



KCG
COLLEGE OF TECHNOLOGY

CS8383 OBJECT ORIENTED PROGRAMMIN G LAB

-A controlled copy (R2017)

Dr S Sankar
Associate Professor
Department of Computer Science and
Engineering

AIM Develop a Java application to generate Electricity bill. Create a class with the following members: Consumer no., consumer name, previous month reading, current month reading, and type of EB connection (i.e domestic or commercial). Compute the bill amount using the following tariff.

If the type of the EB connection is domestic, calculate the amount to be paid as follows:

- First 100 units – Rs. 1 per unit
- 101-200 units – Rs. 2.50 per unit
- 201 -500 units – Rs. 4 per unit
- > 501 units – Rs. 6 per unit

If the type of the EB connection is commercial, calculate the amount to be paid as follows:

- First 100 units – Rs. 2 per unit
- 101-200 units – Rs. 4.50 per unit
- 201 -500 units – Rs. 6 per unit
- > 501 units – Rs. 7 per unit

SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

Instance methods are methods which require an object of its class to be created before it can be called. To invoke a instance method, we have to create an Object of the class in within which it defined. Instance method(s) belong to the Object of the class not to the class i.e. they can be called after creating the Object of the class. Every individual Object created from the class has its own copy of the instance method(s) of that class. They can be overridden since they are resolved using dynamic binding at run time.

Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

'this' is a keyword in Java used to refer instance/object variable of a class. It is often used inside the constructor to differentiate both arguments names of a constructor and instance variable names.

The switch case statement is similar to the else-if ladder as it provides multiple branching or multi-conditional processing. But, the basic difference between switch case and else if ladder is that the switch statement tests the value of variable or expression against a series of different cases or values, until a match is found. Then, the block of code within the match case is executed. If there are no matches found, the optional default case is executed. The switch case is more compact than lot of nested else if. So, switch is considered to be more readable. When you want to solve multiple option type problems, for example: Menu like program, where one value is associated with each option and you need to choose only one at a time, then, switch statement is used.

A class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class –

- Declaration – A variable declaration with a variable name with an object type.
- Instantiation – 'new' keyword is used to create the object.
- Initialization – 'new' keyword is followed by a call to a constructor. This call initializes the new object.

'main' method signature is **public static void main(String[] args)** which is an entry point in Java programs . In general, only one main method is permitted for a class. A Java program can contain more than one "main" method, as long as their return types or arguments are different as shown below.

```
class sample{
    public static void main(String[] args){
        //code
    }
    public static void main(int[] args){
        //code
    }
}
```

But the other "main" methods cannot be used as the function's entry points. The only one which is **public static void main(String[] args)** counted as the entry point for the program.

ALGORITHM

Step1: Create a class called 'ElectricityBill' with main method

Step2: Create necessary instance variables

Step3: Create a constructor with necessary arguments and initialize instance variables

Step4: Create an instance method called 'calculate'. Find total units consumed and calculate bill amount by implementing the given conditions

Step5: Create an instance method called 'print' to print the necessary details

Step 6: Create required instance of class and call constructor with necessary values

Step7: Invoke calculate method using instance

Step8: Invoke print method using instance

PROGRAM

```
public class ElectricityBill {
    //Instance variables declaration
    int consumerNo;
    String consumerName;
    int previousMonthReading;
    int currentMonthReading;
    int connectionType;
    float billamount=0.0f;
    int unitConsumed=0;

    //Constructor to initialize members
    public ElectricityBill(int consumerNo, String consumerName, int previousMonthReading, int
currentMonthReading, int connectionType){
        this.consumerNo=consumerNo;
        this.consumerName=consumerName;
        this.previousMonthReading=previousMonthReading;
        this.currentMonthReading=currentMonthReading;
        this.connectionType=connectionType;
    }

    //Instance Method to calculate bill amount
    void calculate(){
        if(previousMonthReading>currentMonthReading){
            System.out.println("Error in current month reading");
        }
    }
}
```

```

}else{
    unitConsumed=currentMonthReading-previousMonthReading;
}

switch(connectionType){
case 1:
    if (unitConsumed>=0 && unitConsumed<=100){
        billamount=unitConsumed*1;
    }
    else if(unitConsumed>100 && unitConsumed<=200){
        billamount=unitConsumed*2.5f;
    }
    else if(unitConsumed>200 && unitConsumed<=500){
        billamount=unitConsumed*4;
    }
    else{
        billamount=unitConsumed*6;
    }
    break;

case 2:
    if (unitConsumed>=0 && unitConsumed<=100){
        billamount=unitConsumed*2;
    }
    else if(unitConsumed>100 && unitConsumed<=200){
        billamount=unitConsumed*4.5f;
    }
    else if(unitConsumed>200 && unitConsumed<=500){
        billamount=unitConsumed*6;
    }
    else{
        billamount=unitConsumed*7;
    }
    break;
}

}

//Instance method to print details
void print(){
    System.out.println("Consumer Number: "+consumerNo);
    System.out.println("Consumer Name: "+consumerName);
    System.out.println("Previous Month Reading: "+previousMonthReading);
    System.out.println("Current Month Reading: "+currentMonthReading);
    System.out.println("Total Units Consumed: "+unitConsumed);
    System.out.println("Connection Type: "+connectionType);
    System.out.println("Total Units Consumed: "+unitConsumed);
    System.out.println("Bill amount: "+billamount);
}

```

```
public static void main(String arg[]){

    //Object or Instance creation and calling constructor method
    ElectricityBill eb=new ElectricityBill(123, "Raj", 200, 500, 2);

    //Calling method using object
    eb.calculate();

    //Calling method using object
    eb.print();

    //Object creation and calling constructor method
    ElectricityBill eb1=new ElectricityBill(124, "Kumar", 230, 400, 1);

    //Calling method using object
    eb1.calculate();

    //Calling method using object
    eb1.print();
}
}
```

OUTPUT

Consumer Number: 123
Consumer Name: Raj
Previous Month Reading: 200
Current Month Reading: 500
Total Units Consumed: 300
Connection Type: 2
Total Units Consumed: 300
Bill amount: 1800.0

Consumer Number: 124
Consumer Name: Kumar
Previous Month Reading: 230
Current Month Reading: 400
Total Units Consumed: 170
Connection Type: 1
Total Units Consumed: 170
Bill amount: 425.0

CONCLUSION

By executing this program successfully, I learned

- How to create a class and its main method
- How to create instance variables
- How to create constructor with arguments
- How to create instance methods and call them
- How to create instance of a class or call constructor with values
- The use of 'this' keyword and when to use switch control structure and others.

VIVA QUESTIONS

1. Which one of these classes is super class of every class in Java?

- a) String class b) Object class c) Abstract class d) ArrayList class

2. What is true about constructor?

- a) It can contain return type b) It can take any number of parameters
c) It can have any non access modifiers d) Constructor cannot throw exception

3. Which of the following is a valid declaration of an object of class Box?

- a) Box obj = new Box(); b) Box obj = new Box; c) obj = new Box();
d) new Box obj;

4. What is the output of this program?

```
class box
{
    int width;
    int height;
    int length;
}
class mainclass
{
    public static void main(String args[])
    {
        box obj = new box();
        obj.width = 10;
        obj.height = 2;
        obj.length = 10;
        int y = obj.width * obj.height * obj.length;
        System.out.print(y);
    }
}
```

- a) 12 b) 200 c) 400 d) 100

5. What is the output of this program?

```
class box
{
    int width;
    int height;
    int length;
}
class mainclass
{
    public static void main(String args[])
    {
        box obj1 = new box();
        box obj2 = new box();
        obj1.height = 1;
        obj1.length = 2;
```

```
obj1.width = 1;
obj2 = obj1;
System.out.println(obj2.height);
}
```

a) 1

b) 2

c) Runtime error

d) Garbage value

AIM Develop a java application to implement currency converter (Dollar to INR, EURO to INR, Yen to INR and vice versa), distance converter (meter to KM, miles to KM and vice versa), time converter (hours to minutes, seconds and vice versa) using packages.

SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

A package as the name suggests is a pack (group) of classes, interfaces and other packages. In java we use packages to organize our classes and interfaces. We have two types of packages in Java: built-in packages and the packages we can create (also known as user defined package). There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc. **Advantage of Java Package:**

- Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- Java package provides access protection.
- Java package removes naming collision.

The **package keyword** is used to create a package in java. If you use package.* then all the classes and interfaces of this package will be accessible but not sub packages. The import keyword is used to make the classes and interface of another package accessible to the current package. If you import package.classname then only declared class of this package will be accessible. Package inside the package is called the **subpackage**. It should be created **to categorize the package further**.

ALGORITHM

Step1: Create a package called 'javalab' to include a class called 'MyConverter'

Step2: Create an another package in the name of 'Converter' in javalab package

Step3: Create a class called 'CurrencyConverter' in the package 'Converter' wherein mention name of the package it presents

Step3.1: Create required public instance methods to perform different currency conversion

Step4: Create a class called 'DistanceConverter' in the package 'Converter' wherein mention name of the package it presents

Step4.1: Create required public instance methods to perform different distance conversion

Step5: Create a class called 'TimeConverter' in the package 'Converter' wherein mention name of the package it presents

Step5.1: Create required public instance methods to perform different time conversion

Step6: Inside 'javalab' package, create a class called 'MyConverter' with main method wherein specify name of the package it presents and import all classes present in 'Converter' package

Step6.1: Create an instance method to display a main menu which lists available converters

Step6.2: Create an instance method to display a menu to list of currency conversions available wherein based on the user choice perform appropriate currency conversion operation by calling suitable method presents in 'CurrencyConverter' class

Step6.3: Create an instance method to display a menu to list various distance conversions available wherein based on the user choice perform appropriate distance conversion operation by calling suitable method presents in 'DistanceConverter' class

Step6.4: Create an instance method to display a menu to list of time conversions available wherein based on the user choice perform appropriate time conversion operation by calling suitable method presents in 'TimeConverter' class

Step6.5: Create an instance method to print the required details

Step6.6: Inside main method of 'MyConverter' class, create an instance of it and call 'displayMainMenu' using it.

PROGRAM

```
//Creating a package called Converter in side javalab package  
package javalab.Converter;
```

```
//Creating a class CurrencyCoverter in 'Converter' package  
public class CurrencyConverter {
```

```
    //Method to convert dollar to inr  
    public double dollarToInr(double dollar){  
        return dollar*60.0;  
    }
```

```
    //Method to convert inr to dollar  
    public double inrToDollar(double inr){  
        return inr/60.0;  
    }
```

```
    //Method to convert yen to inr  
    public double yenToInr(double yen){  
        return yen*40.0;  
    }
```

```
    //Method to convert inr to yen  
    public double inrToYen(double inr){  
        return inr/40.0;  
    }
```

```
    //Method to convert euro to inr  
    public double euroToInr(double euro){  
        return euro*80.0;  
    }
```

```
    //Method to convert inr to euros  
    public double inrToEuro(double inr){  
        return inr/80.0;  
    }
```

```
}
```

```
//Create Converter package in javalab package  
package javalab.Converter;
```

```
//Creating a class DistanceConverter in Converter package  
public class DistanceConverter {
```

```
    //Method to convert meter to km  
    public double meterToKm(double meter){  
        return meter*0.001;  
    }
```

```
    //Method to convert km to meter
```

```

public double kmToMeter(double km){
    return km/0.001;
}
//Method to convert miles to km
public double milesToKm(double miles){
    return miles*1.609344;
}
//Method to convert km to miles
public double kmToMiles(double km){
    return km/1.609344;
}
}
//Create a Converter package in javalab package
package javalab.Converter;

//Creating a class TimeConverter in Converter package
public class TimeConverter {
    //Method to convert hours to minutes
    public double hoursToMinutes(double hours){
        return hours*60;
    }
    //Method to convert minutes to hours
    public double minutesToHours(double minutes){
        return minutes/60;
    }
    //Method to convert hours to seconds
    public double hoursToSeconds(double hours){
        return hours*3600;
    }
    //Method to convert seconds to hours
    public double secondsToHours(double seconds){
        return seconds/3600;
    }
}

//Create a package called javalab to hold a class called 'MyConverter'
package javalab;

//import classes present in Converter package
import javalab.Converter.CurrencyConverter;
import javalab.Converter.DistanceConverter;
import javalab.Converter.TimeConverter;

import java.util.Scanner;

//Creating a class called 'MyConverter'
public class MyConverter {

