KMM INSTITUTE OF POST GRADUATE STUDIES,TIRUPATI
DEPT OF MCA

LIST OF PROGRAMS
MCA 107P : Object Oriented Programming Lab

1. Java Program using If-Else Statement to find roots of Quadratic Equation.
2. Java Program to Find HCF / GCD Of Given Numbers
3. Java Program for Aggregation of Classes
4. Java Program to Create Package for statistical functions
5. Java program to find Minimum and Maximum of an ARRAY
6. Java Program to implement HashTable Collection
7. Java Program to create thread using Thread and Runnable Interface
8. Java program for Handling Custom Exceptions, using Exception handling
9. Java Program to Count Lines and Word in File
10. Java program to create GUI Application using Swings to store Employ data to Database
11. Java program to create UI Application using Swings to find Employ pay
12. Java Applet Program in Using Parameters
13. Java Program to Implement UDP (User DataGram Protocol).
14. Java Program using TCP Protocol for exchange of Data
15. Java Program using JDBC To View MYSQL Data
16. Java Program to converts words into digits by using StringTokenizer class.

1. Program using If-Else Statement

Aim: Implement java program to find all roots of a quadratic equation and print in all three Cases.

Description:

If statement:

The Java if statement is used to test the condition. It checks boolean condition: true or false.There are various types of if statement in Java.

```
if (Boolean_expression) {
// Statements will execute if the Boolean expression is true
}
if(Boolean_expression) {
// Executes when the Boolean expression is true
}else {
// Executes when the Boolean expression is false
}
```

In this program we use If Statement to find roots of Quadratic Equation in all three cases such as

The term b2-4ac is known as the determinant of a quadratic equation. It specifies the nature of roots. That is,

1. if determinant > 0, roots are real and different
2. if determinant == 0, roots are real and equal
3. if determinant < 0, roots are complex complex and different

Algorithm: Roots of Quadratic Equation

1. Begin
2. Input a,b,c
3. If a=0 then go to step 8
4. Find determinant such as deter=b*b-4*a*c;
5. if deter > 0, roots are real and different
   5.1 root1 = (-b + sqrt(deter) / (2 * a)
   5.2 root2 = (-b - sqrt(deter) / (2 * a)
   Print root1,root2

   Else

6. if deter == 0, roots are real and equal
   6.1 root1 = root2 = -b / (2 * a)
   6.2 print root1,root2

   Else

7. if deter < 0, roots are complex complex and different
   7.1 real = -b / (2 * a)
   7.2 imaginary = sqrt(-deter) / (2 *a)
   7.3 print "complex_roots",real,imaginary
8. end

```
//Program-1
// To find roots of Quadratic Equation
import java.io.*;
import java.lang.*;
import java.util.Scanner;


public class QuadraticEquation {

     public static void main(String[] args) {
          // TODO Auto-generated method stub
Scanner cin=new Scanner(System.in);

          // value a, b, and c
          double a = 2.3, b = 4, c = 5.6;
          System.out.println("Enter Values of a,b,c:  ");
          a=cin.nextDouble();
          b=cin.nextDouble();
          c=cin.nextDouble();
          double root1, root2;

          // calculate the determinant (b2 - 4ac)
          double determinant = b * b - 4 * a * c;

          // check if determinant is greater than 0
          if (determinant > 0) {

            // two real and distinct roots
            root1 = (-b + Math.sqrt(determinant)) / (2 * a);
            root2 = (-b - Math.sqrt(determinant)) / (2 * a);

            System.out.format("root1 = %.2f and root2 = %.2f",
root1, root2);
          }

          // check if determinant is equal to 0
          else if (determinant == 0) {

            // two real and equal roots
```

```
            // determinant is equal to 0
            // so -b + 0 == -b
            root1 = root2 = -b / (2 * a);
            System.out.format("root1 = root2 = %.2f;", root1);
          }

          // if determinant is less than zero
          else {

            // roots are complex number and distinct
            double real = -b / (2 * a);
            double imaginary = Math.sqrt(-determinant) / (2 *
a);
            System.out.format("root1 = %.2f+%.2fi", real,
imaginary);
            System.out.format("\nroot2 = %.2f-%.2fi", real,
imaginary);
          }

      }

}
```

Output:

```
Enter Values of a,b,c:
2.4 4.5 8.9
Roots are Complex and Distinct

   root2 = -0.94-1.68i

Enter Values of a,b,c:
3.0
10.0
5.0
Roots are Real and Distinct
root1 = -0.61 and root2 = -2.72
```

2. Program to Find HCF / GCD OF GIVEN NUMBERS

Aim:- Implement Java Program to find HCF / GCD of Given Numbers using static Functions.

Description:-

In Java, a static method is a method that belongs to a class rather than an instance of a class. ... A static method is not part of the objects it creates but is part of a class definition. Unlike instance methods, a static method is referenced by the class name and can be invoked without creating an object of class

Static function is declared with keyword static

Syntax:

Static return_type function_ name(p1,p2…)

{

Return value;

}

It is the highest number that completely divides two or more numbers.

It is abbreviated for **GCD**. It is also known as the **Greatest Common Factor** (GCF) and the **Highest Common Factor** (HCF). It is used to simplify the fractions.

For Example: Find the GCF of 12 and 8.

Solution:

Factors of **12**: 1, 2, 3, 4, 6, 12
Factors of **8:** 1, 2, 4, 8
Common Factors: 1, 2, 4
Greatest Common Factor: 4
Hence, the GCF of 12 and 8 is 4.

Algorithm:-

Step 1: Start
Step 2: Read number A
Step 3: Read number B
Step 4: If A>B then
A=A-B;
Else
B=B-A;
end if
Step 5: repeat step 3 until A!=B
Step 6: print “GCD is , B
Step 7: Stop

//Program-2

// Java program to find GCD/HCF of given two numbers

```
import java.io.*;
import java.util.Scanner;
public class GCDHCF {
	static  int findGCD(int n1,int n2)
	{
		while (n1!=n2)
		{
		if(n1 >n2)
		n1= n1-n2;
		else
		n2= n2-n1;
		}
		return n2;
	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner cin=new Scanner(System.in);
		int n1,n2;
		System.out.println("Enter two numbers");
		n1=cin.nextInt();
		n2=cin.nextInt();
		System.out.println("GCD/HCF of "+n1 +"  "+n2+"  is  "+findGCD(n1,n2));



	}
}
```

Output:

```
Enter two numbers

28
45
GCD/HCF of 28  45  is  1
```

Enter two numbers

27 36

GCD/HCF of 27 36 is 9

4. Program to demonstrate Polymorphism

Aim: Implement Java program using polymorphism how Msoffice is using polymorphic methods Open,New,Print etc.

Description:

Polymorphism is a feature of object-oriented programming languages that allows a specific routine to use variables of different types at different times. Polymorphism is the ability of a programming language to present the same interface for several different underlying data types.

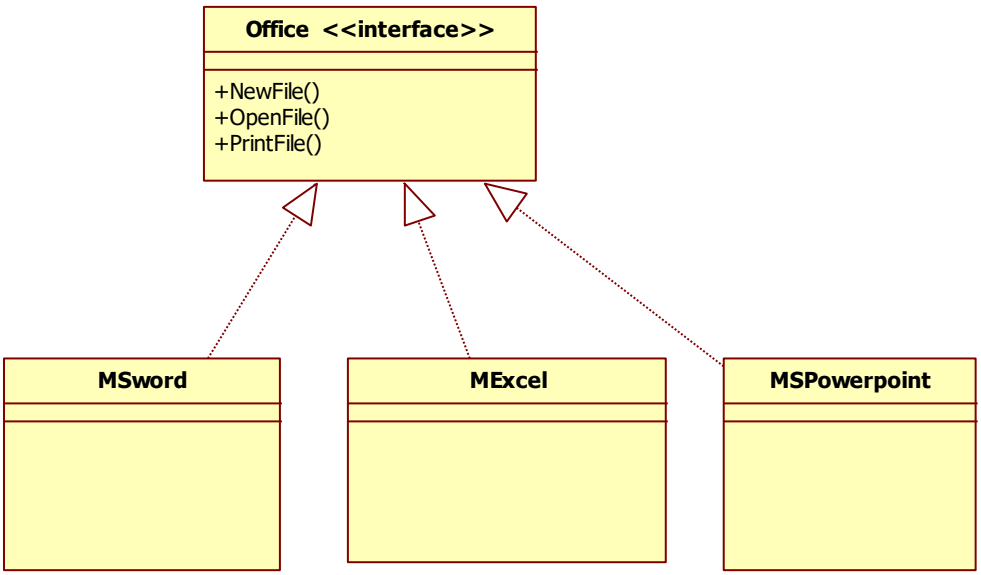
It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding. Method overriding, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class

Polymorphism is achieved by using polymorphic methods in interface, which is defined as

```
Interface interface_name
{
   Abstractmethod1();
   Abstractmethod2();
……..
}
```

Algorithm:

1. Begin
2. Declare interface Office with polymorphic methods like open,new and print.
3. Create classes MSWord,MSExcel,MSPowerpoint with overriding methods open,new and print by implementing interface Office.
4. Display menu , asking user to choose option, which class want to implement polymorphism and call new, open and print methods.
5. Create a refvar of interface , which will take reference of object of the class implementing polymorphism
6. End.


