Creating Your First Application in NETBEANS IDE:

Your first application, HelloWorldApp, will simply display the greeting "Hello World!" To create this program, you will:

- **Create an IDE project**

  When you create an IDE project, you create an environment in which to build and run your applications. Using IDE projects eliminates configuration issues normally associated with developing on the command line. You can build or run your application by choosing a single menu item within the IDE.

- **Add code to the generated source file**

  A source file contains code, written in the Java programming language, that you and other programmers can understand. As part of creating an IDE project, a skeleton source file will be automatically generated. You will then modify the source file to add the "Hello World!" message.

- **Compile the source file into a .class file**

  The IDE invokes the Java programming language *compiler* (javac), which takes your source file and translates its text into instructions that the Java virtual machine can understand. The instructions contained within this file are known as *bytecodes*.

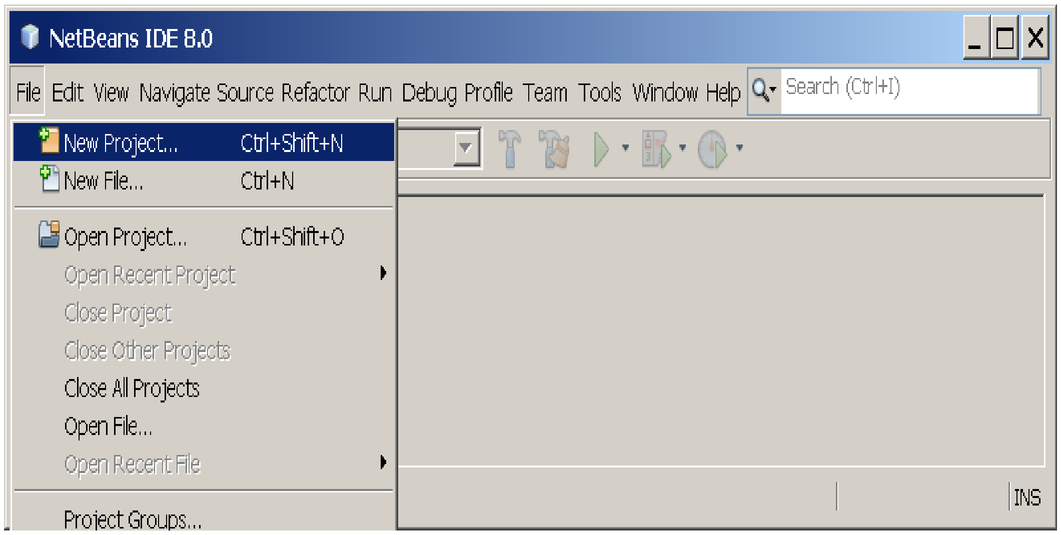
- **Run the program**

  The IDE invokes the Java application *launcher tool* (java), which uses the Java virtual machine to run your application.

Create an IDE Project

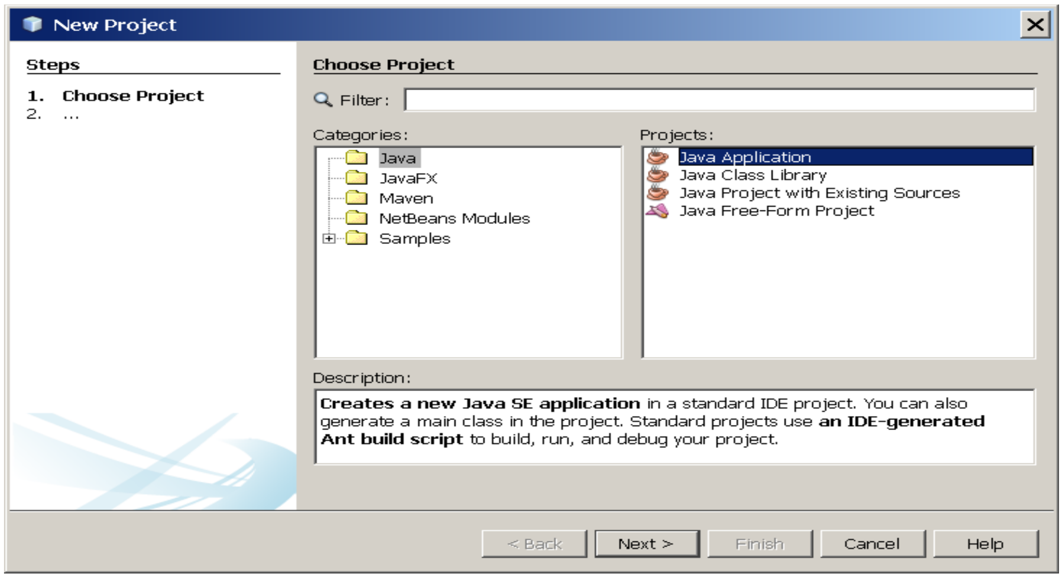
To create an IDE project:

1. Launch the NetBeans IDE.
   - On Microsoft Windows systems, you can use the NetBeans IDE item in the Start menu.
   - On Solaris OS and Linux systems, you execute the IDE launcher script by navigating to the IDE's bin directory and typing./netbeans.
   - On Mac OS X systems, click the NetBeans IDE application icon.
2. In the NetBeans IDE, choose **File | New Project...**.