```

```

//Method to display mainmenu
void displayMainMenu() {
    int yesorno, choice = 0;
    Scanner in = new Scanner(System.in);

    do {
        System.out.println(" Main Menu \n 1) Currency Converter \n 2) Distance Converter \n 3) Time
Converter \n 4) Exit \n");
        System.out.print("Enter your choice[1-4]: ");
        choice = in.nextInt();
        switch (choice) {
            case 1:
                displayCurrencyConverterMenu(); //method call
                break;
            case 2:
                displayDistanceConverterMenu(); //method call
                break;
            case 3:
                displayTimeConverterMenu(); //method call
                break;
            case 4:
                System.exit(0);
            default:
                System.out.println("Enter correct choice");
        }
        System.out.print("Want to continue MainMenu? [Press 1 for Yes / 0 for No]: ");
        yesorno = in.nextInt();
    } while (yesorno == 1);
}

//Method to display currency converter menu
void displayCurrencyConverterMenu() {
    double beforeconversion, afterconversion = 0.0;
    int yesorno, choice = 0;
    Scanner in = new Scanner(System.in);
    do {

        System.out.println(" 1) Dollar to INR \n 2) INR to Dollar \n 3) Euro to INR \n 4) INR to Euro \n 5) Yen to
INR \n 6) INR to Yen \n");
        System.out.print("Enter your choice[1-6]: ");
        choice = in.nextInt();

        System.out.print("Enter the amount to be converted:");
    }
}

```

```

beforeconversion = in.nextDouble();
switch (choice) {
    case 1:
        afterconversion = new CurrencyConverter().dollarToInr(beforeconversion);
        print(beforeconversion, afterconversion, "US$", "INR");
        break;
    case 2:
        afterconversion = new CurrencyConverter().inrToDollar(beforeconversion);
        print(beforeconversion, afterconversion, "INR", "US$");
        break;
    case 3:
        afterconversion = new CurrencyConverter().euroToInr(beforeconversion);
        print(beforeconversion, afterconversion, "EURO", "INR");
        break;
    case 4:
        afterconversion = new CurrencyConverter().inrToEuro(beforeconversion);
        print(beforeconversion, afterconversion, "INR", "EURO");
        break;
    case 5:
        afterconversion = new CurrencyConverter().yenToInr(beforeconversion);
        print(beforeconversion, afterconversion, "YEN", "INR");
        break;
    case 6:
        afterconversion = new CurrencyConverter().inrToYen(beforeconversion);
        print(beforeconversion, afterconversion, "INR", "YEN");
        break;
    default:
        System.out.println("Enter correct choice");
}

System.out.print("Want to continue? [Press 1 for Yes / 0 for No]: ");
yesorno = in.nextInt();
} while (yesorno == 1);
}

//Method to display distence converter menu
void displayDistanceConverterMenu() {
    double beforeconversion, afterconversion = 0.0;
    int yesorno, choice = 0;
    Scanner in = new Scanner(System.in);
    do {
        System.out.println(" 1) Meter to KM \n 2) KM to Meter \n 3) Miles to KM \n 4) KM to Miles \n");
        System.out.print("Enter your choice[1-4]: ");
        choice = in.nextInt();
    }
}

```

```

System.out.print("Enter the distance to be converted:");
beforeconversion = in.nextDouble();

switch (choice) {
    case 1:
        afterconversion = new DistanceConverter().meterToKm(beforeconversion);
        print(beforeconversion, afterconversion, "METER", "KM");
        break;
    case 2:
        afterconversion = new DistanceConverter().kmToMeter(beforeconversion);
        print(beforeconversion, afterconversion, "KM", "METER");
        break;
    case 3:
        afterconversion = new DistanceConverter().milesToKm(beforeconversion);
        print(beforeconversion, afterconversion, "MILES", "KM");
        break;
    case 4:
        afterconversion = new DistanceConverter().kmToMiles(beforeconversion);
        print(beforeconversion, afterconversion, "KM", "MILES");
        break;
    default:
        System.out.println("Enter correct choice");
}
System.out.print("Want to continue? [Press 1 for Yes / 0 for No]: ");
yesorno = in.nextInt();
} while (yesorno == 1);
}

//Method to display time converter menu
void displayTimeConverterMenu() {
    double beforeconversion, afterconversion = 0.0;
    int yesorno, choice = 0;
    Scanner in = new Scanner(System.in);
    do {
        System.out.println(" 1) Hours to Minutes \n 2) Minutes to Hours\n 3) Hours to Seconds \n 4) Seconds
to Hours \n");
        System.out.print("Enter your choice[1-4]: ");
        choice = in.nextInt();

        System.out.println("Enter the time to be converted:");
        beforeconversion = in.nextDouble();
        switch (choice) {
            case 1:

```

```

        afterconversion = new TimeConverter().hoursToMinutes(beforeconversion);
        print(beforeconversion, afterconversion, "HOURS", "MINUTES");
        break;
    case 2:
        afterconversion = new TimeConverter().minutesToHours(beforeconversion);
        print(beforeconversion, afterconversion, "MINUTES", "HOURS");
        break;
    case 3:
        afterconversion = new TimeConverter().hoursToSeconds(beforeconversion);
        print(beforeconversion, afterconversion, "HOURS", "SECONDS");
        break;
    case 4:
        afterconversion = new TimeConverter().secondsToHours(beforeconversion);
        print(beforeconversion, afterconversion, "SECONDS", "HOURS");
        break;
    default:
        System.out.println("Enter correct choice");
    }
    System.out.print("Want to continue? [Press 1 for Yes / 0 for No]: ");
    yesorno = in.nextInt();
} while (yesorno == 1);
}

//Method to display result
void print(double valuebefore, double valueafter, String unitsbefore, String unitsafter) {
    System.out.println(valuebefore + " " + unitsbefore + " = " + valueafter + " " + unitsafter);
}

public static void main(String arg[]) {
    MyConverter im = new MyConverter(); //Object creation
    im.displayMainMenu(); //Calling mainmenu
}}

```

OUTPUT

Main Menu

- 1) Currency Converter
- 2) Distance Converter
- 3) Time Converter
- 4) Exit

Enter your choice[1-4]: 1

- 1) Dollar to INR
- 2) INR to Dollar
- 3) Euro to INR
- 4) INR to Euro

- 5) Yen to INR
- 6) INR to Yen

Enter your choice[1-6]: 1

Enter the amount to be converted: 10

10.0 US\$ = 600.0 INR

Want to continue? [Press 1 for Yes / 0 for No]: 1

- 1) Dollar to INR
- 2) INR to Dollar
- 3) Euro to INR
- 4) INR to Euro
- 5) Yen to INR
- 6) INR to Yen

Enter your choice[1-6]: 2

Enter the amount to be converted:

600

600.0 INR = 10.0 US\$

Want to continue? [Press 1 for Yes / 0 for No]: 0

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 0

CONCLUSION

By compiling and executing this program successfully, I learned what is package, the use of package and how to create package and use it.

VIVA QUESTIONS

1. Which of these access specifiers can be used for a class so that it's members can be accessed by a different class in the different package?

- a) Public
- b) Protected
- c) Private
- d) No Modifier

2. Which of the following is correct way of importing an entire package 'pkg'?

- a) import pkg.
- b) Import pkg.
- c) import pkg.*
- d) Import pkg*

3. Which of the following is incorrect statement about packages?

- a) Package defines a namespace in which classes are stored
- b) A package can contain other package within it
- c) Java uses file system directories to store packages
- d) A package can be renamed without renaming the directory in which the classes are stored

4. Which of the following package stores all the standard java classes?

- a) lang
- b) java
- c) util
- d) java.packages

5. What is the output of this program?

```
package pkg;
class output
{
    public static void main(String args[])
    {
        StringBuffer s1 = new StringBuffer("Hello");
        s1.setCharAt(1, x);
        System.out.println(s1);
    }
}
```

- a) xello b) xxxxx c) Hxlllo d) Hexlo

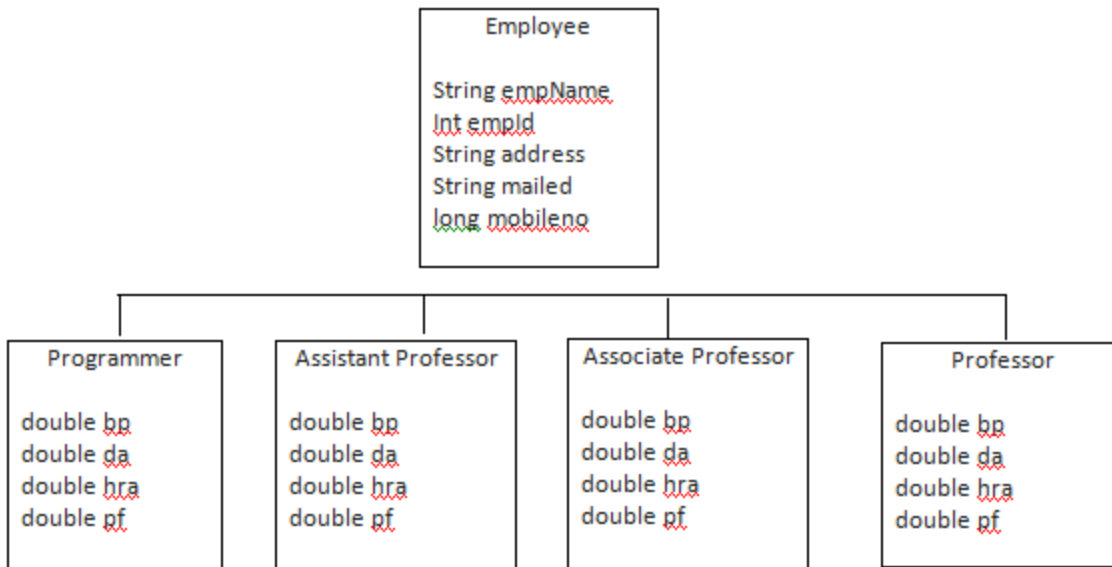
6. What is the output of this program?

```
package pkg;
class output
{
    public static void main(String args[])
    {
        StringBuffer s1 = new StringBuffer("Hello World");
        s1.insert(6, "Good ");
        System.out.println(s1);
    }
}
```

Note : Output.class file is not in directory pkg.

- a) HelloGoodWorld b) HellGoodoWorld c) Compilation error d) Runtime error

AIM Develop a java application with Employee class with empName, empId, address, mailId, mobileNo as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.



SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object. It is an important part of OOPs (Object Oriented programming system). The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the **IS-A relationship**, also known as *parent-child* relationship. Why use inheritance in java?

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality. To reduce the complexity and simplify the language, multiple inheritance is not supported in java but it is possible through the use of interface

ALGORITHM

Step1: Create a base class 'Employee' with data members of empName, mailId, address, empId, mobileNo.

Step2: Create a derived class 'Programmer' inherit the properties of Employee class with its own data members of bp, da, hra, pf, netSalary, crossSalary

Step2.1: Create an instance method 'read' to read data from console with the help of 'BufferedReader' class.

Step2.2: Create an instance method 'calculate' to find net salary and cross salary

Step2.3: Create an instance method 'show' to display all the details

Step3: Create a derived class 'AssistantProfessor' inherit the properties of Employee class with its own data members of bp, da, hra, pf, netSalary, crossSalary

Step3.1: Create an instance method 'read' to read data from console with the help of 'BufferedReader' class.

Step3.2: Create an instance method 'calculate' to find net salary and cross salary

Step3.3: Create an instance method 'show' to display all the details

Step4: Create a derived class 'AssociateProfessor' inherit the properties of Employee class with its own data members of bp, da, hra, pf, netSalary, crossSalary

Step4.1: Create an instance method 'read' to read data from console with the help of 'BufferedReader' class.

Step4.2: Create an instance method 'calculate' to find net salary and cross salary

Step4.3: Create an instance method 'show' to display all the details

Step5: Create a derived class 'Professor' inherit the properties of Employee class with its own data members of bp, da, hra, pf, netSalary, crossSalary

Step5.1: Create an instance method 'read' to read data from console with the help of 'BufferedReader' class.