Office <<interface>>
+NewFile()
+OpenFile()
+PrintFile()
MSword
MExcel
MSPowerpoint

```java
// Program implementing polymorphism
import java.io.*;
import java.lang.*;
import java.util.Scanner;

interface Office
{
     void newFile();  // polymorphic method
     void openFile();  // polymorphic method
     void printFile(); //polymorphic method
}

class MSWord implements Office
{

     @Override
     public void newFile() {
          // TODO Auto-generated method stub
          System.out.println("Creates a New Document");
     }

     @Override
     public void openFile() {
          // TODO Auto-generated method stub
          System.out.println("Opens a Document");
     }

     @Override
     public void printFile() {
          // TODO Auto-generated method stub
          System.out.println("Prints a Document");
     }

}
class MSExcel implements Office
{

     @Override
     public void newFile() {
```

```
            // TODO Auto-generated method stub
            System.out.println("Creates a New SpreadSheet");
      }

      @Override
      public void openFile() {
            // TODO Auto-generated method stub
            System.out.println("Opens a SpreadSheet");
      }

      @Override
      public void printFile() {
            // TODO Auto-generated method stub
            System.out.println("Prints a SpreadSheet");
      }

}

class MSPowerpoint implements Office
{

      @Override
      public void newFile() {
            // TODO Auto-generated method stub
            System.out.println("Creates a New Presentation");
      }

      @Override
      public void openFile() {
            // TODO Auto-generated method stub
            System.out.println("Opens a Presentation");
      }

      @Override
      public void printFile() {
            // TODO Auto-generated method stub
            System.out.println("Prints Slides with
presentation");
      }
```

```
}
public class InterfaceDemo {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Office refvar;
		Scanner cin=new Scanner(System.in);
		int opt=0;
		while(opt!=4) {

			System.out.println("Menu");
			System.out.println("-----");
			System.out.println("1.Work with MSword\n2.Work
with Excel\n3.Work with Powerpoint");
			System.out.println("4.Exit \n  Choose your
option....");
			opt=cin.nextInt();
		switch (opt)
		{
		case 1:
			MSWord w=new MSWord();
			refvar=w;
			refvar.newFile();
			refvar.openFile();
			refvar.printFile();
			break;
		case 2 :

			MSExcel e=new MSExcel();
			refvar=e;
			refvar.newFile();
			refvar.openFile();
			refvar.printFile();
			break;
		case 3:
			MSPowerpoint p=new MSPowerpoint();
			refvar=p;
```

```
                refvar.newFile();
                refvar.openFile();
                refvar.printFile();
                break;
            }
            }
            System.out.println("Exiting program.....");

    }

}
```

Output:

```
Menu
-----
1.Work with MSword
2.Work with Excel
3.Work with Powerpoint
4.Exit
  Choose your option....
1
Creates a New Document
Opens a Document
Prints a Document
Menu
-----
1.Work with MSword
2.Work with Excel
3.Work with Powerpoint
4.Exit
  Choose your option....
3
Creates a New Presentation
Opens a Presentation
Prints Slides with presentation
Menu
-----
1.Work with MSword
2.Work with Excel
3.Work with Powerpoint
4.Exit
  Choose your option....
4
Exiting program.....
```

4. Program to Create Package for statistical functions

Aim: Implement package in Java that contains class Statistics and use it .

Description:

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for: Preventing naming conflicts. It is like a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to different folders on your computer.

To create your own package, you need to understand that Java uses a file system directory to store them. Just like folders on your computer. We have two types of packages in Java: **built-in packages and the packages we can create (also known as user defined package).**

To create a package, use the **package** keyword at the beginning of the class or an interface to be part of the package.

Procedure for creating a New package in Eclipse

You can use the New Java Package wizard to create a Java package. The Java Package wizard can be opened in different ways −

By clicking on the File menu and selecting New → Package.

By right click in the package explorer and selecting New → Package.

By clicking on the package icon which is in the tool bar(Package Icon).

```
// user-defined package containing class statistics
package MyPackage;
public class Statistics {

     public double averageArray(int[] a) {
          int s=0;
          for(int i=0;i<a.length;++i)
          {
               s=s+a[i];
          }
          return((double)s/a.length);

     }
     public void StdDeviation(int[] iArray) {
          int s=0;
          int meanAdd=0;
          double mean=0.0;
          double SDDiff=0.0,SD=0.0;
          int n=iArray.length;
          for (int i = 0 ; i < iArray.length ; i++)
          {
               meanAdd = meanAdd + iArray[i];
          }
          mean = meanAdd/n;
          System.out.println("Mean is "+mean);
          for (int i = 0; i < n; i++)
          {
                    SDDiff += (iArray[i] - mean) *  (iArray[i]
- mean);
          }
          double variance = (double)SDDiff / n;
          System.out.println("Variance is "+variance);
          SD = Math.sqrt(variance);
          System.out.println("Standard Deviation is "+SD);


     }

}
```

```
// Main class using above package
// Program to create USE statistics Package
import java.io.*;
import java.util.Scanner;
import MyPackage.*;

public class PackageClass {

      public static void main(String[] args) {
          // TODO Auto-generated method stub
           Scanner cin=new Scanner(System.in);
Statistics ob=new Statistics();
int[] a= {4,5,20,90,10};
System.out.println("Enter 5 Array Elements");
for(int i=0;i<5;++i)
{
     a[i]=cin.nextInt();
}
System.out.println("Average of an Array is "+
ob.averageArray(a));
ob.StdDeviation(a);
     }

}
```

Output:

```
Enter 5 Array Elements
20
10
5
7
8
Average of an Array is 10.0
Mean is 10.0
Variance is 27.6
Standard Deviation is 5.253570214625479


Enter 5 Array Elements
34
53
17
87
90
Average of an Array is 56.2
Mean is 56.0
Variance is 826.2
Standard Deviation is 28.743694960808362
```

5. Program to find Minimum and Maximum of an ARRAY

Aim:- Implement in Java using arrays to find Minimum & Maximum Numbers

Description:-

Array is a collection of elements of similar datatype.
Array is java is declared as:

```
datatype[] a=new Datatype[size];
Or
datatype a[]=new Datatype[size];
```

In this program in a given list of Elements we have to find Minimum & Maximum Numbers and print the result from the list .

Example

Input: Input array elements: 10, 50, 12, 16, 2

Output:
Maximum = 50
Minimum = 2

Algorithm:-

Step 1: Start
Step 2 : Read array of elements A
Step 3: call getMax(A)
Step 3: call getMin(A)
Step 5: stop

Algorithm for Procedure getMax(A)

Step 1: Assume first element as minimum elementmin = A[0]
Step 2: for(I=0 to Alen)
2.1 if(A[i]<min) then min = a[I]
end if
end for
Step 3: print min
Step 4: return

Algorithm for Procedure getMin(A)

Step 1: Assume first element as maximum elementmax = A[0]
Step 2: for(I=0 to Alen)
21 if (A[I]>max) then max = a[j]
end if end for
Step 3: print max
Step 4: return

```
// Java Program to find Minimum & Maximum Values in an Array
import java.io.*;
class GFG {
static int getMin(int arr[], int n)
{
int res = arr[0];
for (int i = 1; i < n; i++)
res = Math.min(res, arr[i]);
return res;
}
static int getMax(int arr[], int n)
{
int res = arr[0];
for (int i = 1; i < n; i++)
res = Math.max(res, arr[i]);
return res;
}
// Driver code
public static void main (String[] args)
{
int arr[] = { 5, 12, 25, 45, 67, 94, 125 };
int n = arr.length;
System.out.println( "Minimum element" + " of array: " + getMin(arr, n));
System.out.println( "Maximum element" + " of array: " + getMax(arr, n));
}
}
```

Output
Minimum element of array : 1
Maximum element of array: 125

6. Program HashTable Collection

Aim: Implement Hashtable collection in Java, to store Accno and balance values into hash table.

Description:

Java Hashtable class implements a hashtable, which maps keys to values. It inherits Dictionary class and implements the Map interface.

A Hashtable is an array of a list. Each list is known as a bucket. The position of the bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key.

Java Hashtable class contains unique elements.

Java Hashtable class doesn't allow null key or value.

Java Hashtable class is synchronized.

Creation of HashTable

Hashtable() It creates an empty hashtable having the initial default capacity and load factor.