NetBeans IDE with the File | New Project menu item selected.

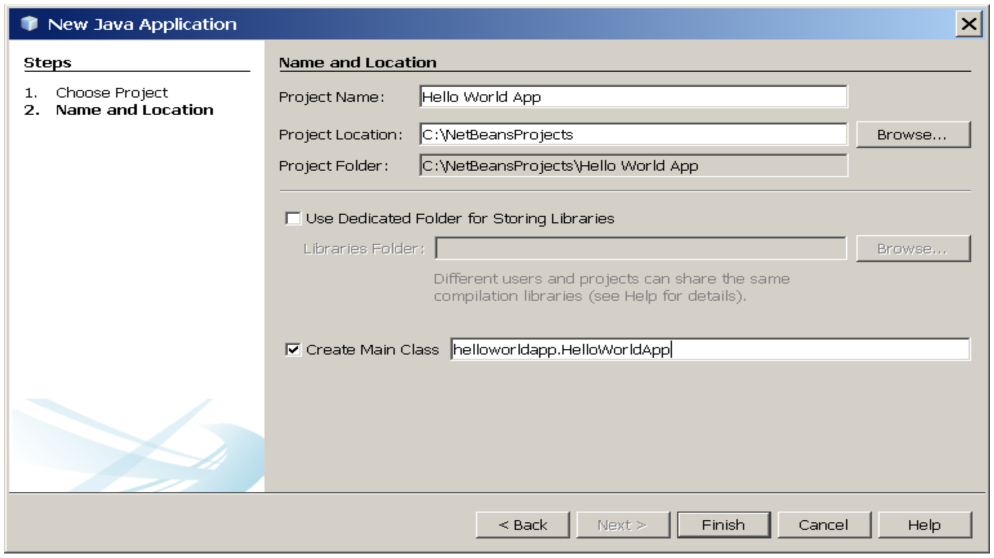
3. In the **New Project** wizard, expand the **Java** category and select **Java Application** as shown in the following figure:



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

NetBeans IDE, New Project wizard, Choose Project page.

4. In the **Name and Location** page of the wizard, do the following (as shown in the figure below):
   - In the **Project Name** field, type Hello World App.
   - In the **Create Main Class** field, type helloworldapp.HelloWorldApp.



Name and Location

Project Name: Hello World App

Project Location: C:\NetBeansProjects Browse...

Project Folder: C:\NetBeansProjects\Hello World App

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

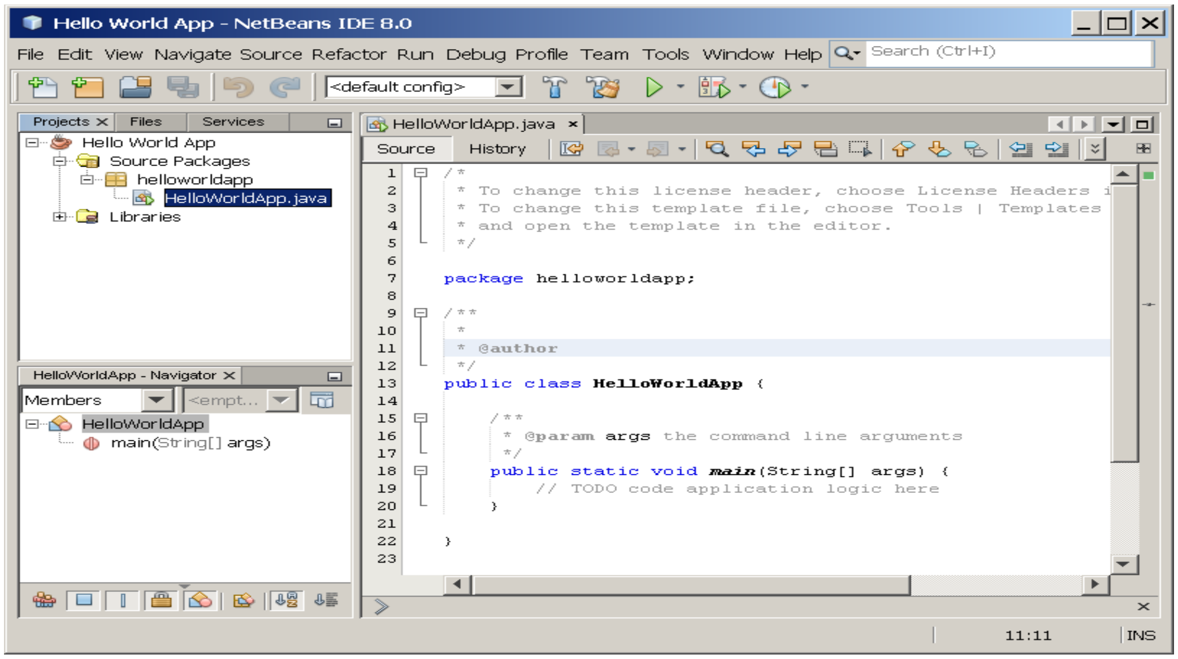
☑ Create Main Class helloworldapp.HelloWorldApp

NetBeans IDE, New Project wizard, Name and Location page.