Step5.2: Create an instance method 'calculate' to find net salary and cross salary

Step5.3: Create an instance method 'show' to display all the details

Step6: Create a class 'HierarchicalInheritance'. In its main method, create an object for all derived classes and call their respective methods

PROGRAM

```
import java.io.*;
//Base class
class Employee {
    String empName, address, mailId;
    int empId;
    long mobileNo;
}

//Programmer class inherited from Employee class
class Programmer extends Employee {

    double bp, da, hra, pf, netSalary, crossSalary;

    //instance method to read details
    void read() throws Exception {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter Employee Name: ");
        empName = in.readLine();
        System.out.print("Enter Employee ID: ");
        empId = Integer.parseInt(in.readLine());
        System.out.print("Enter Employee address: ");
        address = in.readLine();
        System.out.print("Enter Employee mailId: ");
```

```

    mailId = in.readLine();
    System.out.print("Enter Employee mobileNo: ");
    mobileNo = Long.parseLong(in.readLine());
    System.out.print("Enter Employee Basic Pay: ");
    bp = Double.parseDouble(in.readLine());
}

//instance method to find salary detail
void calculate() {
    da = bp * 0.97;
    hra = bp * 0.10;
    pf = bp * 0.001;
    crossSalary = bp + da + hra;
    netSalary = crossSalary - pf;
}

//instance method to display details
void show() {
    System.out.println("Employee Name: " + empName);
    System.out.println("Employee ID: " + empId);
    System.out.println("Employee address: " + address);
    System.out.println("Employee mailId: " + mailId);
    System.out.println("Employee mobileNo: " + mobileNo);
    System.out.println("Basic Pay: " + bp);
    System.out.println("DA: " + da);
    System.out.println("HRA: " + hra);
    System.out.println("PF: " + pf);
    System.out.println("CROSS SALARY: " + crossSalary);
    System.out.println("NET SALARY: " + netSalary);
}
}

//AssistantProfessor inherited from Employee class
class AssistantProfessor extends Employee {

    double bp, da, hra, pf, netSalary, crossSalary;

    void read() throws Exception {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter Employee Name: ");
        empName = in.readLine();
        System.out.print("Enter Employee ID: ");
        empId = Integer.parseInt(in.readLine());
        System.out.print("Enter Employee address: ");

```

```

        address = in.readLine();
        System.out.print("Enter Employee mailId: ");
        mailId = in.readLine();
        System.out.print("Enter Employee mobileNo: ");
        mobileNo = Long.parseLong(in.readLine());
        System.out.print("Enter Employee Basic Pay: ");
        bp = Double.parseDouble(in.readLine());
    }

    void calculate() {
        da = bp * 0.97;
        hra = bp * 0.10;
        pf = bp * 0.001;
        crossSalary = bp + da + hra;
        netSalary = crossSalary - pf;
    }

    void show() {
        System.out.println("Employee Name: " + empName);
        System.out.println("Employee ID: " + empId);
        System.out.println("Employee address: " + address);
        System.out.println("Employee mailId: " + mailId);
        System.out.println("Employee mobileNo: " + mobileNo);
        System.out.println("Basic Pay: " + bp);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("PF: " + pf);
        System.out.println("CROSS SALARY: " + crossSalary);
        System.out.println("NET SALARY: " + netSalary);
    }
}

//AssociateProfessor class inherited from Employee class
class AssociateProfessor extends Employee {

    double bp, da, hra, pf, netSalary, crossSalary;

    void read() throws Exception {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter Employee Name: ");
        empName = in.readLine();
        System.out.print("Enter Employee ID: ");
        empId = Integer.parseInt(in.readLine());
        System.out.print("Enter Employee address: ");

```

```

        address = in.readLine();
        System.out.print("Enter Employee mailId: ");
        mailId = in.readLine();
        System.out.print("Enter Employee mobileNo: ");
        mobileNo = Long.parseLong(in.readLine());
        System.out.print("Enter Employee Basic Pay: ");
        bp = Double.parseDouble(in.readLine());
    }
    void calculate() {
        da = bp * 0.97;
        hra = bp * 0.10;
        pf = bp * 0.001;
        crossSalary = bp + da + hra;
        netSalary = crossSalary - pf;
    }

    void show() {
        System.out.println("Employee Name: " + empName);
        System.out.println("Employee ID: " + empId);
        System.out.println("Employee address: " + address);
        System.out.println("Employee mailId: " + mailId);
        System.out.println("Employee mobileNo: " + mobileNo);
        System.out.println("Basic Pay: " + bp);
        System.out.println("DA: " + da);
        System.out.println("HRA: " + hra);
        System.out.println("PF: " + pf);
        System.out.println("CROSS SALARY: " + crossSalary);
        System.out.println("NET SALARY: " + netSalary);
    }
}

//Professor class inherited from Employee class
class Professor extends Employee {

    double bp, da, hra, pf, netSalary, crossSalary;

    void read() throws Exception {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter Employee Name: ");
        empName = in.readLine();
        System.out.print("Enter Employee ID: ");
        empId = Integer.parseInt(in.readLine());
        System.out.print("Enter Employee address: ");
        address = in.readLine();
    }
}

```

```

    System.out.print("Enter Employee mailId: ");
    mailId = in.readLine();
    System.out.print("Enter Employee mobileNo: ");
    mobileNo = Long.parseLong(in.readLine());
    System.out.print("Enter Employee Basic Pay: ");
    bp = Double.parseDouble(in.readLine());
}

void calculate() {
    da = bp * 0.97;
    hra = bp * 0.10;
    pf = bp * 0.001;
    crossSalary = bp + da + hra;
    netSalary = crossSalary - pf;
}

void show() {
    System.out.println("Employee Name: " + empName);
    System.out.println("Employee ID: " + empId);
    System.out.println("Employee address: " + address);
    System.out.println("Employee mailId: " + mailId);
    System.out.println("Employee mobileNo: " + mobileNo);
    System.out.println("Basic Pay: " + bp);
    System.out.println("DA: " + da);
    System.out.println("HRA: " + hra);
    System.out.println("PF: " + pf);
    System.out.println("CROSS SALARY: " + crossSalary);
    System.out.println("NET SALARY: " + netSalary);
}
}

//class for testing
public class HierarchicalInheritance {
    public static void main(String sr[]) throws Exception {
        //creating an object of Programmer class and calling its instance methods
        Programmer pro = new Programmer();
        pro.read();
        pro.calculate();
        pro.show();

        //creating an object of AssistantProfessor class and calling its instance methods
        AssistantProfessor ap = new AssistantProfessor();
        ap.read();
        ap.calculate();
    }
}

```

```

ap.show();

//creating an object of AssociateProfessor class and calling its instance methods
AssociateProfessor asp = new AssociateProfessor();
asp.read();
asp.calculate();
asp.show();

//creating an object of Professor class and calling its instance methods
Professor p = new Professor();
p.read();
p.calculate();
p.show();
}
}

```

OUTPUT

```

Enter Employee Name: Raj kumar
Enter Employee ID: 001
Enter Employee address: Chennai
Enter Employee mailId: rk@xyz.com
Enter Employee mobileNo: 9962590631
Enter Employee Basic Pay: 5000

```

```

Employee Name: Raj kumar
Employee ID: 1
Employee address: Chennai
Employee mailId: rk@xyz.com
Employee mobileNo: 9962590631
Basic Pay: 5000.0
DA: 4850.0
HRA: 500.0
PF: 5.0
CROSS SALARY: 10350.0
NET SALARY: 10345.0

```

```

Enter Employee Name: Ravi
Enter Employee ID: 002
Enter Employee address: Mumbai
Enter Employee mailId: r@xyz.com
Enter Employee mobileNo: 9962590645
Enter Employee Basic Pay: 8000

```

```

Employee Name: Ravi
Employee ID: 2

```

Employee address: Mumbai
Employee mailId: r@xyz.com
Employee mobileNo: 9962590645
Basic Pay: 8000.0
DA: 7760.0
HRA: 800.0
PF: 8.0
CROSS SALARY: 16560.0
NET SALARY: 16552.0

Enter Employee Name: Raja
Enter Employee ID: 003
Enter Employee address: Chennai
Enter Employee mailId: ra@xyz.com
Enter Employee mobileNo: 996490635
Enter Employee Basic Pay: 12000

Employee Name: Raja
Employee ID: 3
Employee address: Chennai
Employee mailId: ra@xyz.com
Employee mobileNo: 996490635
Basic Pay: 12000.0
DA: 11640.0
HRA: 1200.0
PF: 12.0
CROSS SALARY: 24840.0
NET SALARY: 24828.0

Enter Employee Name: Ramesh
Enter Employee ID: 004
Enter Employee address: Bangalore
Enter Employee mailId: ram@xyz.com
Enter Employee mobileNo: 9964899303
Enter Employee Basic Pay: 18000

Employee Name: Ramesh
Employee ID: 4
Employee address: Bangalore
Employee mailId: ram@xyz.com
Employee mobileNo: 9964899303
Basic Pay: 18000.0
DA: 17460.0
HRA: 1800.0

PF: 18.0

CROSS SALARY: 37260.0

NET SALARY: 37242.0

CONCLUSION

By executing this program successfully, I learned what is hierarchical inheritance and how to implement it.

VIVA QUESTIONS

1. Output of following Java Program?

```
class Base {
    public void show() {
        System.out.println("Base::show() called");
    }
}
class Derived extends Base {
    public void show() {
        System.out.println("Derived::show() called");
    }
}
public class Main {
    public static void main(String[] args) {
        Base b = new Derived();
        b.show();
    }
}
```

a. Derived::show() called b. Base::show() called

```
2. class Base {
    final public void show() {
        System.out.println("Base::show() called");
    }
}
class Derived extends Base {
    public void show() {
        System.out.println("Derived::show() called");
    }
}

class Main {
    public static void main(String[] args) {
        Base b = new Derived();
        b.show();
    }
}
```

```
}
```

- a. Base::show() called b. Derived::show() called c. Compiler Error d. Runtime Error

```
3. class Base {  
    public static void show() {  
        System.out.println("Base::show() called");  
    }  
}  
class Derived extends Base {  
    public static void show() {  
        System.out.println("Derived::show() called");  
    }  
}  
class Main {  
    public static void main(String[] args) {  
        Base b = new Derived();  
        b.show();  
    }  
}
```

- a. Base::show() called b. Derived::show() called c. Compiler Error

4. Which of the following is true about inheritance in Java?

- 1) Private methods are final.
- 2) Protected members are accessible within a package and inherited classes outside the package.
- 3) Protected methods are final.
- 4) We cannot override private methods.

- a. 1, 2 and 4 b. Only 1 and 2 c. 1, 2 and 3 d. 2, 3 and 4

AIM Design a Java interface for ADT Stack. Implement this interface using array and List. Provide necessary exception handling in both the implementations.

SOFTWARE USED TO IMPLEMENT jdk1.8

ALGORITHM

Step1: Create an interface 'StackInterface' with methods 'push', 'pop' and 'print'

Step2: Create a class called 'ArrayStack' and implement all methods of 'StackInterface'

Step2.1: Throw an exception 'Stack is empty' when there is no item in the stack

Step2.2: Throw an exception 'Stack overflow', while inserting an item but it is full.

Step2.2: Throw an exception 'Stack underflow', while deleting an item but there is no item in it..

Step3: Create a class called 'ListNode' to create a node for a list. The node has two fields for the purpose of storing the item and its address of the next node.

Step4: Create a class called 'ListStack' and implement all methods of 'StackInterface'

Step4.1: Throw an exception 'Stack is empty' when there is no item in the liststack

Step4.2: Throw an exception 'Stack overflow', while inserting an item but it is full.

Step4.2: Throw an exception 'Stack underflow', while deleting an item but there is no item in it..

Step5: Create a class called 'Stack' to show required menus and to test the array stack and list stack

TUTORIALS

In a stack, elements are added (pushed) to the one side of the stack and then retrieved (popped) from the same side of the stack. In other words, the last element you insert is the first element you can retrieve. This type of data structure is known as LIFO (Last In, First Out). An array is a way to organize data (and its storage allocations), whereas stack has a policy to insert, remove or access data. Different policies can be combined with the organizations to construct different data structures according to your needs.