Hashtable(int capacity) It accepts an integer parameter and creates a hash table that contains a specified initial capacity.

HashTable T = new HashTable();

Algorithm:

Step 1 Begin

Step 2 Create HashTable using HashTable class

Step 4: Input Key and value pairs

Step 5: Add to HashTable

Step 6: Input to key to Search

Step 7: Find From HashTable for key and display value.

Step 8 : End.

```java
//Program to store names and balances in Hashtable
import java.util.*;
public class HashTable {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		    // Create a hash map
		    Hashtable balance = new Hashtable();
		    Enumeration names;
		    String str;
		    double bal;

		    balance.put("Arun", new Double(3434.34));
		    balance.put("Kamal", new Double(123.22));
		    balance.put("Ayan", new Double(1378.00));
		    balance.put("Daisy", new Double(99.22));
		    balance.put("Harini", new Double(-219.08));

		    // Show all balances in hash table.
		    names = balance.keys();

		    while(names.hasMoreElements()) {
		      str = (String) names.nextElement();
		      System.out.println(str + ": " + balance.get(str));
		    }
		    System.out.println();

		    // Deposit 1,000 into Zara's account
		    bal = ((Double)balance.get("Kamal")).doubleValue();
		    balance.put("Kamal", new Double(bal + 1000));
		    System.out.println("Kamals's new balance: " + balance.get("Kamal"));
		  }
		}
```

Arun: 3434.34

Daisy: 99.22

Kamal: 123.22

Harini: -219.08

Ayan: 1378.0

Kamals's new balance: 1123.22

7. Program using Threads

Aim: Implement Java Program to create Threads using Thread class and Runnable Interface.

Description:

Thread is a light weight Process. It is created by using

1. Thread Class
2. Runnable interface.

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface. Thread is created by using constructors

Thread()
Thread(String name)
Thread(Runnable r)

Some Life cycle methods and other methods in Thread class are

public void run(): is used to perform action for a thread.

public void start(): starts the execution of the thread.JVM calls the run() method on the thread.

public void sleep(long miliseconds): Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

public void join(): waits for a thread to die.

public void join(long miliseconds): waits for a thread to die for the specified miliseconds.

Runnable interface:

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run().

public void run(): is used to perform action for a thread.

Starting a thread:

The start() method of Thread class is used to start a newly created thread.

Algorithm:

1. Begin
2. Create thread to display Fibonacci series by extending Thread Class.
3. In Run Method()
4. 4.1 Initialize fn=0,sn=1
   4.2 Repeat steps below until I <=20.
   4.3 tn=sn+tn
   4.4 Print tn
   4.5 fn=sn, sn=tn
   4.6 increment i
   End loop
5 End

1. Begin
2. Create Thread to display primes to by implementing Runnable interface
3. In Run Method()
4. 4.1 c=0
   4.2 repeat steps until i<=20
   4.3 repeat steps until j <=20
   If(I %j)==0)
   c++
   increment j;
   end loop step 4.3
   4.4 if c==2
   Print “Prime”,i
   4.5 increment i
   4.6 end loop step 4.2
5. end

```
//Program using Threads printing Fibbonacii series and Prime
Numbers
import java.io.*;
import java.lang.*;
class MyThread1 extends Thread
{
     int fn=0,sn=1,tn;

     public void run()
     {
          try {

          for(int i=1;i<=20;i++)
          {
               tn=fn+sn;
               System.out.println("From Thread-Fib  :"+ tn);
               fn=sn;
               sn=tn;
               sleep(1000);
          }
          }
          catch(Exception e)
          {
               System.out.println(e);
          }
     }
}
class MyThread2 implements Runnable
{
     int c=0;
     public void run()
     {
       try {

          for(int i=2;i<=30;i++)
          {
               c=0;
               for(int j=1;j<=i;++j) {
                    if(i%j==0)
                         c++;
               }
```

```
                if(c==2)
            System.out.print("From Thread-Prime: "+ i +"    ");
                Thread.sleep(1000);
            }
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
        }
}


public class ThreadDemo {

    public static void main(String[] args)throws Exception {
        // TODO Auto-generated method stub
        MyThread1 t1=new MyThread1();
        t1.start();

        MyThread2 t2=new MyThread2();

        Thread TR=new Thread(t2);
        TR.start();

    }

}
```

OUTPUT:

```
From Thread-Fib  :1
From Thread-Prime: 2
From Thread-Prime: 3
From Thread-Fib  :2
From Thread-Fib  :3
From Thread-Fib  :5
From Thread-Prime: 5
From Thread-Fib  :8
From Thread-Fib  :13
From Thread-Prime: 7
From Thread-Fib  :21
From Thread-Fib  :34
From Thread-Fib  :55
From Thread-Prime: 11   From Thread-Fib  :89
From Thread-Fib  :144
From Thread-Prime: 13   From Thread-Fib  :233
From Thread-Fib  :377
From Thread-Fib  :610
From Thread-Fib  :987
From Thread-Fib  :1597
From Thread-Prime: 17   From Thread-Fib  :2584
From Thread-Prime: 19   From Thread-Fib  :4181
From Thread-Fib  :6765
From Thread-Fib  :10946
From Thread-Prime: 23   From Thread-Prime: 29
```

8. Program using Exception Handling

Aim: To implement Java program for Handling Custom Exceptions.

Description:
An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. When an error occurs within a method, the method creates an object and hands it off to the runtime system. When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

Java try block is used to enclose the code that might throw an exception. It must be used within the method.
Syntax:
try{

//code that may throw an exception

}catch(Exception_class_Name ref)
{
Exception handling code
}

Java provides Java.lang.Exception class for handling the exceptions which inherit the properties and methods of Object and Throwable class. The Exception class has a set of sub-classes for handling different types of exceptions such as IOException, NotBoundException, and NotOwnerException etc.other exception include

ArthimeticException is raised when number is divided by zero.

IOException is raised when reading or writing into input /ouput device is failed or not responding.

FileNotFoundException is raised when file is not found for reading or writing.

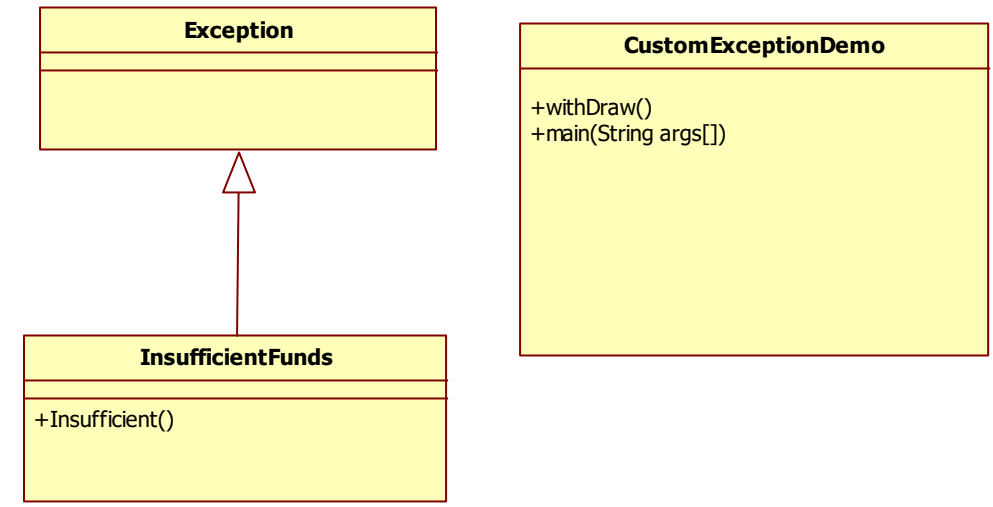
NullPointerException is raised when accessing object which is not created or not initialized.

Throw The Java throw keyword is used to throw an exception explicitly.We specify the exception object which is to be thrown.

Throws The Java throws keyword is used to declare an exception. It gives an information to the programmer that there may occur an exception.

Algorithm:
Step-1 Begin
Step-2 Create class custom exception class Insufficient Funds which extends Exception.
Step-3 Create method withdraw
3.1 accept amt
3.2 check if(bal-amt <500), raise/throw Insufficient funds exception.
Step-4 create customer objects and call withdraw method and check for Exception, if raised handle it.
Step-5 end.

