn.

AWT provides various components like button, label, checkbox, etc. used as objects inside a java program. AWT components use the resources of the operating system. The class for AWT are provided by the java.awt package for various AWT component. The following is the hierarchy.



```
bdiv=new Button("/");
bdiv.setBounds(260,130,50,50);
bmod=new Button("%");
bmod.setBounds(190,130,50,50);
bcalc=new Button("=");
bcalc.setBounds(245,410,65,50);
bclr=new Button("CE");
bclr.setBounds(50,130,65,50);

f.add(l);
f.add(b1); f.add(b2);
f.add(b3); f.add(b4);
f.add(b5); f.add(b6); f.add(b7);
f.add(b8); f.add(b9); f.add(b0);
f.add(badd); f.add(bsub); f.add(bmult);
f.add(bdiv); f.add(bmod); f.add(bcalc);
f.add(bclr); f.add(bpts); b.add(bneg);
f.add(bback);
f.setSize(360,500);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String[] args) {
    new MyCalc();
}
}
```

1. Component - The component class stands at the top of the AWT hierarchy which is an abstract class that contains all the properties of the component visible on the screen. The component object contains information about the currently selected foreground and background color. It also has information about the currently selected text color.
2. Container - The container is a ~~compound cop~~ component that contains other components. However, it ~~doesn't contain~~ a subclass of the component class.
3. Panel - Panel can be defined as a container that can be used to hold other components. However, it doesn't contain the title bar, menu bar or border.
4. Window - Window can be defined as a container that doesn't contain my border or menu bar. It creates a top-level view. However, we must have a frame, dialog or another window for creating a window.
5. Frame - The frame is a subclass of window. It can be defined as a container with components like buttons, textfield, label, etc. In other words, AWT applications are mostly created using frame container.
6. Button - This is used to create a button on the user interface with a specified label. We can design code to execute some logic on the most.
7. List

CONCLUSION

In this experiment, we created a calculator using awt in Java. We created buttons to enter numbers and arithmetic operations.

12/10/22