For example:

- Stack using an array
- Stack using a linked list

An ArrayStack is intern based on an array. The most important difference of them all is, that a Stack is based on the LIFO (Last In First Out) system, so you add your elements to the top (push) and if you want to take an element form the stack (pop), you also take it from the top. what is the difference between array stack and Linked stack? Arrays must use contiguous memory - namely, the whole array has to be given a block in memory which is one long uninterrupted chunk. On the other hand, a linked list consists of individual nodes linked by pointers - there is no requirement that it be contiguous at all. This makes resizing arrays problematic - essentially, we have to make a new array, and copy everything into it to resize it. This can be a problem with stacks, as we often have no idea precisely how large the stack will get. Linked lists don't have this problem - we just create a new node and link it to the list as it was before.

PROGRAM

```
// Interface for stack
public interface StackInterface
{
```

```

        public void push(int element);
        public int pop();
        public void print() throws Exception;
    }

// Stack implemented using array
import java.io.*;
class ArrayStack implements StackInterface
{
    int top, MAX=3;
    int stk[]=new int[MAX];
    public ArrayStack()
    {
        top=-1;
    }
    //Stack method to insert an element
    public void push( int element)
    {
        if(top<(MAX-1))

            stk[++top]=element;

        else
            System.out.println("Stack Overflow");
    }
    //Stack method to remove an element
    public int pop()
    {
        int element=0;
        if(top>=0)
        {
            element=stk[top--];
            return 0;
        }
        else
            return -1;
    }
    //method to display stack detail
    public void print() throws Exception
    {
        int i;
        if(top== -1)
            throw new Exception( "Stack is empty");
        else

```

```

        {
            System.out.println("The content of the stack is ");
            for(i=top; i>=0; i--)
                System.out.print("--->" +stk[i]);
        }
    }
}

```

//Stack implemented using List

```

class ListNode
{
    int element;
    ListNode next;
    public ListNode(int ele)
    {
        this(ele,null);
    }
    public ListNode(int ele,ListNode n)
    {
        element=ele;
        next=n;
    }
}

```

class ListStack implements StackInterface

```

{
    ListNode top;
    ListNode newnode;
    public ListStack()
    {
        top=null;
    }
    public void push(int x)
    {
        newnode = new ListNode(x,top);
        top=newnode;
    }

    public int pop()
    {
        int ele=0;
        if(top==null)
            return -1;
    }
}

```

```

        else
        {
            ele=top.element;
            top=top.next;
            return ele;
        }
    }

    public void print() throws Exception
    {
        ListNode current=top;
        if(top==null)
            throw new Exception("List stack is empty");
        else
        {
            System.out.println("The content of the stack is");
            while(current!=null)
            {
                System.out.print(current.element+"-->");
                current=current.next;
            }
        }
    }
}

```

```

//Class to test
public class Stack
{
    static void displaysubmenu()
    {
        System.out.println();
        System.out.println("\tSub menu");
        System.out.println("\t1.push");
        System.out.println("\t2.pop");
        System.out.println("\t3.Back to main menu");
    }
    static void displaymainmenu()
    {
        System.out.println("\tM a i n M e n u");
        System.out.println("\t1. ArrayStack");
        System.out.println("\t2. ListStack");
        System.out.println("\t3. Exit");
        System.out.print("\tOption[1-3]:");
    }
}

```

```

}
public static void main(String sr[]) throws Exception
{
int option=0, data=0, status, choice=0;
do
{
displaymainmenu();
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
option=Integer.parseInt(br.readLine());
switch(option)
{
case 1:
ArrayStack s=new ArrayStack();
do
{
try{
displaysubmenu();
choice=Integer.parseInt(br.readLine());
if(choice==1)
{
System.out.println("Enter the element");
data=Integer.parseInt(br.readLine());
s.push(data);
s.print();
}
else if(choice==2)
{
status=s.pop();
if(status== -1)
throw new Exception("Stack underflow");
else
{
System.out.println("The popped value is"+data);
s.print();
}
}
}
else
break;
}catch(Exception e){System.out.println(e);}
}while(choice<=4);
break;
case 2:
ListStack mystk=new ListStack();
do

```

```

        {
        try{
displaysubmenu();
choice=Integer.parseInt(br.readLine());
if(choice==1)
{
        System.out.print("Enter the element : ");
        data=Integer.parseInt(br.readLine());
        mystk.push(data);
        mystk.print();
}
else if(choice==2)
{
        status=mystk.pop();
        if(status== -1)
                System.out.println("List is empty");
        else
                System.out.println("Popped element is:"+status);
                mystk.print();
}
else
        break;
}catch(Exception e){}
}while(choice<=4);
break;
case 3:
        System.exit(0);
}
}while(option<=3);
}
}

```

OUTPUT

```

Main Menu
1. ArrayStack
2. ListStack
3. Exit
Option[1-3]:1

```

```

Sub menu
1.push
2.pop
3.Back to main menu

```

Enter the element

10

The content of the stack is

--->10

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

20

The content of the stack is

--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

2

The popped value is20

The content of the stack is

--->10

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

20

The content of the stack is

--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

30

The content of the stack is

--->30--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

3

M a i n M e n u

1. ArrayStack

2. ListStack

3. Exit

Option[1-3]:2

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element : 10

The content of the stack is

10-->

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element : 20

The content of the stack is

20-->10-->

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element : 30

The content of the stack is

30-->20-->10-->

Sub menu

1.push

2.pop

3.Back to main menu

2

Popped element is:30

The content of the stack is

20-->10-->

Sub menu

1.push

2.pop

3.Back to main menu

2

Popped element is:20

The content of the stack is

10-->

Sub menu

1.push

2.pop

3.Back to main menu

2

Popped element is:10

Sub menu

1.push

2.pop

3.Back to main menu

2

List is empty

Sub menu

1.push

2.pop

3.Back to main menu

3

M a i n M e n u

1. ArrayStack

2. ListStack

3. Exit

Option[1-3]:1

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

10

The content of the stack is

--->10

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

20

The content of the stack is

--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

30

The content of the stack is

--->30--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

1

Enter the element

40

Stack Overflow

The content of the stack is

--->30--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

2

The popped value is40

The content of the stack is

--->20--->10

Sub menu

1.push

2.pop

3.Back to main menu

2

The popped value is40

The content of the stack is

--->10

Sub menu

1.push

2.pop

3.Back to main menu

2

The popped value is40

java.lang.Exception: Stack is empty

Sub menu
1.push
2.pop
3.Back to main menu

3

M a i n M e n u
1. ArrayStack
2. ListStack
3. Exit
Option[1-3]:3

CONCLUSION

Upon executing this program, I learned how to implement stack principles using array and list

VIVA QUESTIONS

1. Which of the following real world scenarios would you associate with a stack data structure?

- a) piling up of chairs one above the other b) people standing in a line to be serviced at a counter
c) offer services based on the priority of the customer d) all of the mentioned

2. What does the following function check for? (all necessary headers to be included and function is called from main)

```
#define MAX 10

typedef struct stack
{
    int top;
    int item[MAX];
}stack;

int function(stack *s)
{
    if(s->top == -1)
        return 1;
    else return 0;
}
```

- a) full stack b) invalid index c) empty stack d) infinite stack

3. What does 'stack underflow' refer to?

- a) accessing item from an undefined stack b) adding items to a full stack
c) removing items from an empty stack d) index out of bounds exception

4. What is the time complexity of pop() operation when the stack is implemented using an array?

- a) O(1) b) O(n) c) O(logn) d) O(nlogn)

5. Which of the following array position will be occupied by a new element being pushed for a stack of size N elements(capacity of stack > N).

- a) $S[N-1]$. b) $S[N]$. c) $S[1]$. d) $S[0]$.

6. What happens when you pop from an empty stack while implementing using the Stack ADT in Java?

- a) Undefined error b) Compiler displays a warning
c) `EmptyStackException` is thrown d) `NoStackException` is thrown

7. 'Array implementation of Stack is not dynamic', which of the following statements supports this argument?

- a) space allocation for array is fixed and cannot be changed during run-time
b) user unable to give the input for stack operations
c) a runtime exception halts execution
d) all of the mentioned

8. Which of the following array element will return the top-of-the-stack-element for a stack of size N elements(capacity of stack > N).

- a) $S[N-1]$. b) $S[N]$. c) $S[N-2]$. d) $S[N+1]$.

AIM Write a program to perform string operations using ArrayList. Write functions for the following:

- a) Append – add at end
- b) Insert – add at particular index
- c) Search
- d) List all string starts with given letter
- e) List all strings

SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

ArrayList is a part of collection framework and is present in java.util package. It provides us dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed.

- ArrayList inherits AbstractList class and implements List interface.
- ArrayList is initialized by a size, however the size can increase if collection grows or shrunk if objects are removed from the collection.
- Java ArrayList allows us to randomly access the list.
- ArrayList can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases (see this for details).
- ArrayList in Java can be seen as similar to vector in C++.
- Java ArrayList class can contain duplicate elements.
- Java ArrayList class maintains insertion order.
- Java ArrayList class is non synchronized.
- Java ArrayList allows random access because array works at the index basis.
- In Java ArrayList class, manipulation is slow because a lot of shifting needs to be occurred if any element is removed from the array list.

ArrayList methods

1. void add(int index, Object element): This method is used to insert a specific element at a specific position index in a list.
2. int indexOf(Object O): The index the first occurrence of a specific element is either returned, or -1 in case the element is not in the list.
3. boolean add(Object o): This method is used to append a specific element to the end of a list.
4. int size() will give you the number of elements in the Array List
5. boolean contains(Object o) method will return true if the list contains the specified element.
6. public E get(int index): Returns the element at the specified position in this list.
7. public boolean isEmpty(): Returns true if this list contains no elements.

ALGORITHM

Step1: Create a class called 'ArrayListExample' and declare list as ArrayList to hold list of items and also create a variable for BufferedReader to read values from console

Step2: Create a constructor where add some list of names to list

Step3: create an instance method 'displayMainMenu' to show list of operations that are mentioned in the problem statement. Based on the selected option, call specific method to take appropriate action.

Step4: Create an instance method 'appendAtEnd' to add the specified item at the end of list.

Step5: Create an instance method 'listAllStringBeginsWithLetter' to show all the items begin with the specified letter. If there is no item begins with the specific character then print 'No item begins with the specified letter'.

Step6: Create an instance method 'listAllString' to show all the items in the list. If there is no item then print 'No item presents in the list'.

Step7: Create an instance method 'search' to find whether an item present in the list or not. If present print 'Item present' otherwise print 'Item not present'.

Step8: Inside main method create an object for the class and call 'displayMainMenu'.