Class Diagram (draw with pencil)



```
// Custom Exception Handling
import java.io.*;
import java.net.*;

class InsufficientFunds extends Exception
{
     InsufficientFunds(String s)
     {
          super(s);
     }
}

public class CustomExceptionDemo {
     int accno;
     double bal;

 CustomExceptionDemo(int a, double b)
 {
      accno=a;
      bal=b;
 }

 void withdraw(double tamt)
 {
      try {
           checkAmount(tamt);
           bal=bal-tamt;
           System.out.println("\n"+"Account no :" +accno+ "
Balance available  after Transaction..."+bal);
      }
      catch(InsufficientFunds e)
      {
           System.out.println(e);
      }
 }

 void checkAmount(double amt)throws InsufficientFunds{
          if((bal-amt)<500.0)
```

```
            throw new InsufficientFunds("For Accountno : " +accno
+"  Funds not Available- Transaction cannot be completed"+"
Balance is  "+bal);
          else
            System.out.println("\nTransaction can be processed
Amount "+amt + " can be withdrawn");
        }

     public static void main(String[] args) {
          // TODO Auto-generated method stub
        //For customer-1
          try {
          CustomExceptionDemo obj=new
CustomExceptionDemo(190111,8000.00);
          obj.withdraw(8000);
           //For customer-2
          CustomExceptionDemo obj2=new
CustomExceptionDemo(229011,89000.00);
          obj2.withdraw(80000.00);
          int y=100;
          int x=y/0;
          }
          catch(ArithmeticException e)
          {
               System.out.println(e);
          }
          catch(Exception ex)
          {
               System.out.println(ex);
          }
          finally {
               System.out.println("This code is executed at
end..... of the program");
          }

     }
}
```

Output:

```
InsufficientFunds: For Accountno : 190111  Funds not Available-
Transaction cannot be completed Balance is  8000.0

Transaction can be processed  Amount 80000.0 can be withdrawn

Account no :229011  Balance available  after
Transaction...9000.0

java.lang.ArithmeticException: / by zero

This code is executed at end..... of the program
```

9. Program to Count Lines and Words in File

Aim: Implement Java Program To Count number of line, words from a given file name give as a Command Line arguments.

Description:

A stream is a sequence of data. In Java, a stream is composed of bytes. A named location on secondary storage used to store related information is known as a File. There are several File Operations like creating a new File, getting information about File, writing into a File, reading from a File and deleting a File. Stream oriented data is handle by Stream classes in java and character oriented data is handled by Reader and Writer classes in java.io package.

Program will count number of lines and words from a file given at java command line. As

C:\> java progname filename.txt

Following classes are used

FileReader: Java FileReader class is used to read data from the file. It returns data in byte format like FileInputStream class. It is character-oriented class which is used for file handling in java.

FileReader(String file) It gets filename in string. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.

int read() It is used to return a character in ASCII form. It returns -1 at the end of file.

void close() It is used to close the FileReader class.

Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast.

int read() It is used for reading a single character.

int read(char[] cbuf, int off, int len) It is used for reading characters into a portion of an array.

String readLine() It is used for reading a line of text.

StringTokenizer : The string tokenizer class **allows an application to break a string into tokens**. The tokenization method is much simpler than the one used by the StreamTokenizer class.

Methods

boolean hasMoreTokens() It checks if there is more tokens available.

String nextToken() It returns the next token from the StringTokenizer object.

String nextToken(String delim) It returns the next token based on the delimiter.

boolean hasMoreElements() It is the same as hasMoreTokens() method.

Algorithm:

1. Begin
2. Use Filename = args[0]; filename is give at command line argument.
3. Open Filename using BufferedReader and FileReader class as
   BufferedReader reader = new BufferedReader(new FileReader(filename));
4. Set line_counter=0;
5. While not eof(reader)
   5.1 Read(line) from file
   5.2 Increment line_counter++;
   5.3 Split line into Tokens by using StingTokenizer class
   5.4 Words_count=count_Tokens
   End of loop step:5
6. Print "Number of lines ", line_counter
7. Print "Number of Words",word_count
8. End

```
// program to count number of lines and words in  a give file name
import java.io.*;
import java.util.*;
public class ReaderExample {
   public static void main(String[] args) {
      try {
         int c=0;
         BufferedReader reader = new BufferedReader(new FileReader(args[0]));
         StringTokenizer stz;
         int wc=0;
        String data = reader.readLine();
        System.out.println("Contents of file is.....\n");
         while (data != null) {
            stz=new StringTokenizer(data);
            wc=wc+stz.countTokens();
            c++;
            System.out.print(data);
            data = reader.readLine();
         }
         System.out.println("\n\n No. of Lines in a File: "+c +"\n No. of Words:  "+wc);
         reader.close();
      } catch (Exception ex) {
         System.out.println(ex.getMessage());
      }
   }
}
```

Output:

C:\jdk1.8\bin>java ReaderExample file.txt

Contents of file is.....

Java Reader is an abstract class for reading character streams.The java command-line argument is an argument i.e. passed at the time of running the java program.The arguments passed from the console can be received in the java program and it can be used as an input.

No. of Lines in a File: 3

No. of Words: 48

C:\jdk1.8\bin>

10. GUI Application using Swings to store Employ data to Database

Aim: Implement Java program in Swings to calculate Employ pay details and store data into database using JDBC.

Description:

Java Swing package javax.swing is used to add Swing classes to program. It is part of JFC. For Database Connectivity, JDBC classes are in java.sql package, so it is to be imported.

Following classes/components are used to create GUI.

JFrame -- To create a window for application.

setTitle(String)— to set text at title bar of a window.

setSize(x,y)-to set width and height of application windows.

setVisible(true)- to make window visible to user.

JLabel -- To create a non-editible text .to diplay message.

JTextField –To create text field to enter input values by user.

setText()- to set/assign text to the text boxes.

getText()- to access text entered by users in text fields.

JButton()- to intiate some action or to run code to compute salary details.

JComboBox – It is combination of list and textfield used to show list of options and also supports to enter values.

addActionListener()--
ActionListener-The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent.
actionPerformed(ActionEvent)- method in interface to add logic/code to do some action when button is pressed on user interface screen.

JDBC—DBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. Java.sql package contains classes/interface like DriverManager, Connection, ResultSet, Statement to store data.

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

1. Register the Driver class
2. Create connection
3. Create statement
4. Execute queries
5. Close connection

Algorithm:

```
Step 1:  Begin
Step 2:  Create GUI Window by extending JFrame class
Step 3:  Add ActionListener Interface to handle Events.
Step 4:  Add Label l1 to l8 using JLabel class to design
form as in figure.
Step 5:  As in Step 5 create TextFields t1 to t7 using
JTextField class.
Step 6:  Create Button b1,b2 using JButton class and add to
Frame.
Step 7:  Register Buttons with ActionListener.
Step 8:  In Listener Function actionPerformed  add following
code.
Function actionPerfomed() for  Calculate Button.
          1.  If grade=”A” then
               Da=110% of Basic
               Hra = 85% of Basic
               Ded = 45% of Deduction
             Else If grade=”B” then
               ……..
          Else If grade=”C” then
          ……
          …
           2. Update Components in Form with calculated
              Values.
 For  New Button ()
            Clear all TextField values using
                       Text.setText(“ “);

For Store Employ Button

     a). Load JDBC driver
     b). Get Connection to MySql Database TESTDB
```

```
     c). Create PreparedStatement
     d). Prepare Query (Insert Statement)
     e). setfields in Table
     f). executequery for SQL Insert Statement
     g). Close connection
Step 9:   Set FrameSize
Step 10:  Set Visible to True

Step 11: end.
```