5. Click Finish.

The project is created and opened in the IDE. You should see the following components:

- The **Projects** window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The **Source Editor** window with a file called HelloWorldApp.java open.
- The **Navigator** window, which you can use to quickly navigate between elements within the selected class.

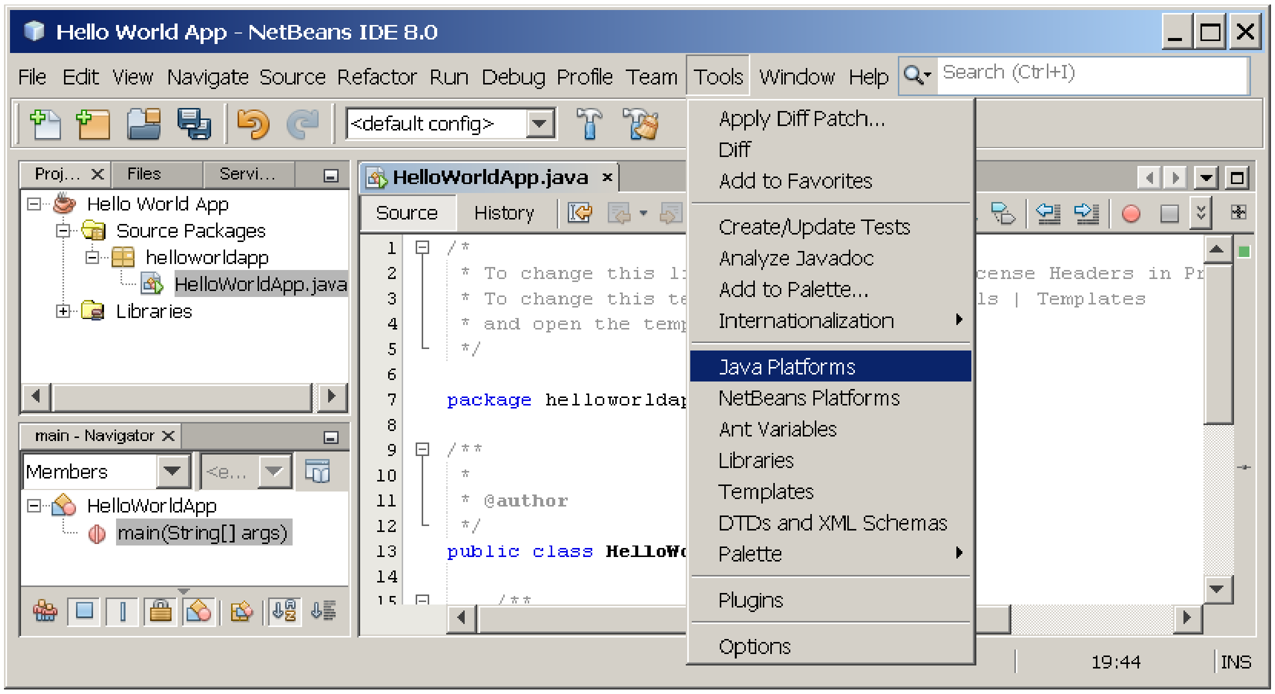


```
/*
 * To change this license header, choose License Headers i
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;

/**
 *
 * @author
 */
public class HelloWorldApp {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

NetBeans IDE with the HelloWorldApp project open.

Add JDK 8 to the Platform List (if necessary)

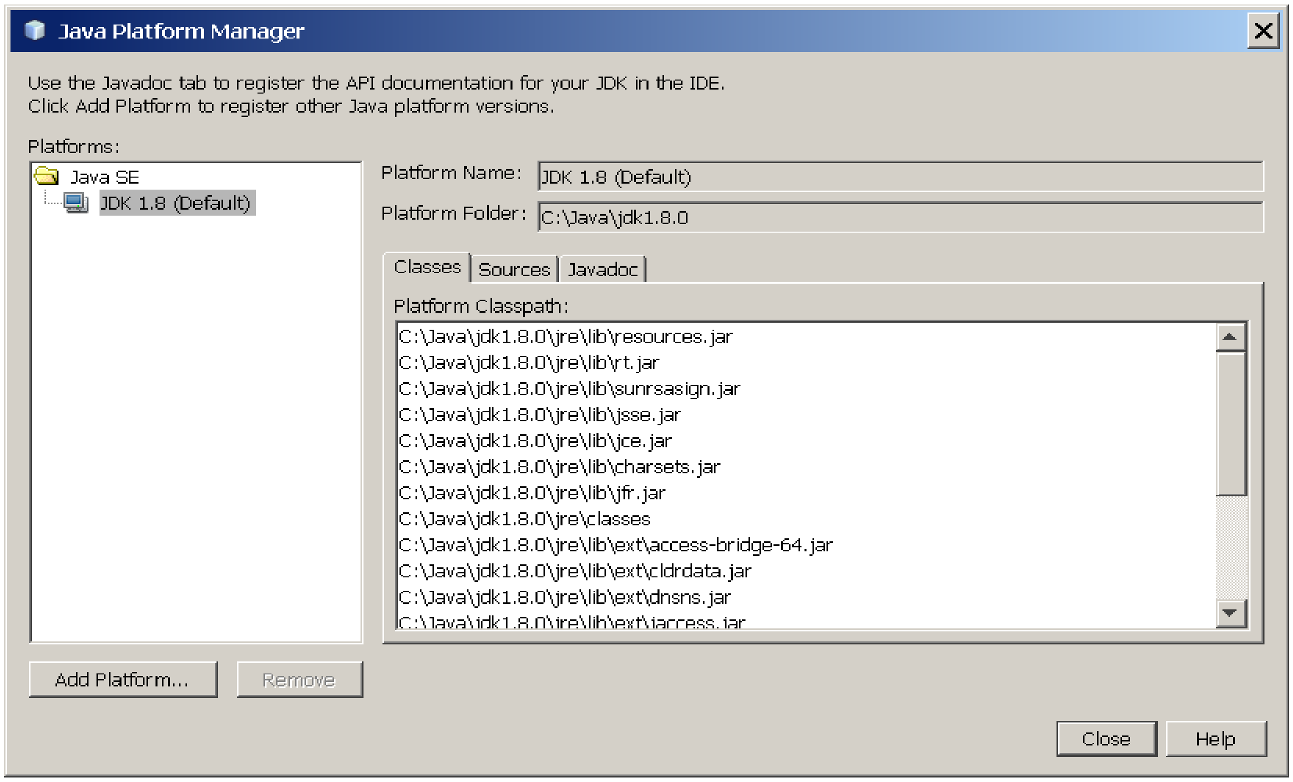
It may be necessary to add JDK 8 to the IDE's list of available platforms. To do this, choose Tools | Java Platforms as shown in the following figure:



```
/*
 * To change this li                cense Headers in Pr
 * To change this te                ls | Templates
 * and open the temp
 */

package helloworldap

/**
 *
 * @author
 */
public class HelloWo

    /**
```

Selecting the Java Platform Manager from the Tools Menu

If you don't see JDK 8 (which might appear as 1.8 or 1.8.0) in the list of installed platforms, click **Add Platform**, navigate to your JDK 8 install directory, and click **Finish**. You should now see this newly added platform:



Use the Javadoc tab to register the API documentation for your JDK in the IDE.
Click Add Platform to register other Java platform versions.

Platforms:
Java SE
JDK 1.8 (Default)

Platform Name: JDK 1.8 (Default)

Platform Folder: C:\Java\jdk1.8.0

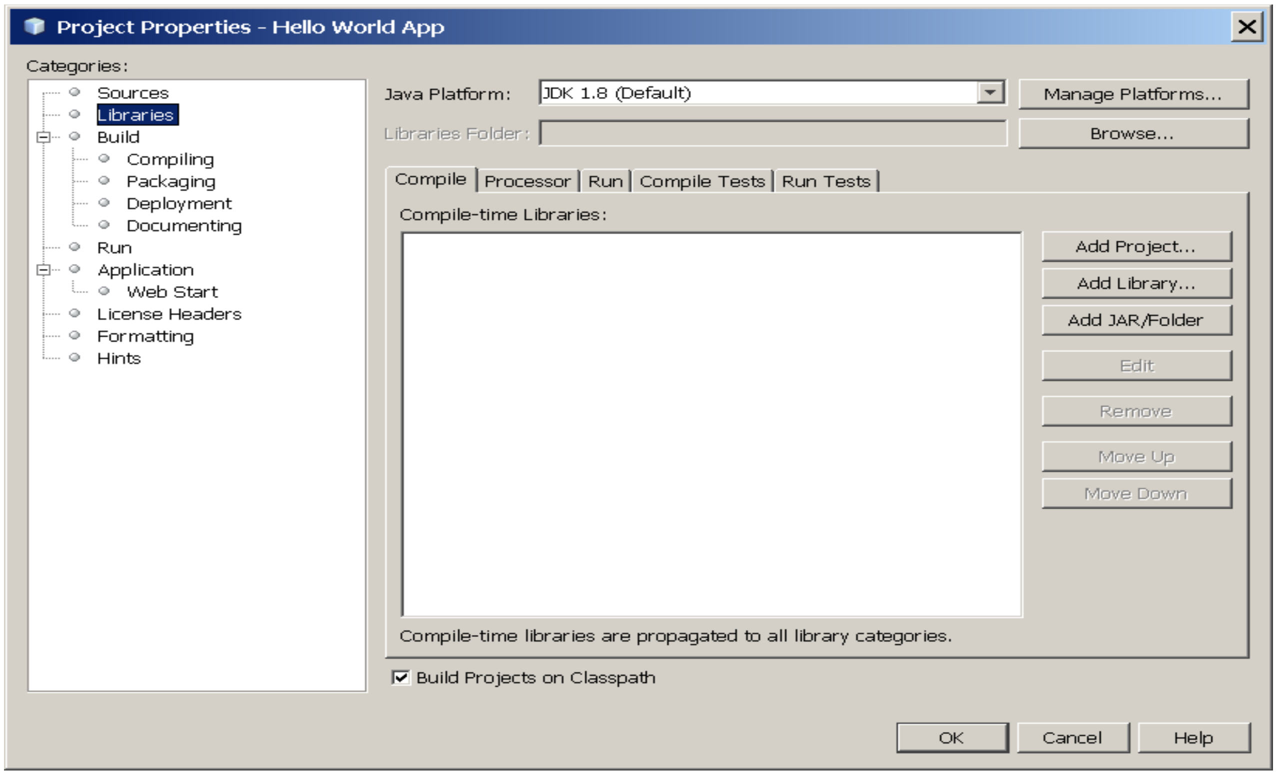
Platform Classpath:

```
C:\Java\jdk1.8.0\jre\lib\resources.jar
C:\Java\jdk1.8.0\jre\lib\rt.jar
C:\Java\jdk1.8.0\jre\lib\sunrsasign.jar
C:\Java\jdk1.8.0\jre\lib\jsse.jar
C:\Java\jdk1.8.0\jre\lib\jce.jar
C:\Java\jdk1.8.0\jre\lib\charsets.jar
C:\Java\jdk1.8.0\jre\lib\jfr.jar
C:\Java\jdk1.8.0\jre\classes
C:\Java\jdk1.8.0\jre\lib\ext\access-bridge-64.jar
C:\Java\jdk1.8.0\jre\lib\ext\cldrdata.jar
C:\Java\jdk1.8.0\jre\lib\ext\dnsns.jar
C:\Java\jdk1.8.0\jre\lib\ext\jaccess.jar
```

The Java Platform Manager

To set this JDK as the default for all projects, you can run the IDE with the --jdkhome switch on the command line, or by entering the path to the JDK in the netbeans_j2sdkhome property of your INSTALLATION_DIRECTORY/etc/netbeans.conf file.

To specify this JDK for the current project only, select **Hello World App** in the **Projects** pane, choose **File** | **Project Properties (Hello World App)**, click **Libraries**, then select **JDK 1.8** in the **Java Platform** pulldown menu. You should see a screen similar to the following:


Project Properties - Hello World App

Categories: Sources, Libraries, Build, Compiling, Packaging, Deployment, Documenting, Run, Application, Web Start, License Headers, Formatting, Hints

Java Platform: JDK 1.8 (Default)

Libraries Folder:

Compile-time Libraries:

Compile-time libraries are propagated to all library categories.

☑ Build Projects on Classpath

The IDE is now configured for JDK 8.

Add Code to the Generated Source File

When you created this project, you left the **Create Main Class** checkbox selected in the **New Project** wizard. The IDE has therefore created a skeleton class for you. You can add the "Hello World!" message to the skeleton code by replacing the line:

// TODO code application logic here with the line:
System.out.println("Hello World!"); // Display the string.

Optionally, you can replace these four lines of generated code:

```
/**
 *
 * @author
 */
```

with these lines:

```
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
```

These four lines are a code comment and do not affect how the program runs. Later sections of this tutorial explain the use and format of code comments.

Be Careful When You Type

Note: Type all code, commands, and file names exactly as shown. Both the compiler (javac) and launcher (java) are *case-sensitive*, so you must capitalize consistently.

HelloWorldApp is *not* the same as helloworldapp.

Save your changes by choosing **File** | **Save**.

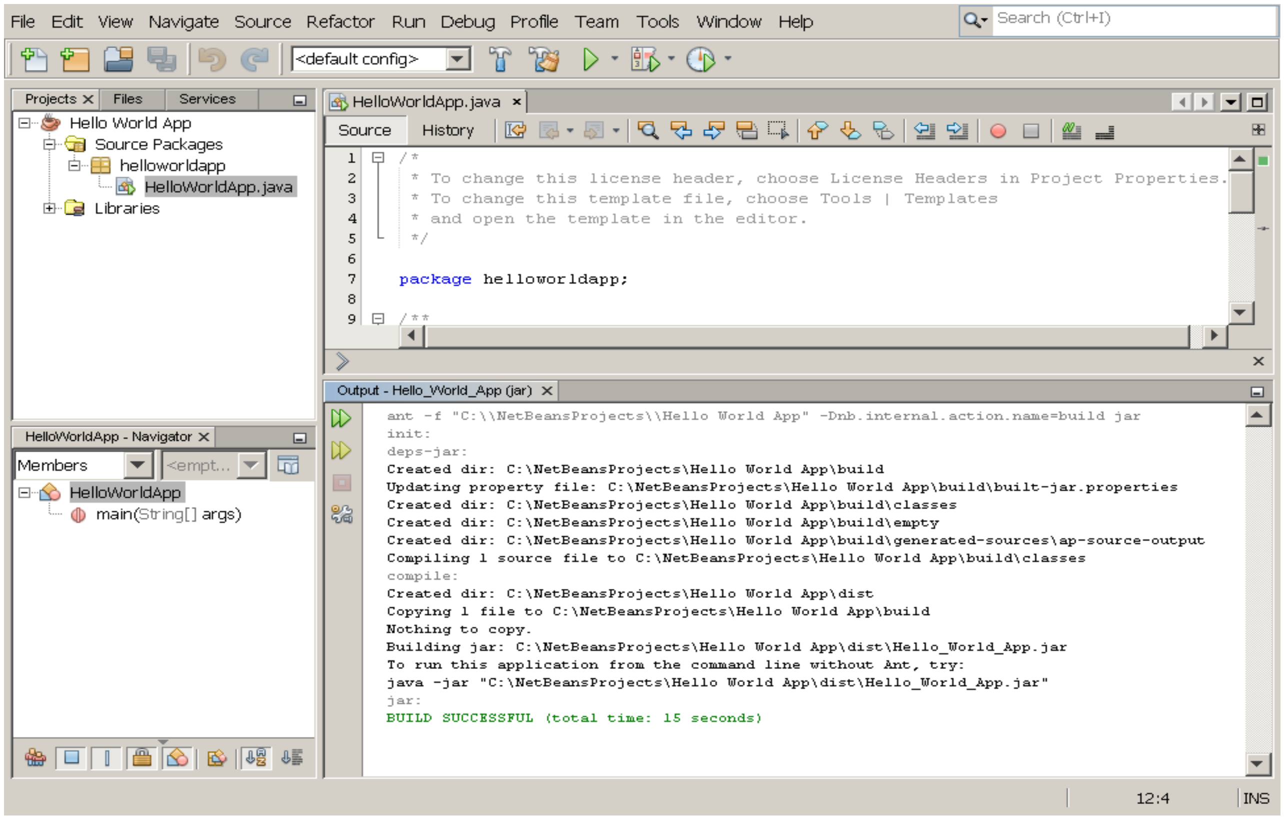
The file should look something like the following:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;

/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */

public class HelloWorldApp {

     /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      System.out.println("Hello World!"); // Display the string.
   }

}
```

Compile the Source File into a .class File

To compile your source file, choose **Run** | **Build Project (Hello World App)** from the IDE's main menu.

The Output window opens and displays output similar to what you see in the following figure:



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;

/**
```

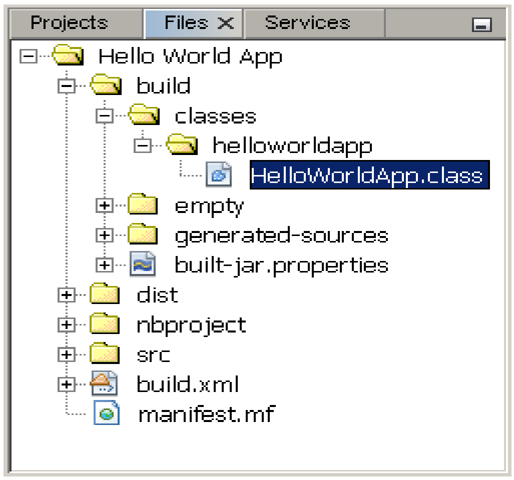
```
ant -f "C:\\NetBeansProjects\\Hello World App" -Dnb.internal.action.name=build jar
init:
deps-jar:
Created dir: C:\NetBeansProjects\Hello World App\build
Updating property file: C:\NetBeansProjects\Hello World App\build\built-jar.properties
Created dir: C:\NetBeansProjects\Hello World App\build\classes
Created dir: C:\NetBeansProjects\Hello World App\build\empty
Created dir: C:\NetBeansProjects\Hello World App\build\generated-sources\ap-source-output
Compiling 1 source file to C:\NetBeansProjects\Hello World App\build\classes
compile:
Created dir: C:\NetBeansProjects\Hello World App\dist
Copying 1 file to C:\NetBeansProjects\Hello World App\build
Nothing to copy.
Building jar: C:\NetBeansProjects\Hello World App\dist\Hello_World_App.jar
To run this application from the command line without Ant, try:
java -jar "C:\NetBeansProjects\Hello World App\dist\Hello_World_App.jar"
jar:
BUILD SUCCESSFUL (total time: 15 seconds)
```

Output window showing results of building the HelloWorld project.

If the build output concludes with the statement BUILD SUCCESSFUL, congratulations! You have successfully compiled your program!

If the build output concludes with the statement BUILD FAILED, you probably have a syntax error in your code. Errors are reported in the Output window as hyperlinked text. You double-click such a hyperlink to navigate to the source of an error. You can then fix the error and once again choose **Run** | **Build Project**.

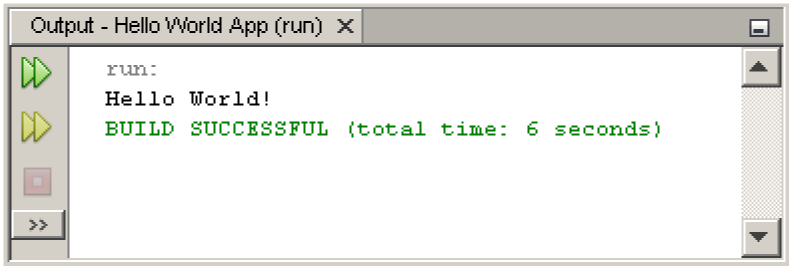
When you build the project, the bytecode file HelloWorldApp.class is generated. You can see where the new file is generated by opening the **Files** window and expanding the **Hello World App/build/classes/helloworldapp** node as shown in the following figure.



Files window, showing the generated .class file.

Now that you have built the project, you can run your program.

Run the Program
From the IDE's menu bar, choose **Run** | **Run Main Project**.
The next figure shows what you should now see.