SOFTWARE USED TO IMPLEMENT jdk1.8

PROGRAM

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

public class ArrayListExample {

    ArrayList<String> list = new ArrayList<String>();
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String item = "";

    public ArrayListExample() {
        list.add("Raj kumar");
        list.add("Ravi");
        list.add("Kumar");
        list.add("Karan");
        list.add("Suresh");
        list.add("Ram");
        list.add("Siva");
        list.add("Lorenz");
    }

    void displayMainMenu() throws Exception {
        int yesorno, choice = 0;

        do {
            System.out.println(" Main Menu \n 1) Append String at end \n 2) Insert String at particular index \n 3)
            Search String \n 4) List all string starts with given letter \n 5) List all strings \n 6) Exit \n");
            System.out.print("Enter your choice[1-6]: ");
            choice = Integer.parseInt(br.readLine());
            switch (choice) {

                case 1:
                    System.out.print("Enter the string to be inserted: ");
```

```

        item = br.readLine();
        appendAtEnd(list, item);
        break;
    case 2:
        System.out.print("Enter the string to be inserted: ");
        item = br.readLine();
        System.out.print("Enter the location where to insert: ");
        int index = Integer.parseInt(br.readLine());
        insertAtIndex(item, index);
        break;
    case 3:
        System.out.print("Enter the string to be searched: ");
        item = br.readLine();
        search(list, item);
        break;
    case 4:
        System.out.print("Enter the first letter of string to be searched: ");
        item = br.readLine();
        listAllStringBeginsWithLetter(list, item);
        break;
    case 5:
        listAllString(list);
        break;

    case 6:
        System.exit(0);
    default:
        System.out.println("Enter correct choice");
}
System.out.print("Want to continue MainMenu? [Press 1 for Yes / 0 for No]: ");
yesorno = Integer.parseInt(br.readLine());
} while (yesorno == 1);
}

void search(ArrayList list, String item) {
    if (list.contains(item)) {
        System.out.println("Item present in the list");
    } else {
        System.out.println("Item not present in the list");
    }
}

ArrayList insertAtIndex(String item, int index) {
    list.add(index, item);
}

```

```

    return list;
}

void listAllString(ArrayList list) {
    if (list.isEmpty()) {
        System.out.println("No item in the list");
    } else {
        System.out.println(list);
    }
}

void listAllStringBeginsWithLetter(ArrayList list, String ch) {
    ArrayList<String> temp = new ArrayList();
    for (int i = 0; i < list.size(); i++) {
        item = list.get(i).toString();
        if (item.startsWith(ch)) {
            temp.add(item);
        }
    }
    if (temp.isEmpty()) {
        System.out.println("No item begins with this letter");
    } else {
        System.out.println(temp);
    }
}

ArrayList appendAtEnd(ArrayList list, String item) throws Exception {
    list.add(item);
    return list;
}

public static void main(String args[]) throws Exception {
    ArrayListExample ale = new ArrayListExample();
    ale.displayMainMenu();
}
}

```

OUTPUT

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter

- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 5

[Raj kumar, Ravi, Kumar, Karan, Suresh, Ram, Siva, Lorenz]

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 1

Enter the string to be inserted: Bill

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 5

[Raj kumar, Ravi, Kumar, Karan, Suresh, Ram, Siva, Lorenz, Bill]

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 2

Enter the string to be inserted: Sudhakar

Enter the location where to insert: 2

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String

- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 5

[Raj kumar, Ravi, Sudhakar, Kumar, Karan, Suresh, Ram, Siva, Lorenz, Bill]

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 3

Enter the string to be searched: Lorenz

Item present in the list

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 3

Enter the string to be searched: Gates

Item not present in the list

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 1

Main Menu

- 1) Append String at end
- 2) Insert String at particular index
- 3) Search String
- 4) List all string starts with given letter
- 5) List all strings
- 6) Exit

Enter your choice[1-6]: 4

Enter the first letter of string to be searched: L

[Lorenz]

Want to continue MainMenu? [Press 1 for Yes / 0 for No]: 0

CONCLUSION

Upon executing this program successfully, I learned what are the methods supported by ArrayList and can be used to perform string operations.

VIVA QUESTIONS

1. Which of these standard collection classes implements a dynamic array?

- a) AbstractList b) LinkedList c) ArrayList d) AbstractSet

2. Which of these class can generate an array which can increase and decrease in size automatically?

- a) ArrayList() b) DynamicList() c) LinkedList() d) MallocList()

3. Which of one these method of ArrayList class is used to obtain present size of an object?

- a) size() b) length() c) index() d) capacity()

4. Which of these methods can be used to obtain a static array from an ArrayList object?

- a) Array() b) covertArray() c) toArray() d) covertToArray()

5. What is the output of this program?

```
import java.util.*;
class ArrayList
{
    public static void main(String args[])
    {
        ArrayList obj = new ArrayList();
        obj.add("A");
        obj.add("B");
        obj.add("C");
        obj.add(1, "D");
        System.out.println(obj);
    }
}
```

- a) [A, B, C, D]. b) [A, D, B, C]. c) [A, D, C]. d) [A, B, C].

6. What is the output of this program?

```
import java.util.*;
class Output
{
    public static void main(String args[])
    {
        ArrayList obj = new ArrayList();
        obj.add("A");
        obj.add(0, "B");
```

```
        System.out.println(obj.size());
    }
}
```

a) 0 b) 1 c) 2 d) Any Garbage Value

7. What is the output of this program?

```
import java.util.*;
class Output
{
    public static void main(String args[])
    {
        ArrayList obj = new ArrayList();
        obj.add("A");
        obj.ensureCapacity(3);
        System.out.println(obj.size());
    }
}
```

a) 1 b) 2 c) 3 d) 4

AIM Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named print Area(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.

SOFTWARE USED TO IMPLEMENT jdk1.8

ALGORITHM

Step1: Create an abstract class 'Shape' with two abstract variables and a method 'printArea'

Step2: Create a class 'Rectangle' which extends abstract class 'Shape' wherein read values of abstract variables with the help of 'BufferedReader' and implement abstract method 'printArea' of abstract class 'Shape' to find and print area of rectangle

Step3: Create a class 'Triangle' which extends abstract class 'Shape' wherein read value of abstract variable with the help of 'BufferedReader' to know the side of a triangle and implement abstract method 'printArea' of abstract class 'Shape' to find and print area of triangle

Step4: Create a class 'Circle' which extends abstract class 'Shape' wherein read value of abstract variable with the help of 'BufferedReader' to know the radius of circle and implement abstract method 'printArea' of abstract class 'Shape' to find and print area of triangle

Step5: Create a class 'Area' and inside main method, create an object for the above classes and call 'printArea' of respective classes with the help of those objects.

SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

An abstract class is a class that is declared **abstract**—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be sub classed. An abstract class may have static fields and static methods. When an abstract class is sub classed, the subclass usually provides implementations for all of the abstract methods in its parent class. Any subclass of an abstract class must either implement all of the abstract method or be itself declared abstract.

An abstract method is a method that is declared without an implementation. An abstract class can be used as a type of template for other classes, and most commonly used for inheritance. When we use abstract class? Suppose we were modeling the behavior of animals, by creating a class hierarchy that started with a base class called Animal. Animals are capable of doing different things like flying, digging and walking, but there are some common operations as well like eating and sleeping. Some common operations are performed by all animals, but in a different way as well. When an operation is performed in a different way, it is a good candidate for an abstract method (forcing subclasses to provide a custom implementation).

Consider using abstract classes if any of these statements apply to your situation:

- You want to share code among several closely related classes.
- You expect that classes that extend your abstract class have many common methods or fields or require access modifiers other than public (such as protected and private).
- You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.

PROGRAM

```
import java.io.*;

// Abstract class with abstract method and variable
abstract class Shape {
    int a, b;
    abstract void printArea();
}

//Class Rectangle extends abstract class
class Rectangle extends Shape {
    //Implementation of abstract method defined in abstract class
    void printArea() {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Area of Rectangle");
            System.out.print("Enter length:");
            a = Integer.parseInt(br.readLine());
            System.out.print("Enter breadth:");
            b = Integer.parseInt(br.readLine());
            double area = a * b;
            System.out.println("Area of Rectangle: " + area);
        } catch (Exception e) {
        }
    }
}

//Class Triangle extends abstract class

class Triangle extends Shape {
    //Implementation of abstract method defined in abstract class
    void printArea() {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Area of Triangle");
            System.out.print("Enter side:");
            a = Integer.parseInt(br.readLine());
            double area = a * a;
            System.out.println("Area of Square: " + area);
        } catch (Exception e) {
        }
    }
}

//Class circle extends abstract class
```

```

class Circle extends Shape {

    //Implementation of abstract method defined in abstract class
    void printArea() {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Area of Circle");
            System.out.print("Enter radius:");
            a = Integer.parseInt(br.readLine());
            double area = 3.14 * a * a;
            System.out.println("Area of Square: " + area);
        } catch (Exception e) {
        }
    }
}

//Class Area to test
class Area {
    public static void main(String args[]) {
        //Create object of 'Rectangle' and call its 'printArea'
        Rectangle r = new Rectangle();
        r.printArea();

        //Create object of 'Triangle and call its 'printArea'
        Triangle t = new Triangle();
        t.printArea();

        //Create object of 'Circle' and call its 'printArea'
        Circle c = new Circle();
        c.printArea();
    }
}

```

OUTPUT

```

Area of Rectangle
Enter length:10
Enter breadth:20
Area of Rectangle: 200.0
Area of Triangle
Enter side:20
Area of Square: 400.0
Area of Circle
Enter radius:50
Area of Square: 7850.0

```

CONCLUSION

By executing this exercise successfully, I learned

- What is abstract method and when to use it
- How to create an abstract method and how to extend to other classes.
- How to provide different implementation in subclasses for same abstract method

VIVA QUESTIONS

1. If a class inheriting an abstract class does not define all of its function then it will be known as?

- a) Abstract b) A simple class c) Static class d) None of the mentioned

2. Which of these is not a correct statement?

- a) Every class containing abstract method must be declared abstract
b) Abstract class defines only the structure of the class not its implementation
c) Abstract class can be initiated by new operator
d) Abstract class can be inherited

3. What is the output of this program?

```
abstract class A
{
    int i;
    abstract void display();
}
class B extends A
{
    int j;
    void display()
    {
        System.out.println(j);
    }
}
class Abstract_demo
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.j=2;
        obj.display();
    }
}
```

- a) 0 b) 2 c) Runtime Error d) Compilation Error

4. Which of the following is FALSE about abstract classes in Java

- a) If we derive an abstract class and do not implement all the abstract methods, then the derived class should also be marked as abstract using 'abstract' keyword
- b) Abstract classes can have constructors
- c) A class can be made abstract without any abstract method
- d) A class can inherit from multiple abstract classes.

5. Predict the output of the following program.

```
abstract class demo
{
    public int a;
    demo()
    {
        a = 10;
    }
    abstract public void set();
    abstract final public void get();
}

class Test extends demo
{
    public void set(int a)
    {
        this.a = a;
    }

    final public void get()
    {
        System.out.println("a = " + a);
    }

    public static void main(String[] args)
    {
        Test obj = new Test();
        obj.set(20);
        obj.get();
    }
}
```

- a) a=10
- b) a=20
- c) Compile time error
- d)None

AIM Write a Java program to implement user defined exception handling. When the user enters negative amount, program will throw an user defined exception called 'Invalid Amount'

SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

A Java Exception is an abnormal condition that arises during the execution of a program and also called a run-time error. An exception in Java signals the unusual or catastrophic situations that can arise.

There are two types of exceptions:

1. **Checked Exceptions:** These are the type of exceptions for which the compiler checks to ensure that your code is prepared for handling such exceptions. List of checked exception:
 - *NoSuchMethodException* A requested method does not exist.
 - *ClassNotFoundException* Class not found.
 - *IllegalAccessException* Access to a class is denied.
2. **Unchecked Exceptions:** The compiler does not check for such type of exceptions. Unchecked Exceptions comprise of run time exceptions (of type *RuntimeException* or its subclasses) and errors (of type *Error* or its subclasses). Runtime Exceptions occur due to program bugs and include exceptions such as division by zero and invalid array indexing. List of unchecked exception:
 - *ArithmeticException* Arithmetic error, such as divide-by-zero.
 - *ArrayIndexOutOfBoundsException* Array index is out-of-bounds.
 - *ArrayStoreException* Assignment to an array element of an incompatible type.
 - *ClassCastException* Invalid cast.

You can also create your own exception sub class simply by extending java Exception class. You can define a constructor for your Exception sub class and you can override the `toString()` function to display your customized message on catch. Steps:

1. Extend the Exception class to create your own exception class.
2. You don't have to implement anything inside it, no methods are required.
3. You can have a Constructor if you want.
4. You can override the `toString()` function, to display customized message.

Why do we need to create a new exception, instead of using the ones defined by JDK? When you couldn't find any relevant exceptions in the JDK, it's time to create new ones of your own.

ALGORITHM

Step1: Create a class called 'NegativeAmtException' and extend the class Exception to it.

Step1.1: Create constructor for 'NegativeAmtException' with an argument to set customized exception message to a string variable 'msg'

Step1.2: Override the `toString()` method, to display customized message

Step2: Create a class called 'UserDefinedException' with 'main' method

Step2.1: Read amount from console with the help of Scanner object.

Step2.2: Create a try block wherein check if amount is less than zero or not.

Step2.2.1: If the amount is less than zero then throw the customized exception message called 'Invalid Amount' by creating an object of 'NegativeAmtException' and passing the customized message to it.

Step2.2.2: If the amount is not less than zero then just print it.

Step2.3: Next to the try block create a catch block wherein handle the thrown message, that is print the message to console.

PROGRAM

```
import java.util.Scanner;
class NegativeAmtException extends Exception
{
    String msg;
    NegativeAmtException(String msg)
    {
        this.msg=msg;
    }
    public String toString()
    {
        return msg;
    }
}

public class UserDefinedException
{
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        System.out.print("Enter Amount:");
        int a=s.nextInt();
        try
        {
            if(a<0)
            {
                throw new NegativeAmtException("Invalid Amount");
            }
            System.out.println("Amount Deposited");
        }
        catch(NegativeAmtException e)
        {
            System.out.println(e);
        }
    }
}
```

OUTPUT

Enter Amount:-1000
Invalid Amount
Enter Amount:1000
Amount Deposited

CONCLUSION

Upon completion of this experiment, student will be able to demonstrate how to create user defined exception and handle it.

VIVA QUESTIONS

1. Predict the output of following Java program

```
class Test extends Exception { }  
class Main {  
    public static void main(String args[]) {  
        try {  
            throw new Test();  
        }  
        catch(Test t) {  
            System.out.println("Got the Test Exception");  
        }  
        finally {  
            System.out.println("Inside finally block ");  
        }  
    }  
}
```

- a) Got the Test Exception b) Got the Test Exception c) Inside finally block d) Compiler Error
 Inside finally block

2. Output of following Java program?