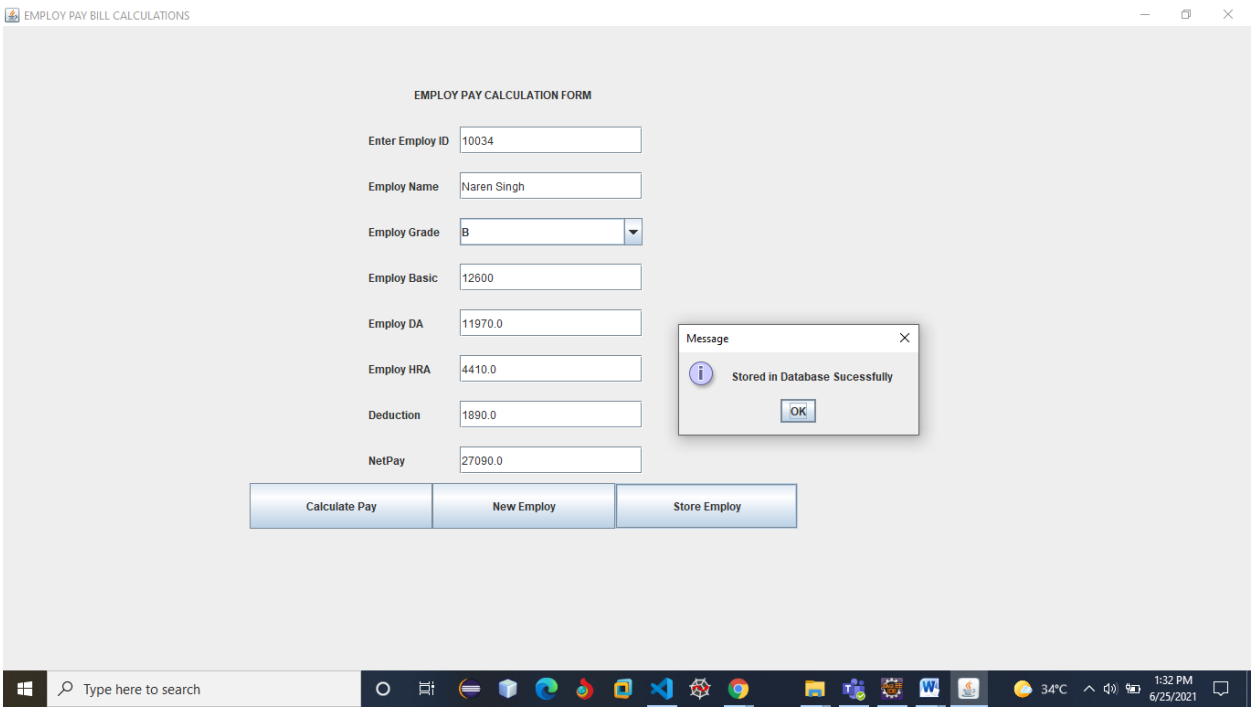
```
//Program to store Employ Data into MySQl Database using
JDBC
import javax.swing.*;
import java.sql.*;
import java.awt.event.*;

public class SwingProgram extends JFrame implements
ActionListener {

      JLabel l1;
      JLabel l2,l3,l4,l5,l6,l7,l8,l9;
      double da=0.0,hra=0.0,ded=0.0,npay=0.0;
          double basic=0.0;
          String gr=null;
      JTextField t1,t2,t4,t5,t6,t7,t8;
      JComboBox t3;
      JButton b1,b2,b3;
     SwingProgram()
     {
          l1=new JLabel("EMPLOY PAY CALCULATION FORM");
          l2= new JLabel("Enter Employ ID");
          l3=new JLabel("Employ Name");
          l4=new JLabel("Employ Grade");
          l5=new JLabel("Employ Basic");
          l6=new JLabel("Employ DA");
          l7=new JLabel("Employ HRA");
          l8=new JLabel("Deduction");
          l9=new JLabel("NetPay");

          t1=new JTextField();
          t2=new JTextField();
          t3=new JComboBox();
          t4=new JTextField();
          t5=new JTextField();
          t6=new JTextField();
          t7=new JTextField();
          t8=new JTextField();

          t3.addItem("A");
```

```
t3.addItem("B");
t3.addItem("C");
t3.setEditable(true);

b1=new JButton("Calculate Pay");
b2=new JButton("New Employ");
b3=new JButton("Store Employ");
setTitle("EMPLOY PAY BILL CALCULATIONS");
add(l1);add(l3);add(l4);add(l5);
add(l6);add(l7);add(l8);add(l9);
add(t2);add(t3);add(t4);add(t5);
add(t6);add(t7);add(t8);
add(b1);add(b2);add(b3);
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
l1.setBounds(450, 50, 200, 50);
add(l2);add(t1);
l2.setBounds(400, 100, 200, 50);
t1.setBounds(500, 110,200,30);

l3.setBounds(400, 150, 200, 50);
t2.setBounds(500,160,200,30);

l4.setBounds(400, 200, 200, 50);
t3.setBounds(500, 210,200,30);

l5.setBounds(400, 250, 200, 50);
t4.setBounds(500,260,200,30);

l6.setBounds(400, 300, 200, 50);
t5.setBounds(500,310,200,30);

l7.setBounds(400, 350, 200, 50);
t6.setBounds(500,360,200,30);

l8.setBounds(400, 400, 200, 50);
t7.setBounds(500,410,200,30);
```

```
        l9.setBounds(400, 450, 200, 50);
        t8.setBounds(500,460,200,30);

        b1.setBounds(270,500,200,50);
        b2.setBounds(470,500,200,50);
        b3.setBounds(670,500,200,50);

        setLayout(null);
        setSize(500,500);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae)
    {

        String cm=ae.getActionCommand();
        if (cm.equals("New Employ")){
             t1.setText("");
             t2.setText("");
             t4.setText("0.0");
             t5.setText("0.0");
             t6.setText("0.0");
             t7.setText("0.0");
             t8.setText("0.0");

        }
        else if(cm.equals("Calculate Pay")){
        basic=Double.parseDouble(t4.getText());
         gr=t3.getSelectedItem().toString();

        switch(gr)
        {
        case "A" : case "a" :
                                  da=basic * (110.0/100.0);
                                  hra=basic * (45.0/100.0);
                                  ded=basic *(20.0/100.0);
                                  break;
        case "B" : case "b" :
                                 da=basic * (95.0/100.0);
```

```
                    hra=basic * (35.0/100.0);
                    ded=basic *(15.0/100.0);
                    break;
            case "C" : case "c" :
            da=basic * (85.0/100.0);
            hra=basic * (30.0/100.0);
            ded=basic *(15.0/100.0);
                    break;

            }}
            else if (cm.equals("Store Employ")){

                try {

                    Class.forName("com.mysql.cj.jdbc.Driver");  //
Loading driver

                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb","
root","Admin2k19");
String sql="insert into employ values(?,?,?,?,?,?,?,?)";
                    PreparedStatement stmt=con.prepareStatement(sql);
// Statement to execute at database

     stmt.setInt(1,Integer.parseInt(t1.getText()));
                    stmt.setString(2,t2.getText());
                    stmt.setDouble(3, basic);
                    stmt.setString(4,gr);
                    stmt.setDouble(5,da);
                    stmt.setDouble(6,hra);
                    stmt.setDouble(8, npay);
                    stmt.setDouble(7, ded);
                    int n=stmt.executeUpdate();
     JOptionPane.showMessageDialog(this, "Stored in Database
Sucessfully");
                }
                catch(Exception e)
                {
                    JOptionPane.showMessageDialog(this,
e.toString());
                }
```

```
            }

            npay=basic+da+hra-ded;
            t5.setText(new Double(da).toString());
            t6.setText(new Double(hra).toString());
            t7.setText(new Double(ded).toString());
            t8.setText(new Double(npay).toString());

      }

            public static void main(String[] args) {

            // TODO Auto-generated method stub
      new SwingProgram();
       }

}
```



mysql> select * from employ;

+-------+-------------+----------+-------+----------+----------+----------+----------+

| empid | ename | salary | grade | da | hra | ded | netpay |

+-------+-------------+----------+-------+----------+----------+----------+----------+

| 9800 | Naren | 78900.00 | A | 86790.00 | 35505.00 | 15780.00 | 185415.00 |

| 10034 | Naren Singh | 12600.00 | B | 11970.00 | 4410.00 | 1890.00 | 27090.00 |

+-------+-------------+----------+-------+----------+----------+----------+----------+

2 rows in set (0.07 sec)

11. GUI Application using Swings to find Employ pay

Aim: Implement Java program in Swings to calculate Employ pay details.

Description:

Java Swing package javax.swing is used to add Swing classes to program. It is part of JFC. For Database Connectivity, JDBC classes are in java.sql package, so it is to be imported.

Following classes/components are used to create GUI.

JFrame -- To create a window for application.

setTitle(String)— to set text at title bar of a window.

setSize(x,y)-to set width and height of application windows.

setVisible(true)- to make window visible to user.

JLabel -- To create a non-editible text .to diplay message.

JTextField –To create text field to enter input values by user.

setText()- to set/assign text to the text boxes.

getText()- to access text entered by users in text fields.

JButton()- to intiate some action or to run code to compute salary details.

JComboBox – It is combination of list and textfield used to show list of options and also supports to enter values.

addActionListener()--
ActionListener-The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent.
actionPerformed(ActionEvent)- method in interface to add logic/code to do some action when button is pressed on user interface screen.

Algorithm:

Step 1: Begin
Step 2: Create GUI Window by extending JFrame class
Step 3: Add ActionListener Interface to handle Events.
Step 4: Add Label l1 to l8 using JLabel class to design form as in figure.
Step 5: As in Step 5 create TextFields t1 to t7 using JTextField class.
Step 6: Create Button b1,b2 using JButton class and add to Frame.
Step 7: Register Buttons with ActionListener.
Step 8: In Listener Function actionPerformed add following code.
Function actionPerfomed() for Calculate Button.

1. If grade=”A” then
   Da=110% of Basic
   Hra = 85% of Basic
   Ded = 45% of Deduction
   Else If grade=”B” then
   ……..
   Else If grade=”C” then
   ……
   …
2. Update Components in Form with calculated Values.

For New Button ()
Clear all TextField values using
Text.setText(“ “);

Step 9: Set FrameSize
Step 10: Set Visible to True
Step 11: end.