```
class Main {  
    public static void main(String args[]) {  
        int x = 0;  
        int y = 10;  
        int z = y/x;  
    }  
}
```

- a) Compiler Error b) Compiles and runs fine c) Compiles fine but throws ArithmeticException except
Predict the output of following Java program

3. Output of following Java program?

```

class Base extends Exception {}
class Derived extends Base {}

public class Main {
    public static void main(String args[]) {
        // some other stuff
        try {
            // Some monitored code
            throw new Derived();
        }
        catch(Base b) {
            System.out.println("Caught base class exception");
        }
        catch(Derived d) {
            System.out.println("Caught derived class exception");
        }
    }
}

```

- | | |
|--|--|
| a) Caught base class exception | b) Caught derived class exception |
| c) Compiler Error because derived is not throwable
is caught before derived class | d) Compiler Error because base class exception |

4. Output of following Java program?

```

class Test
{
    public static void main (String[] args)
    {
        try
        {
            int a = 0;
            System.out.println ("a = " + a + "\n");
            int b = 20 / a;
            System.out.println ("b = " + b);
        }

        catch(ArithmeticException e)
        {
            System.out.println ("Divide by zero error");
        }

        finally
        {

```

```

        System.out.println ("inside the finally block");
    }
}
}

```

- a) Compile error b) Divide by zero error c) a = 0 d) inside the finally block
e) a = 0
 Divide by zero error
 inside the finally block

5. Predict the output of the following program.

```

class Test
{
    String str = "a";
    void A()
    {
        try
        {
            str += "b";
            B();
        }
        catch (Exception e)
        {
            str += "c";
        }
    }

    void B() throws Exception
    {
        try
        {
            str += "d";
            C();
        }
        catch(Exception e)
        {
            throw new Exception();
        }
        finally
        {
            str += "e";
        }
        str += "f";
    }
}

```

```

    }

    void C() throws Exception
    {
        throw new Exception();
    }
    void display()
    {
        System.out.println(str);
    }
    public static void main(String[] args)
    {
        Test object = new Test();
        object.A();
        object.display();
    }
}

```

- a) abdef b) abdec c) abdefc

6. Which of these is a super class of all errors and exceptions in the Java language?

- a) RuntimeExceptions b) Throwable c) Catchable d) None of the above

7. The built-in base class in Java, which is used to handle all exceptions is

- a) Raise b) Exception c) Error d) Throwable

Exercise No.: 08

Java program to display given file metadata information

AIM Write a Java program that reads a file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

SOFTWARE USED TO IMPLEMENT jdk1.8

ALGORITHM

Step1: Create a class called 'FileDemo' with main method

Step2: Inside main method, create file object. While creating, pass name of a file

Step3: Print file required file metadata by using the various file methods

PROGRAM

```
import java.io.*;
class FileDemo
{
    public static void main(String args[])
    {
        File f1=new File("e:\\test.txt");
        System.out.println("File name: "+f1.getName());
        System.out.println("Path: "+f1.getPath());
        System.out.println("Parent: "+f1.getParent());
        System.out.println("Is exist: "+f1.exists());
        System.out.println("Can Read: "+f1.canRead());
        System.out.println("Can write: "+f1.canWrite());
        System.out.println("Is Directory: "+f1.isDirectory());
        System.out.println("Is File: "+f1.isFile());
        System.out.println("Last Modified: "+new Date(f1.lastModified()));
        System.out.println("File Length: "+f1.length()+"bytes");
    }
}
```

OUTPUT

File name: test.txt

Path: e:\test.txt

Parent: e:\

Is exist: true

Can Read: true

Can write: true

Is Directory: false

Is File: true

Last Modified: Tue Jun 26 13:16:05 IST 2018

File Length: 37 bytes

CONCLUSION

Upon completing this experiment, student will be able to print the various file metadata using various methods of file object.

VIVA QUESTIONS

1. The package contains a large number of stream classes that provide capabilities for processing all types of data.

- A) java.awt B) java.io C) java.util D) java.net

2. State whether the following statements about the stream in Java.

- i) The two basic streams used are the input and the output streams.
ii) Filters are used to read data from one stream and write it to another stream.

- A) True, True B) True, False C) False, True D) False, False

3. What is the output of the code?

```
try{  
    File f=new File("a.txt");  
}catch(Exception e) {}  
}catch(IOException ie){}
```

Is this code create new file name a.txt

- a) true b) false c) compiler error d) run time error

4. System class is defined in

- a) java.util b) java.lang c) java.io d) java.awt

5. What package holds the File class?

- a). java.io b). java.file c). java.util d). java.lang

6. Does constructing a File object automatically create a disk file?

- a. No—a File object is an interface to a file or directory which might not actually exist.
b. No—if the file already exists it is not created; otherwise it is.
c. Yes—the File object contains all the data of the file in memory can be used in place of the disk file.
d. Yes—constructing a File object always creates a new file.

7. The path name of a file is given by: C:\MyFiles\Programs\Examples\someFile.txt

Is this a relative or an absolute path name?

- A. relative B. absolute C. both D. neither

8. The path name of a file is given by: C:\MyFiles\Programs\Examples\someFile.txt

What is the absolute path name of the directory the file is located in?

- A. Examples B. ..\Examples\someFile.txt
C. C:\ D. C:\MyFiles\Programs\Examples

9. What method of File is used to test if a file or directory exists?

- A. `isFile()` B. `isDirectory()` C. `list()` D. `exists()`

10. What File method is used to remove a file?

- A. `delete()` B. `length()` C. `exists()` D. This cannot be done.

11. In the following, what is the directory separator character? `C:\MyFiles\Programs\Examples\someFile.txt`

- A. `:` B. `\` C. `/` D. `.`

AIM Write Java Program that implements a multithread application that has three threads. First thread generates Random integer for every second and if the value is even, second thread computes the square of number and prints. If the value is odd, the third thread will print the value of cube of number

SOFTWARE USED TO IMPLEMENT jdk1.8

TUTORIALS

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms:

1. Extending the Thread class
2. Implementing the Runnable Interface

By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

public void start() - starts the thread in a separate path of execution, then invokes the run() method on this Thread object.

public void run() - If this Thread object was instantiated using a separate Runnable target, the run() method is invoked on that Runnable object.

public static void sleep(long millisec) - Causes the currently running thread to block for at least the specified number of milliseconds.

ALGORITHM

Step1: Create a class called 'even' and implement 'Runnable' interface. Use this class to receive even number from a thread and to print square of that number in 'run' method of it

Step2: Create a class called 'odd' and implement 'Runnable' interface. Use this class to receive odd number from a thread and to print cube of that number in 'run' method of it

Step3: Create a class called 'A' that should extend 'Thread' class. Consider it as main thread.

Step3.1: Inside the 'run' method of this class generate and print random number by using 'Random' class and its method

Step3.2: Check the random is even, create an object of 'even' thread class and call its 'run' method just by starting it.

Step3.3: If the number is odd, then create an object of 'odd' thread class and call its 'run' method just by starting it.

Step3.4: Sleep the main thread for 1000 milliseconds.

Step4: Create a class called 'Multithread' with main method wherein create object for the thread class 'A' and call its run method just by starting it.

PROGRAM

```
import java.util.*;
class even implements Runnable
{
    public int x;
    public even(int x)
    {
        this.x = x;
    }
    public void run()
    {
        System.out.println("New Thread "+ x +" is EVEN and Square of "+ x + " is: " + x * x);
    }
}

class odd implements Runnable
{
    public int x;
    public odd(int x)
    {
        this.x = x;
    }
    public void run(){
        System.out.println("New Thread "+ x +" is ODD and Cube of "+ x + " is: " + x * x * x);
    }
}

class A extends Thread
{
    public void run()
    {
        int num = 0;
        Random r = new Random();
        try
        {
            for (int i = 0; i < 5; i++)
            {
                num = r.nextInt(100);
                System.out.println("Main Thread and Generated Number is " + num);
                if (num % 2 == 0)
                {
                    Thread t1 = new Thread(new even(num));
                    t1.start();
                } else {
                    Thread t2 = new Thread(new odd(num));
                }
            }
        }
    }
}
```

```

        t2.start();
    }
    Thread.sleep(1000);
    System.out.println("-----");
}
}
catch (Exception ex)
{
    System.out.println(ex.getMessage());
}
}
}
public class Multithread
{
    public static void main(String[] args)
    {
        A a = new A();
        a.start();
    }
}

```

OUTPUT

Main Thread and Generated Number is 10
 New Thread 10 is EVEN and Square of 10 is: 100

Main Thread and Generated Number is 14
 New Thread 14 is EVEN and Square of 14 is: 196

Main Thread and Generated Number is 83
 New Thread 83 is ODD and Cube of 83 is: 571787

Main Thread and Generated Number is 1
 New Thread 1 is ODD and Cube of 1 is: 1

Main Thread and Generated Number is 20
 New Thread 20 is EVEN and Square of 20 is: 400

VIVA QUESTIONS

1. In Java, a thread can be created by:

- | | |
|--------------------------------|-------------------------------------|
| A. Extending the thread class. | B. Implementing Runnable interface. |
| C. Both of the above | D. None of these |

2. When a class extends the Thread class, it should override ----- method of Thread class to start that thread

- A. start() B. run() C. init() D. go()

3. What will be the output?

```
public class Text implements Runnable
{
    public static void main(String ar[])
    {
        Thread t=new Thread(this);
        t.start();
    }
    public void run()
    {
        System.out.println("test");
    }
}
```

- A. The program does not compile because this cannot be referenced in a static method.
B. The program compiles fine, but it does not print because it does not invoke run()
C. This compiles and runs fine and displays 'test' on the console.
D. None of the above

4. what is the output?

```
public class Test implements Runnable
{
    public static void main(String srg[])
    {
        Test t=new Test();
        t.start();
    }
    public void run() { }
```

- A. Program does not compile because start() is not defined
B. Program compiles, but it does not run because start() is not defined
C. Program compiles but it does not run because run() is not implemented
D. Program compiles and runs fine

5. what is the output?