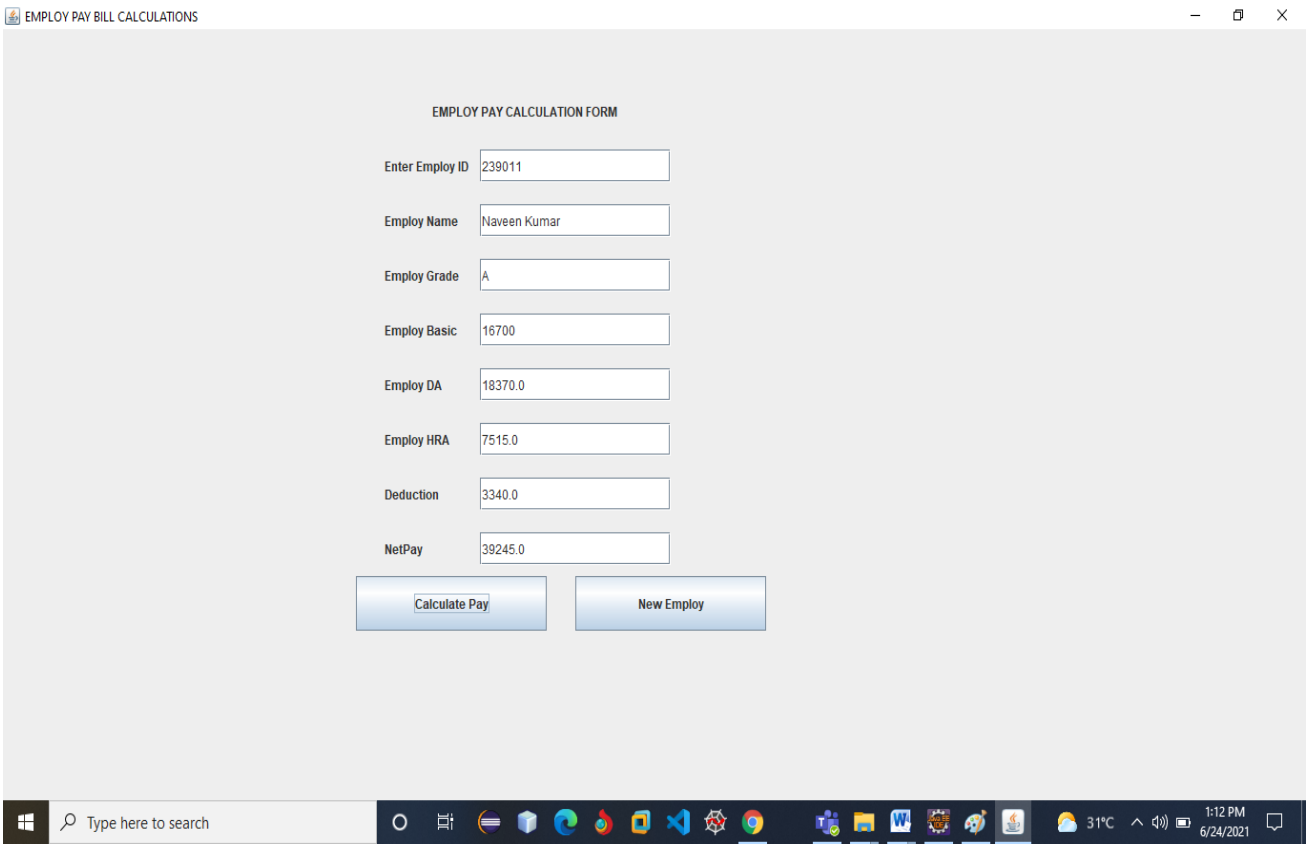
```
//Program for Employ Pay Bill Calculations using Swing
//Controls
import javax.swing.*;
import java.awt.event.*;

public class SwingProgram extends JFrame implements
ActionListener {

      JLabel l1;
      JLabel l2,l3,l4,l5,l6,l7,l8,l9;

      JTextField t1,t2,t3,t4,t5,t6,t7,t8;
      JButton b1,b2;
     SwingProgram()
     {
          l1=new JLabel("EMPLOY PAY CALCULATION FORM");
          l2= new JLabel("Enter Employ ID");
          l3=new JLabel("Employ Name");
          l4=new JLabel("Employ Grade");
          l5=new JLabel("Employ Basic");
          l6=new JLabel("Employ DA");
          l7=new JLabel("Employ HRA");
          l8=new JLabel("Deduction");
          l9=new JLabel("NetPay");

          t1=new JTextField();
          t2=new JTextField();
          t3=new JTextField();
          t4=new JTextField();
          t5=new JTextField();
          t6=new JTextField();
          t7=new JTextField();
          t8=new JTextField();

          b1=new JButton("Calculate Pay");
          b2=new JButton("New Employ");
          setTitle("EMPLOY PAY BILL CALCULATIONS");
          add(l1);add(l3);add(l4);add(l5);
          add(l6);add(l7);add(l8);add(l9);
```

```
        add(t2);add(t3);add(t4);add(t5);
        add(t6);add(t7);add(t8);
        add(b1);add(b2);
        b1.addActionListener(this);
        b2.addActionListener(this);
        l1.setBounds(450, 50, 200, 50);
        add(l2);add(t1);
        l2.setBounds(400, 100, 200, 50);
        t1.setBounds(500, 110,200,30);

        l3.setBounds(400, 150, 200, 50);
        t2.setBounds(500,160,200,30);

        l4.setBounds(400, 200, 200, 50);
        t3.setBounds(500, 210,200,30);

        l5.setBounds(400, 250, 200, 50);
        t4.setBounds(500,260,200,30);

        l6.setBounds(400, 300, 200, 50);
        t5.setBounds(500,310,200,30);

        l7.setBounds(400, 350, 200, 50);
        t6.setBounds(500,360,200,30);

        l8.setBounds(400, 400, 200, 50);
        t7.setBounds(500,410,200,30);

        l9.setBounds(400, 450, 200, 50);
        t8.setBounds(500,460,200,30);

        b1.setBounds(370,500,200,50);
        b2.setBounds(600,500,200,50);

        setLayout(null);
        setSize(500,500);
        setVisible(true);
    }
```

```
public void actionPerformed(ActionEvent ae)
{
     String cm=ae.getActionCommand();
     if (cm.equals("New Employ")){
          t1.setText("");
          t2.setText("");
          t3.setText("");
          t4.setText("0.0");
          t5.setText("0.0");
          t6.setText("0.0");
          t7.setText("0.0");
          t8.setText("0.0");

     }
     else{

     double basic=Double.parseDouble(t4.getText());
     String gr=t3.getText();
     double da=0.0,hra=0.0,ded=0.0,npay=0.0;
     switch(gr)
     {
     case "A" : case "a" :
                              da=basic * (110.0/100.0);
                              hra=basic * (45.0/100.0);
                              ded=basic *(20.0/100.0);
                              break;
     case “B" : case "b" :
                    da=basic * (95.0/100.0);
          hra=basic * (35.0/100.0);
                              ded=basic *(15.0/100.0);
                                   break;
     case "C" : case "c" :
                              da=basic * (85.0/100.0);
                              hra=basic * (30.0/100.0);
                              ded=basic *(15.0/100.0);
                                   break;

     }
```

```
            npay=basic+da+hra-ded;
            t5.setText(new Double(da).toString());
            t6.setText(new Double(hra).toString());
            t7.setText(new Double(ded).toString());
            t8.setText(new Double(npay).toString());
            }
      }

            public static void main(String[] args) {

            // TODO Auto-generated method stub
      new SwingProgram();
      }

}
```

KMMIPS::TIRUPATI



12. Applet Program in Java Using Parameters

Aim: Implement a Java Program using Applets to display text send to applet by Parameter in different Text and Fonts.

Description:

Applet is a JAVA program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Applet is created by using applet class in java.applet package. Applet is executed in BROWSER OR in APPLETVIEWER.

Applet class contains following methods

public void init(): is used to initialized the Applet. It is invoked only once.

public void start(): is invoked after the init() method or browser is maximized. It is used to start the Applet.

public void stop(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.

public void destroy(): is used to destroy the Applet. It is invoked only once.

paint() method: in java.awt.Graphics class is used to draw or to display on Applet as it is a panel window container.

java.awt.Graphics class provides many methods for graphics programming.

public abstract void drawString(String str, int x, int y): is used to draw the specified string.

public void drawRect(int x, int y, int width, int height): draws a rectangle with the specified width and height.

public abstract void setColor(Color c): is used to set the graphics current color to the specified color.

public abstract void setFont(Font font): is used to set the graphics current font to the specified font.

Applet Tag :

To embed applet in HTML we use following TAG.

```
<applet code="APPLETCLASS NAME" width="300" height="300">
   <param name="message" value="HelloWorld" >
</applet>
```

Algorithm:

1. Begin
2. Create an appletclass that ie.. Extends Applet class.
3. Create Font objects f1,f2
4. Initialize f1= new Font(“Serif”,”Bold”,34); and also f2.
5. In Paint Method() set Font and Color by using setFont() and setColor() methods
6. Use getParameter() to get values of parameter from applet tag in html file.
7. In Paint method use drawstring method to display string.
8. End.

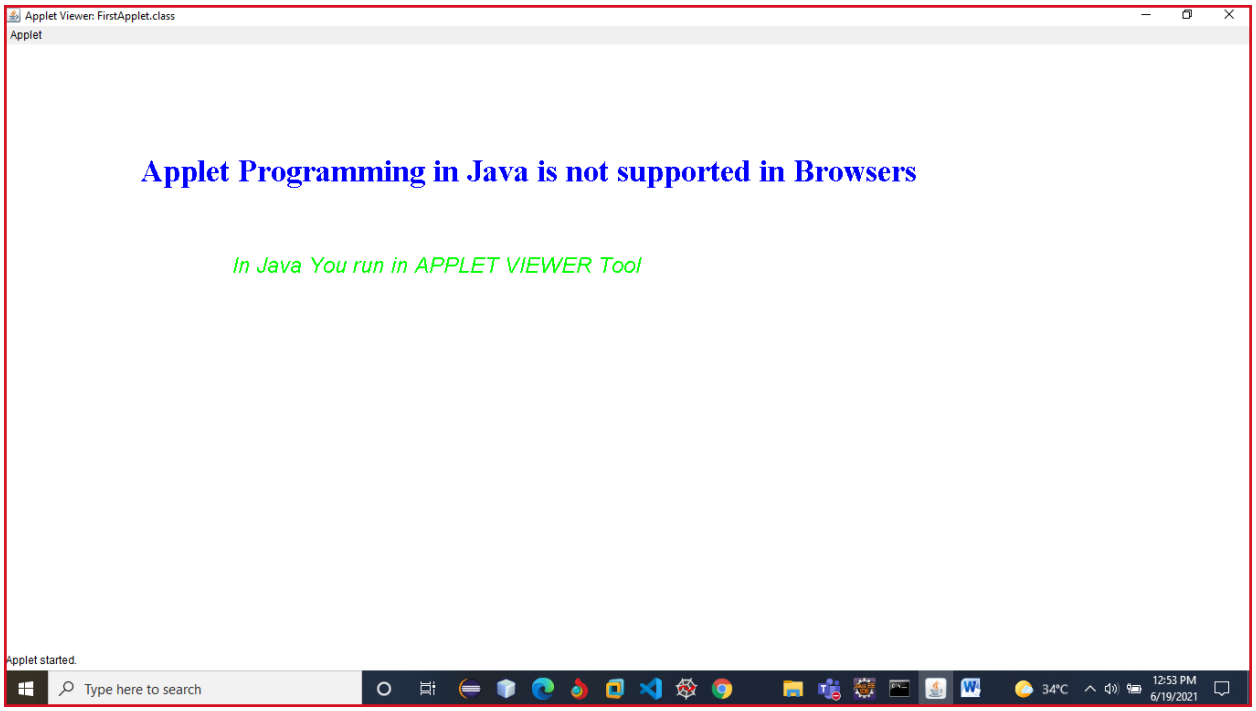
// Applet Code

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.*;
import javax.swing.*;
public class FirstApplet extends JApplet{
   Font f1,f2;
   public void init()
   {
      f1=new Font("TimesRoman",Font.BOLD,34);
      f2=new Font("Arial",Font.ITALIC,24);

   }

public void paint(Graphics g){
   String str1=getParameter("msg1");
   String str2=getParameter("msg2");
   g.setFont(f1);
   g.setColor(Color.blue);
   g.drawString(str1,150,150);
   g.setFont(f2);
   g.setColor(Color.green);
   g.drawString(str2,250,250);
```

```
} }
```

Html code to display Applet

```
<html>

<body>

<applet code="FirstApplet.class" width="300" height="300">

   <param name="msg1" value="Applet Programming in Java is not supported in Browsers">

   <param name="msg2" value="In Java You run in APPLET VIEWER Tool">

</applet>

</body>

</html>
```


Applet Viewer: FirstApplet.class
Applet
Applet Programming in Java is not supported in Browsers
In Java You run in APPLET VIEWER Tool
Applet started.
Type here to search
34°C
12:53 PM
6/19/2021

13. Program to Implement UDP (User DataGram Protocol).

Aim: Implement UDP protocol in Java to Send message to other network device.

Description:

UDP: User Datagram Protocol provides a connection-less protocol service by allowing packet of data to be transferred along two or more nodes. Java provides DatagramSocket to communicate over UDP instead of TCP. It is also built on top of IP. DatagramSockets can be used to both send and receive packets over the Internet.

Creation of DatagramSocket:- , a datagramSocket object is created to carry the packet to the destination and to receive it whenever the server sends any data. To create a datagramSocket following constructors can be used:

public DatagramSocket() throws SocketException

public DatagramSocket(int port) throws SocketException

Parameters: port - port to which socket is to be bound

Creation of DatagramPacket: In this step, the packet for sending/receiving data via a datagramSocket is created.

Constructor to send data: DatagramPacket(byte buf[], int length, InetAddress inetaddress, int port)

Invoke a send() or receive() call on socket object

Syntax: void send(DatagramPacket packet) throws SocketException

For sending a packet via UDP, we should know 4 things, the message to send, its length, ipaddress of destination, port at which destination is listening.

Algorithm: Sender

Step1 : Begin

Step 2: Create DataGramSocket (ds)

step 3 : Get IPAdress of the Receiver

step 4 : Create a String Message to sent to Receiver (msg)

step 5 : Create DataGramPacket dp (Send to IpAddress given in step 3)

step 6: Send datagrampacket (dp) through datagramsocket (ds)

step 7: close socket

step 8 : end

Algorithm : Receiver

Step 1 : Begin

Step 2 : Create DataGramsocket (ds) at Receive with Portno.

Step 3: Create buffer for storing Datagrampacket from Sender

Step 4 : Receive Datagrampacket from Receive to buffer via Socket(of the sender)

Step 5: Extract data from datagrampacket

Step 6: Display data

Step 7: end

```
// Program Receiving data Client
import java.net.*;
 public class DReceiver
{  public static void main(String[] args) throws Exception
{  DatagramSocket ds = new DatagramSocket(6000);
 byte[] buf = new byte[1024];
 DatagramPacket dp = new DatagramPacket(buf, 1024);
 ds.receive(dp);
 String str = new String(dp.getData(), 0, dp.getLength());

System.out.println(str);
 ds.close();
 }
 }
```

```
// Program to send data to Client
import java.net.*;
public class DSender
{
```

```
public static void main(String[] args) throws Exception
{
DatagramSocket ds = new DatagramSocket();
 String str = "Welcome java for 1st MCA";
 InetAddress ip = InetAddress.getByName("127.0.0.1");
 DatagramPacket dp = new DatagramPacket(str.getBytes(), str.l
ength(), ip, 6000);
 ds.send(dp);
System.out.println("Data Send Sucessfully");
 ds.close();  }
 }
```

Output:

C:\jdk1.8\bin>javac DSender.java

C:\jdk1.8\bin>java DSender

Data Send Sucessfully

C:\jdk1.8\bin>

C:\jdk1.8\bin>javac DReceiver.java

C:\jdk1.8\bin>java DReceiver

Welcome java for 1st MCA

14. Program using TCP Protocol for exchange of Data

Aim : Implement Java Program to Communicate between client and Service using TCP protocol.

Description:

TCP: Transmission Control Protocol provides reliable communication between the sender and receiver. TCP is used along with the Internet Protocol referred as TCP/IP

Java uses **java.net package** for implementing TCP AND UDP.

In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

Socket :A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket("127.0.0.1", 5000)

Establish a Socket Connection

To write a server application two sockets are needed.

A ServerSocket which waits for the client requests (when a client makes a new Socket())

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client:
To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Algorithm: Server

1. Begin
2. Create a ServerSocket() object using ServerSocket Class
3. Bind ServerSocket to Port 6666
4. Accept Socket Connection from Client
5. Get DataInputStream From Client Socket to Read Data
6. String msg=socket.read()
7. Write "From Client", msg
8. close Socket()
9. end

Algorithm: CLIENT

1. Begin
2. Create Socket() object using Socket class
3. Connect Socket object to ServerSocket("localhost",6666)
4. Get DataoutputStream of ServerSocket
5. Write /Send data to ServerSocket() to Server
6. Close Socket
7. end.

//Program to implement TCP Server

```
import java.io.*;
import java.net.*;
public class KMMSERVER {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}catch(Exception e){System.out.println(e);}
}
}
```

// PROGRAM TO IMPLEMENT CLIENT

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class StudentClient {
public static void main(String[] args) {
```

```
   Scanner cin=new Scanner(System.in);
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
System.out.println("Enter Message to sent to Server");
String msg=cin.nextLine();
dout.writeUTF(msg);
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
}
```