```
public class Test implements Runnable
{
```

```
public static void main(String srg[])
{
    Test t=new Test();
}
public Test()
{
    Thread t=new Thread(this);
    t.start();
}
public void run() { System.out.print("test");}
}
```

- A. Program gives compiler error because t is defined twice
- B. Program compiles with error, but it has 'this' keyword
- C. Program compiles but it does not run because run() is not called
- D. Program compiles and runs fine

AIM Write a java program to find the maximum value from the given type of elements using a generic function.

SOFTWARE USED TO IMPLEMENT jdk1.8

ALGORITHM

Step1: Create a class called 'Max'.

Step2: Create a static generic method called 'maximum' and declare it to return a generic 'Comparable' object

Step2.1: Create a generic variable 'max' and assign one of the arguments of 'maximum'

Step2.2: Compare 'max' with other arguments of 'maximum' using 'compareTo' method

Step2.3: If 'compareTo' returns true then assign that argument to 'max' and return it.

Step3: Inside main method, call generic method 'maximum' four times. During the first call pass three int values; second call pass three double values; third call pass three string values and fourth call pass three char values

PROGRAM

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

```
public class Max{
```

```
// determines the largest of three Comparable objects
```

```
public static <T extends Comparable<T>> T maximum(T x, T y, T z) {
```

```
    System.out.print("Max of "+ x+" "+y+" "+z+" is : ");
```

```
    T max = x; // assume x is initially the largest
```

```
    if(y.compareTo(max) > 0) {
```

```
        max = y; // y is the largest so far
```

```
    }
```

```
    if(z.compareTo(max) > 0) {
```

```
        max = z; // z is the largest now
```

```
    }
```

```
    return max; // returns the largest object
```

```
}
```

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

```
    System.out.println(maximum( 3, 4, 5 ));
```

```
    System.out.println(maximum( 6.6, 8.8, 7.7 ));
```

```
    System.out.println(maximum("pear", "apple", "orange"));
```

```
    System.out.println(maximum('b', 'a', 'o'));
```

```
}  
}
```

OUTPUT

Max of 3 4 5 is : 5

Max of 6.6 8.8 7.7 is : 8.8

Max of pear apple orange is : pear

Max of b a o is : o

CONCLUSION

Upon executing this program successfully, students will be able to demonstrate how to create and use a generic method to find a maximum among three values and he/she will be able to know the use of Comparable interface and its method.

VIVA QUESTIONS

1. What are generic methods?

- a) Generic methods are the methods defined in a generic class
- b) Generic methods are the methods that extend generic class's methods
- c) Generic methods are methods that introduce their own type parameters
- d) Generic methods are methods that take void parameters

2. Which of these type parameters is used for a generic methods to return and accept any type of object?

- a) K
- b) N
- c) T
- d) V

3. Which of these type parameters is used for a generic methods to return and accept a number?

- a) K
- b) N
- c) T
- d) V

4. Which of these is an correct way of defining generic method?

- a) name(T1, T2, ..., Tn) { /* ... */ }
- b) public name { /* ... */ }
- c) class name[T1, T2, ..., Tn] { /* ... */ }
- d) name{T1, T2, ..., Tn} { /* ... */ }

5. Which of the following is incorrect statement regarding the use of generics and parameterized types in Java?

- a) Generics provide type safety by shifting more type checking responsibilities to the compiler
- b) Generics and parameterized types eliminate the need for down casts when using Java Collections
- c) When designing your own collections class (say, a linked list), generics and parameterized types allow you to achieve type safety with just a single class definition as opposed to defining multiple classes
- d) All of the mentioned

6. Which of the following allows us to call generic methods as a normal method?

- a) Type Interface
- b) Interface
- c) Inner class
- d) All of the mentioned

7. What is the output of this program?

```
import java.util.*;
public class genericstack <E>
{
    Stack <E> stk = new Stack <E>();
    public void push(E obj)
    {
        stk.push(obj);
    }
    public E pop()
    {
        E obj = stk.pop();
        return obj;
    }
}
class Output
{
    public static void main(String args[])
    {
        genericstack <String> gs = new genericstack<String>();
        gs.push("Hello");
        System.out.println(gs.pop());
    }
}
```

- a) H
- b) Hello
- c) Runtime Error
- d) Compilation Error

8. What is the output of this program?

```
import java.util.*;
public class genericstack <E>
{
    Stack <E> stk = new Stack <E>();
    public void push(E obj)
    {
        stk.push(obj);
    }
}
```

```

public E pop()
{
    E obj = stk.pop();
    return obj;
}
}
class Output
{
    public static void main(String args[])
    {
        genericstack <Integer> gs = new genericstack<Integer>();
        gs.push(36);
        System.out.println(gs.pop());
    }
}

```

- a) 0 b) 36 c) Runtime Error d) Compilation Error

9. What is the output of this program?

```

import java.util.*;
public class genericstack <E>
{
    Stack <E> stk = new Stack <E>();
    public void push(E obj)
    {
        stk.push(obj);
    }
    public E pop()
    {
        E obj = stk.pop();
        return obj;
    }
}
class Output
{
    public static void main(String args[])
    {
        genericstack <String> gs = new genericstack<String>();
        gs.push("Hello");
        System.out.print(gs.pop() + " ");
        genericstack <Integer> gs = new genericstack<Integer>();
        gs.push(36);
    }
}

```

```
        System.out.println(gs.pop());
    }
}
```

a) Error b) Hello c) 36 d) Hello 36

10. What is the output of this program?

```
import java.util.*;
public class genericstack <E>
{
    Stack <E> stk = new Stack <E>();
    public void push(E obj)
    {
        stk.push(obj);
    }
    public E pop()
    {
        E obj = stk.pop();
        return obj;
    }
}
class Output
{
    public static void main(String args[])
    {
        genericstack <Integer> gs = new genericstack<Integer>();
        gs.push(36);
        System.out.println(gs.pop());
    }
}
```

a) H b) Hello c) Runtime Error d) Compilation Error

AIM Design a calculator using event-driven programming paradigm of Java with the following options. a) Decimal manipulations b) Scientific manipulations

SOFTWARE USED TO IMPLEMENT jdk1.8

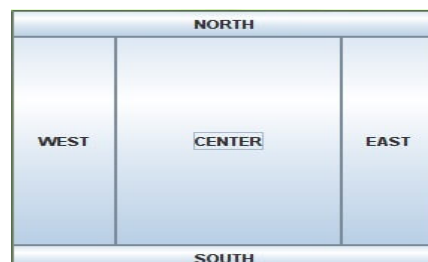
TUTORIALS

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

The javax.swing.JFrame class is a type of container that contain title bar and can have menu bars which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

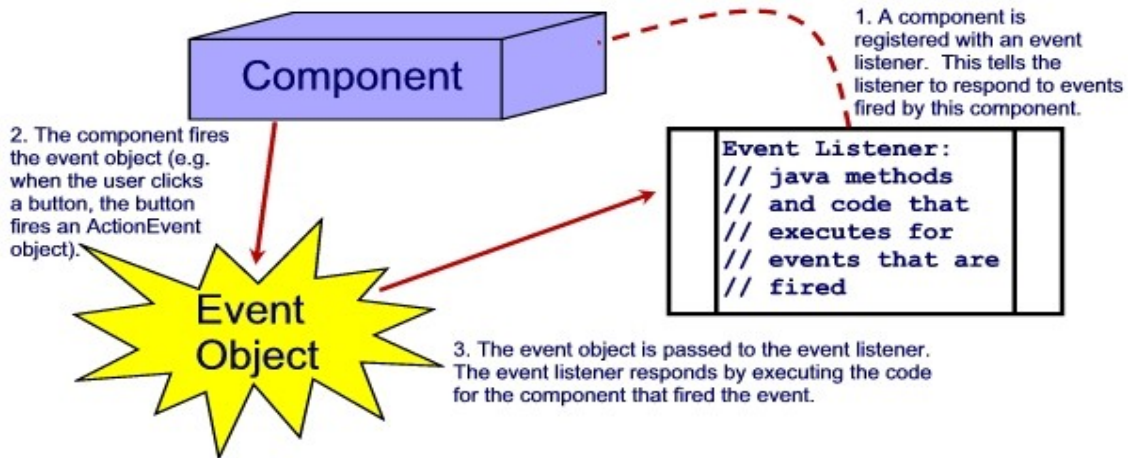
The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers. The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window.



The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle. Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Event Handling is the mechanism that controls the **event** and decides what should happen if an **event** occurs. This mechanism has the code which is known as **event handler** that is executed when an **event** occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.



Steps involved in event handling

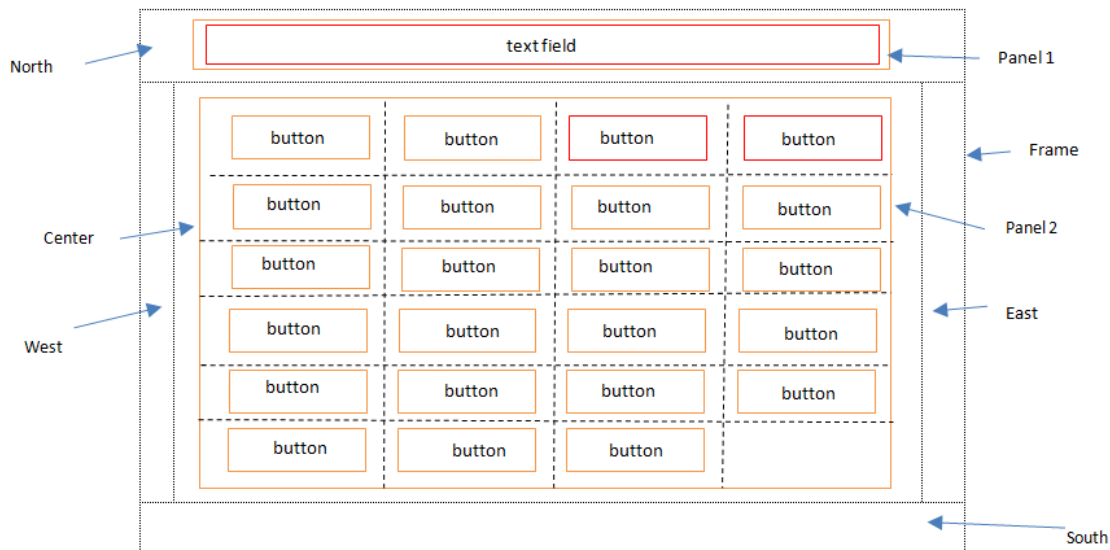
- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically
- Event object is forwarded to the method of registered listener class. The method in listener is now getting executed and returns

ALGORITHM

Step1: Create a class called 'Calc' and inherit the class JFrame

Step2: Create necessary instance variables to create buttons, text field, panels and others

Step2: Create constructor of the class 'Calc' wherein design the frame as shown below.



Step2.1: Create a panel p1 and set grid layout (1,1) to hold a text field

Step2.2: Create a text field for display the result and add it on panel p1

Step2.3: Create a panel p2 and set grid layout (6,4) to it.

Step2.4: Create necessary buttons

Step2.5: Add the buttons to panel p2

Step2.6: Add action listener to all buttons

Step2.7: Set border layout to jframe

Step2.7: Add panels p1 on north and p2 on center of jframe

Step2.9: Add widow listener to JFrame

Step3: Implement the action event listener in class 'B' wherein perform specified mathematical operations.

Step4: Inside the main method, create an object for 'Calc' JFrame, set title, set size, specify the screen location where to display and finally visible it.

PROGRAM

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Calc extends JFrame
{
    JButton jButton1, jButton2, jButton3, jButton4, jButton5, jButton6, jButton7, jButton8, jButton9,
        jButton10, jButton11, jButton12, jButton13, jButton14, jButton15, jButton16, jButton17,
        jButton18, jButton19, jButton20, jButton21, jButton22, jButton23;

    JTextField jTextField1;
    Panel p1, p2;

    String num="";
    long lo;
    long result=0;
    int option;

    public CalcFrame()
    {
        //Creation of panel1 and setting layout
        p1=new Panel();
        p1.setLayout(new GridLayout(1,1));

        //Creation of text field to display the result
        jTextField1 = new JTextField(20);
        jTextField1.setEditable(false);
        jTextField1.setBackground(Color.white);

        //add the text filed on panel1
        p1.add(jTextField1);

        //Creation of panel2 and setting layout
        p2=new Panel();
        p2.setLayout(new GridLayout(6,4));

        //Creation of buttons
        jButton1 = new JButton("C");
        jButton1.setForeground(Color.red);
```

```

jButton2 = new JButton("9");
jButton3 = new JButton("8");
jButton4 = new JButton("+");
jButton4.setForeground(Color.red);
jButton5 = new JButton("7");
jButton6 = new JButton("6");
jButton7 = new JButton("5");
jButton8 = new JButton("-");
jButton8.setForeground(Color.red);
jButton9 = new JButton("4");
jButton10 = new JButton("3");
jButton11 = new JButton("2");
jButton12 = new JButton("*");
jButton12.setForeground(Color.red);
jButton13 = new JButton("1");
jButton14 = new JButton("0");
jButton15 = new JButton("=");
jButton15.setForeground(Color.red);
jButton16 = new JButton("/");
jButton16.setForeground(Color.red);
jButton17 = new JButton("Backspace");
jButton17.setForeground(Color.red);
jButton18 = new JButton("sin");
jButton18.setForeground(Color.red);
jButton19 = new JButton("cos");
jButton19.setForeground(Color.red);
jButton20 = new JButton("tan");
jButton20.setForeground(Color.red);
jButton21 = new JButton("Sqrt");
jButton21.setForeground(Color.red);
jButton22 = new JButton("x^2");
jButton22.setForeground(Color.red);
jButton23 = new JButton("x^3");
jButton23.setForeground(Color.red);

//add the buttons on panel2
p2.add(jButton1);
    p2.add(jButton2);
    p2.add(jButton3);
    p2.add(jButton4);
    p2.add(jButton5);
    p2.add(jButton6);
    p2.add(jButton7);
    p2.add(jButton8);