OUTPUT

AT SERVER WINDOW

C:\jdk1.8\bin>java KMMSERVER

message= hello

C:\jdk1.8\bin>java KMMSERVER

message= hello i am student studying in MCA

AT CLIENT WINDOW

C:\jdk1.8\bin>javac StudentClient.java

C:\jdk1.8\bin>javac StudentClient.java

C:\jdk1.8\bin>java StudentClient

Enter Message to sent to Server

hello i am student studying in MCA

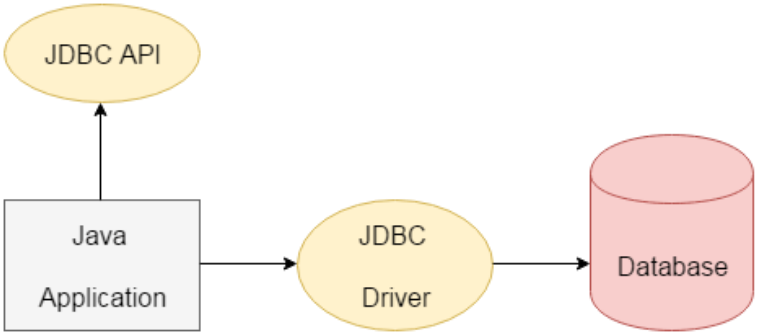
15. JDBC EXAMPLE TO VIEW MYSQL DATA

AIM : Implement JAVA Program using JDBC to insert and view Employ Records.

Description:

In this program we are using JDBC API which is in java.sql package. Following classes are used in the java program to store employ data to MYSQL Database. Mysql Driver is loaded for connection with database.

JDBC - JDBC is a Java API (java.sql) to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.



Algorithm /Steps to Connect to Java Program to JDBC:

Step 1) Register the driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of forName() method

Class.forName("com.mysql.cj.jdbc.Driver")

Step 2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb","root","Admin2k19");

Step 3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method

Statement stmt1=con.createStatement();

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

executeQuery() method

ResultSet rs=stmt1.executeQuery("select * from employ");

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method

public void close()throws SQLException

// Java Program to access data from database

```
import java.sql.*;
import java.util.Scanner;
class MysqlInsert{
public static void main(String args[]){
   int eid;
   String ename;
   String dept;
   double salary;
   Scanner cin=new Scanner(System.in);

try{
   // TO INSERT NEW EMPLOY RECORD INTO DATABASE
System.out.println("NEW  EMPLOY  ENTRY");
System.out.println("Enter EmpId");
eid=cin.nextInt();
System.out.println("Enter Emp Name");
ename=cin.next();
System.out.println("Enter Salary");
salary=cin.nextDouble();
System.out.println("Enter Dept");
dept=cin.next();

Class.forName("com.mysql.cj.jdbc.Driver");  // Loading driver

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb","root","Admin2k19");
 String sql="insert into employ values(?,?,?,?)";
PreparedStatement stmt=con.prepareStatement(sql);  // Statement to execute at database
```

```
stmt.setInt(1,eid);

stmt.setString(2,ename);

stmt.setDouble(3,salary);

stmt.setString(4,dept);

int n=stmt.executeUpdate();

System.out.println("Record Stored Sucessfully.....");



// TO VIEW RECORDS FROM EMPLOY TABLE

Statement stmt1=con.createStatement();

ResultSet rs=stmt1.executeQuery("select * from employ");

System.out.println("\nEmploy Records are....\n");

while(rs.next())

{

   System.out.println(rs.getInt("empid")+ "  "+ rs.getString("ename")+ "  "+rs.getDouble("Salary")+"  "
+rs.getString("Dept"));


}

con.close();

}catch(Exception e)

{ System.out.println(e);}

}

}
```

OUTPUT:-

NEW EMPLOY ENTRY

Enter EmpId

6701

Enter Emp Name

KESAVA

Enter Salary

88900.00

Enter Dept

ACCOUNTS

Record Stored Sucessfully.....

Employ Records are....

1001 Smith 11000.0 Sales

1901 Edwards 23900.0 Production

1006 Clarke 16000.0 Sales

1090 Jones 89000.0 Management

1036 KEERTHI 7800.0 Sales

7670 ASWINI 8000.0 Production

6701 KESAVA 88900.0 ACCOUNTS

Conclusion :

Program executed successfully and results are noted as above.

16. TO CONVERT GIVEN PHONE NUMBER IN WORDS TO DIGITS.

Aim: - Write java program to convert given phone number in words to digits using StringTokenizer class.

Description:

Program uses StringTokenizer class to split given string into tokens and converts it into digits.

Following constructors are used:

StringTokenizer(String str) It creates StringTokenizer with specified string.

StringTokenizer(String str, String delim) It creates StringTokenizer with specified string and delimiter.

Methods:-

boolean hasMoreTokens() It checks if there is more tokens available.

String nextToken() It returns the next token from the StringTokenizer object.

boolean hasMoreElements() It is the same as hasMoreTokens() method.

Algorithm:-

Step 1: Begin
Step 2: ReadString(str) (phone number in words)
Step 3: Create Starray[]=StringTokenizer(str)
Step 4: Repeat steps 5 & 6 until starry<>isnotempty
Step 5: compare each arrayelement[i] with words
Step 6: concatenate str with digit(1,2,…)
Step 7: Print str (phone no in digits.)
Step 8: end.

```
// Program to convert words to digits using STRINGTOKENIZER
import java.util.*;
public class WORDTODIGIT {
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            Scanner cin=new Scanner(System.in);
      System.out.println("Enter phone number in words");
            String phword=cin.nextLine();
      StringTokenizer st = new StringTokenizer(phword," ");
            String ph=new String("");
            String dch="";
            String tch="";
           while (st.hasMoreTokens()) {

                  switch(st.nextToken())
                  {
                  case "double":   dch="1";
                                   break;
                  case  "triple":  tch="1";
                                   break;

                  case "one" :
                              if(dch.equals("1")) {
                              ph=ph+"1";
                              dch="";}
                              if(tch.equals("1"))
                                   {ph=ph+"1";
                                    ph=ph+"1";
                                    tch="";}

                              ph=ph+"1";
                                        break;

                  case "two" :
                                        if(dch.equals("1")) {
                                        ph=ph+"2";
                                        dch="";}

                                        if(tch.equals("1"))
                                      {ph=ph+"2";
                                       ph=ph+"2";
                                       tch="";}

                                        ph=ph+"2";
                                             break;
                  case "three" :
                                        if(dch.equals("1")) {
                                        ph=ph+"3";
                                        dch="";}
```

```
                    if(tch.equals("1"))
               {ph=ph+"3";
                ph=ph+"3";
                tch="";}
                    ph=ph+"3";
                         break;
case "four" :
   if(tch.equals("1"))
      {ph=ph+"4";
       ph=ph+"4";
       tch="";}
              if(dch.equals("1")) {
                       ph=ph+"4";
                       dch="";}
                     ph=ph+"4";
                       break;
case "five" :
                    if(dch.equals("1")) {
                         ph=ph+"5";
                         dch="";
                    }

                    if(tch.equals("1"))
               {ph=ph+"5";
                ph=ph+"5";
                tch="";}

                    ph=ph+"5";
                         break;
case "six" :
   if(dch.equals("1")) {
            ph=ph+"6";
            dch="";
   }
   if(tch.equals("1"))
      {ph=ph+"6";
       ph=ph+"6";
       tch="";}
                         ph=ph+"6";
                         break;
case "seven" :
   if(dch.equals("1")) {
                    ph=ph+"7";
                    dch="";
          }
          if(tch.equals("1"))
             {ph=ph+"7";
              ph=ph+"7";
              tch="";}
```

```
                        ph=ph+"7";
                              break;
                  case "eight" :
                        if(dch.equals("1")) {
                        ph=ph+"8";
                        dch="";}

                     if(tch.equals("1"))
                        {ph=ph+"8";
                         ph=ph+"8";
                         tch="";}
                              ph=ph+"8";
                              break;
                  case "nine" :
                  if(dch.equals("1")) {
                        ph=ph+"9";
                        dch="";
                  }
                  if(tch.equals("1"))
                  {ph=ph+"9";
                   ph=ph+"9";
                   tch="";}

                  ph=ph+"9";
                              break;
                  case "zero" :
                              if(dch.equals("1")) {
                                    ph=ph+"0";
                                    dch="";}
                              if(tch.equals("1"))
                        {ph=ph+"0";
                         ph=ph+"0";
                         tch="";}
                              ph=ph+"0";
                              break;
            }
                  }
System.out.println(ph);        }  }
```

OUTPUT:

Enter phone number in words
eight seven six double one two
876112

Enter phone number in words

seven zero triple nine seven five six four

709997564