```

```
p2.add(jButton9);
p2.add(jButton10);
p2.add(jButton11);
p2.add(jButton12);
p2.add(jButton13);
p2.add(jButton14);
p2.add(jButton15);
p2.add(jButton16);
p2.add(jButton17);
p2.add(jButton18);
p2.add(jButton19);
p2.add(jButton20);
p2.add(jButton21);
p2.add(jButton22);
p2.add(jButton23);
```

```
//Adding event listener to buttons
```

```
jButton1.addActionListener(new B());
jButton2.addActionListener(new B());
jButton3.addActionListener(new B());
jButton4.addActionListener(new B());
jButton5.addActionListener(new B());
jButton6.addActionListener(new B());
jButton7.addActionListener(new B());
jButton8.addActionListener(new B());
jButton9.addActionListener(new B());
jButton10.addActionListener(new B());
jButton11.addActionListener(new B());
jButton12.addActionListener(new B());
jButton13.addActionListener(new B());
jButton14.addActionListener(new B());
jButton15.addActionListener(new B());
jButton16.addActionListener(new B());
jButton17.addActionListener(new B());
jButton18.addActionListener(new B());
jButton19.addActionListener(new B());
jButton20.addActionListener(new B());
jButton21.addActionListener(new B());
jButton22.addActionListener(new B());
jButton23.addActionListener(new B());
```

```
//setting layout to JFrame
```

```
setLayout(new BorderLayout());
```

```

//add the panels to JFrame
add("North",p1);
add("Center",p2);
pack();

//add window listener to JFrame
addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
    }
});
}

//while clicking a button an action is generated which is implemented here
class B implements ActionListener
{
    public void actionPerformed(ActionEvent ae)
    {
        try
        {
            JButton jb=(JButton)ae.getSource();
            String buttonName=jb.getText();
            num=num+buttonName;
            if(buttonName.equals("1"))
            {
                lo=Long.parseLong(num);
                jTextField1.setText(""+lo);
            }
            else if(buttonName.equals("2"))
            {
                lo=Long.parseLong(num);
                jTextField1.setText(""+lo);
            }
            else if(buttonName.equals("3"))
            {
                lo=Long.parseLong(num);
                jTextField1.setText(""+lo);
            }
            else if(buttonName.equals("4"))
            {
                lo=Long.parseLong(num);
                jTextField1.setText(""+lo);
            }
        }
    }
}

```

```

}
else if(buttonName.equals("5"))
{
    lo=Long.parseLong(num);
    jTextField1.setText(""+lo);
}
else if(buttonName.equals("6"))
{
    lo=Long.parseLong(num);
    jTextField1.setText(""+lo);
}
else if(buttonName.equals("7"))
{
    lo=Long.parseLong(num);
    jTextField1.setText(""+lo);
}
else if(buttonName.equals("8"))
{
    lo=Long.parseLong(num);
    jTextField1.setText(""+lo);
}
else if(buttonName.equals("9"))
{
    lo=Long.parseLong(num);
    jTextField1.setText(""+lo);
}
else if(buttonName.equals("0"))
{
    lo=Long.parseLong(num);
    jTextField1.setText(""+lo);
}
else if(buttonName.equals("+"))
{
    option=1;
    result+=Long.parseLong(jTextField1.getText());
    num="";
    jTextField1.setText(num);
}
else if(buttonName.equals("-"))
{
    option=2;
    result+=Long.parseLong(jTextField1.getText());
    num="";
    jTextField1.setText(num);
}

```

```

}
else if(buttonName.equals("/"))
{
    option=3;
    result+=Long.parseLong(jTextField1.getText());
    num="";
    jTextField1.setText(num);
}
else if(buttonName.equals("*"))
{
    option=4;
    result+=Long.parseLong(jTextField1.getText());
    num="";
    jTextField1.setText(num);
}
else if(buttonName.equals("sin"))
{
    result+=Long.parseLong(jTextField1.getText());
    double d=Math.sin(Double.parseDouble(jTextField1.getText()));
    jTextField1.setText(""+d);
}
else if(buttonName.equals("cos"))
{
    result+=Long.parseLong(jTextField1.getText());
    double d=Math.cos(Double.parseDouble(jTextField1.getText()));
    jTextField1.setText(""+d);
}
else if(buttonName.equals("tan"))
{
    result+=Long.parseLong(jTextField1.getText());
    double d=Math.tan(Double.parseDouble(jTextField1.getText()));
    jTextField1.setText(""+d);
}
else if(buttonName.equals("Sqrt"))
{
    result+=Long.parseLong(jTextField1.getText());
    double d=Math.sqrt(Double.parseDouble(jTextField1.getText()));
    jTextField1.setText(""+d);
}
else if(buttonName.equals("x^2"))
{
    result+=Long.parseLong(jTextField1.getText());
    double d=Math.pow(Double.parseDouble(jTextField1.getText()),2.0);
    jTextField1.setText(""+d);
}

```

```

}
else if(buttonName.equals("x^3"))
{
    result+=Long.parseLong(jTextField1.getText());
    double d=Math.pow(Double.parseDouble(jTextField1.getText()),3.0);
    jTextField1.setText(""+d);
}
else if(buttonName.equals("C"))
{
    num="";
    jTextField1.setText(num);
}
else if(buttonName.equals("Backspace"))
{
    String s=jTextField1.getText();
    int len=s.length();
    if(len==1)
    {
        num="";
        jTextField1.setText(num);
    }
    else
    {
        s=s.substring(0, len-1);
        jTextField1.setText(s);
        num=s;
    }
}
else if(buttonName.equals("="))
{
    switch(option)
    {
        case 1:
            result=result+Long.parseLong(jTextField1.getText());
            jTextField1.setText(""+result);
            break;
        case 2:
            result=result-Long.parseLong(jTextField1.getText());
            jTextField1.setText(""+result);
            break;
        case 3:
            result=result/Long.parseLong(jTextField1.getText());
            jTextField1.setText(""+result);
            break;
    }
}

```

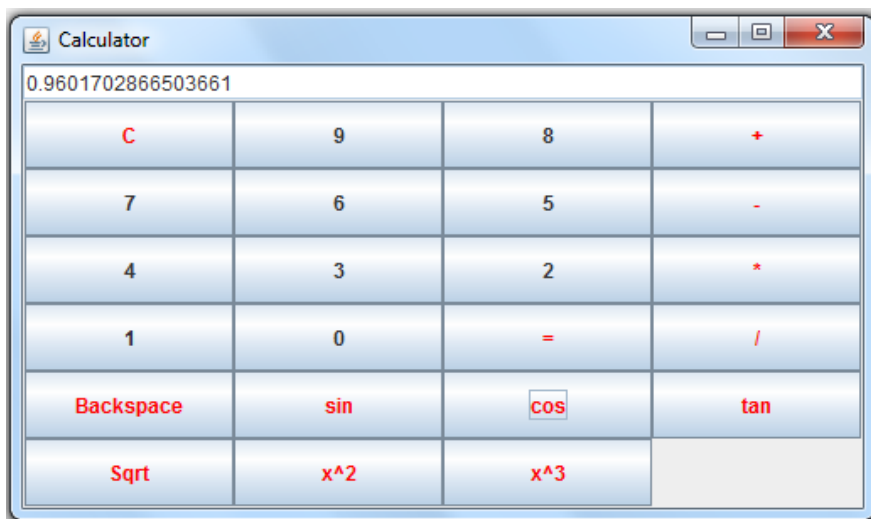
```

        case 4:
            result=result*Long.parseLong(jTextField1.getText());
            jTextField1.setText(""+result);
            break;
        }
        num="";
        result=0;
    }
}catch(Exception e){}
}
}

public static void main(String sr[])
{
    Calc c=new Calc();
    c.setTitle("Calculator");
    c.setSize(400,300);
    c.setLocation(200, 200);
    c.setVisible(true);
}
}

```

OUTPUT



CONCLUSION

By executing this program successfully I learned,

- The working principle of event driven programming model

- The use of layout managers, containers such as frame and panel and other swing components such as button, and textfield
- How to design front end GUI
- How to register event listener to components and where to define event listener methods

VIVA QUESTIONS

1. Which of these packages contains all the classes and methods required for even handling in Java?

- a) java.applet b) java.awt c) java.event d) java.awt.event

2. Where are the following four methods commonly used?

- 1) public void add(Component c)
- 2) public void setSize(int width,int height)
- 3) public void setLayout(LayoutManager m)
- 4) public void setVisible(boolean)

- a. Graphics class b. Component class c. Both A & B d. None of the above

3. Implement the Listener interface and overrides its methods is required to perform in event handling.

- a. True b. False

4. Which is the container that doesn't contain title bar and MenuBars but it can have other components like button, textfield etc?

- a. Window b. Frame c. Panel d. Container

5. These two ways are used to create a Frame: By creating the object of Frame class (association) and By extending Frame class (inheritance)

- a. True b. False

6. Give the abbreviation of AWT?

- a. Applet Windowing Toolkit b. Abstract Windowing Toolkit c. Absolute Windowing Toolkit
d. None of the above

7. Which method is used to set the graphics current color to the specified color in the graphics class?

- a. public abstract void setFont(Font font) b. public abstract void setColor(Color c)
c. public abstract void drawString(String str, int x, int y) d. None of the above

8. The Java Foundation Classes (JFC) is a set of GUI components which simplify the development of desktop applications.

- a. True b. False

CONTENT BEYOND SYLLABUS

AIM Develop a java application to demonstrate multithreaded chat server.

PROGRAM

```
// Client.java
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.awt.event.*;
public class Client extends JFrame
{
    JTextField jtf;
    JButton jb1;
    Socket s;
    String temp="";
    DataInputStream dis;

    public Client()
    {
        try
        {
            s=new Socket(InetAddress.getLocalHost(),1090);
            dis=null;
        }catch(Exception e) {}

        jtf=new JTextField(20);
        jb1=new JButton("Click to send");
        jb1.addActionListener(new B());
        getContentPane().add("North", jtf);

        getContentPane().add("South", jb1);
        pack();
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
            }
        });
    }

    class B implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            String line="";
            try
            {
                dis=new DataInputStream(System.in);
```

```

        String n=jtf.getText();
        PrintStream ps=new PrintStream(s.getOutputStream(), true);
        ps.println(n);
        }catch(Exception e){}

    }
}
public static void main(String ar[])throws Exception
{
    Client cl=new Client();
    cl.setSize(200,200);
    cl.setLocation(200,200);
    cl.setVisible(true);
    cl.setTitle("Client");
}
}

```

//Server.java

```

import javax.swing.*;
import java.io.*;
import java.net.*;
import java.awt.event.*;
public class Server extends JFrame
{
    static JTextArea jta;
    Socket client;
    ServerSocket ss;
    DataInputStream dis;
    PrintWriter out;
    public Server()
    {
        jta=new JTextArea(100,100);
        jta.setEditable(false);
        getContentPane().add("Center", jta);
        pack();
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
            }
        });
    }

    void connection()
    {
        try
        {
            ss=new ServerSocket(1090);
            System.out.println("Connection established");
            while(true)
            {
                ClientWorker w;

```

```

        w = new ClientWorker(ss.accept(), jta);
        Thread t = new Thread(w);
        t.start();
    }
} catch (Exception e) { }
}

public static void main(String sr[])
{
    Server se=new Server();
    se.setSize(200,200);
    se.setLocation(200,200);
    se.setVisible(true);
    se.setTitle("Server");
    se.connection();
}
}

class ClientWorker extends Server implements Runnable
{
    Socket client;
    public ClientWorker(Socket client, JTextArea jta)
    {
        this.client = client;
        this.jta=jta;
    }

    public void run()
    {
        try
        {
            String line="";
            DataInputStream dis=new DataInputStream(client.getInputStream());
            while(true)
            {
                line=dis.readLine();
                jta.append(line+"\n");
            }
        }catch(Exception e){}
    }
}
}

```

OUTPUT