```
run:
Hello World!
BUILD SUCCESSFUL (total time: 6 seconds)
```

The program prints "Hello World!" to the Output window (along with other output from the build script).

WEEK 1

(A) AIM: To write a Java program that prints all real solutions to the quadratic equation $ax^2 + bx + c = 0$. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions.

THEORY: The Quadratic $\boldsymbol{ax^2 + bx + c}$**.** where "*a* ", "*b*", and "*c*" are just numbers; they are the "numerical coefficients" of the quadratic equation. The term b^2-4ac is known as the determinant of a quadratic equation.

The determinant tells the nature of the roots.

- If determinant is greater than 0, the roots are real and different.
- If determinant is equal to 0, the roots are real and equal.
- If determinant is less than 0, the roots are complex and different

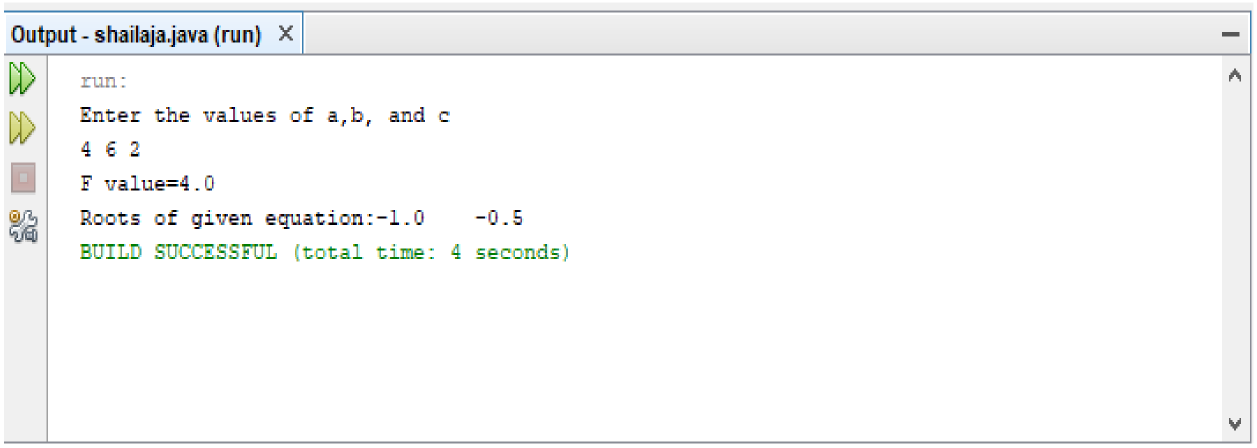
ALGORITHM :

```
STEP1: START
STEP2: Enter the value of a, b and c.
STEP3: Calculate determinant as $D= b^2-4ac$
STEP4: If (D> 0) then continue else 'GOTO Step 9
STEP5: Calculate root1= $-b + \sqrt{D}/2a$
STEP6: Calculate root2= $-b – \sqrt{D}/2a$
STEP7: Print value of root1 and root2 these are Real and different
STEP8: GOTO END
STEP9: If (D==0) then continue else GOTO Step 13
STEP10: Calculate root1 = root2 = -b/2*a
STEP11: Print value of root1 and root2 the roots are real and equal.
STEP12: GOTO END
STEP13: Calculate RealPart = -b/2a
STEP14: Calculate   imaginaryPart = $\sqrt{(-D)}/2a$
STEP15: Print value of root1= RealPart + i imaginaryPart
STEP16: Print value of root2= RealPart – i imaginaryPart the roots are complex and
        different
STEP17: END
```

SOURCE CODE:

```
import java.util.Scanner;

class Roots
        {
                public static void main(String args[])
                {
                        int a,b,c;
                        double x,y;
                        Scanner s=new Scanner(System.in);
                        System.out.println("Enter the values of a,b, and c");
                        a=s.nextInt();
                        b=s.nextInt();
                        c=s.nextInt();
                        double f=(b*b)-4*a*c;
                        System.out.println("F value="+f);
                        if(f<0)
                        {
                                System.out.println("No real roots");
                        }
                        else
                        {
                                double l=Math.sqrt(f);
                                x=((-b-l)/(2*a));
                                y=((-b+l)/(2*a));
                                System.out.println("Roots of given equation:"+x+"\t"+y);
                        }
                }
        }
```

OUTPUT:



```
run:
Enter the values of a,b, and c
4 6 2
F value=4.0
Roots of given equation:-1.0    -0.5
BUILD SUCCESSFUL (total time: 4 seconds)
```

(B)AIM: To write a Java program that prompts the user for an integer and then prints out all prime numbers up to that integer. (use Scanner class to read input)

THEORY: A **prime number** (or a **prime**) is a natural number greater than 1 that cannot be formed by multiplying two smaller natural numbers. A natural number greater than 1 that is not prime is called a composite number. A number which is divided only by 1 and itself is known as the prime number.

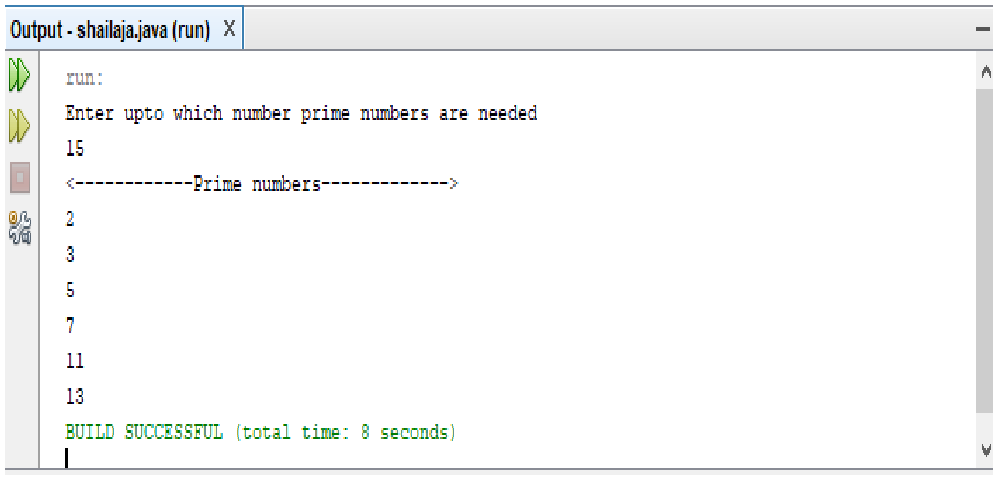
ALGORITHM: Prime Numbers Generations

```
STEP 1: START
STEP 2: Enter the value of n.
STEP 3: for i := 2 to n do
STEP 4: Initialize p to 0
STEP 5: for j := 2 to i / 2 do
STEP 6:  if j divides i
STEP 7: proceed to the next j
STEP 8: p =1,break;
STEP 9:  If  ( p=0)  print i
STEP10: proceed to the next i
STEP11: END
```

SOURCE CODE:

```
import java.util.Scanner;
class a
 {
        public static void main(String[] args)
        {
                int n,p;
                Scanner s=new Scanner(System.in);
System.out.println("Enter upto which number prime numbers are needed");
                n=s.nextInt();
                System.out.println("<------------Prime numbers------------->");
                for(int i=2;i<n;i++)
                {
                        p=0;
                        for(int j=2;j<i/2;j++)
                        {
                                if(i%j==0)
                                p=1;
                        }
                        if(p==0)
                        System.out.println(i);
                }
 }
}
```

OUTPUT:



```
run:
Enter upto which number prime numbers are needed
15
<------------Prime numbers------------->
2
3
5
7
11
13
BUILD SUCCESSFUL (total time: 8 seconds)
```

VIVA VOCE:

1. Difference between Procedure-Oriented and Object-oriented programming

Properties	**Procedure-Oriented programming**	**Object-oriented programming**
Access Modifiers (public, private, protected, Default)	Not Supports	Supports
Exception Handling Mechanisms	No	No
Approach	Top Down Approach.	Bottom Up Approach.
Adding data and Functions	Not Easy.	Easy.
overloading	Not Possible.	Possible

2. D22. Define a class. Write a syntax to create a class in java

Class is a template that is used to create objects, and used to define object data types and methods.

Syntax: public class <class name>

```
{
// instance variable
//static variable/class variable/Global variable
//constructors
//static methods
//blocks
//final methods
//user defined methods
//main method
}
```

3. List Control statements in java.
   Different types of control statements:
   1. Selection/decision making statements (if-then, if-then-else and switch),
   2. Iterative/ looping statements (while, do-while and for) and
   3. jumping/ branching statements (break, continue and return).

4. List different data types in java.
   1.Primitive Data Types: byte, short, int, long, float, double, boolean and char,
   2. Non -Primitive Data Types: String, Arrays , Classes and interfaces

5. List java operators

   1.Arithmetic operators.
   2.Relational operators.
   3.logical operators.
   4.bitwise operators.
   5.Assignment operators (=).
   6. conditional operator (? :).

WEEK 2

(A)AIM: To write a Java program to create a Student class with following fields

i.I. Hall ticket number
i.II. Student Name
i.III. Department

Create 'n' number of Student objects where 'n' value is passed as input to constructor.

THEORY: A constructor is a member function of a class which initializes objects of a class. In java , Constructor is automatically called when object(instance of class) create. It is special member function of the class.

A constructor is different from normal functions in following ways:

- Constructor has same name as the class itself
- Constructors don't have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, java compiler generates a default constructor for us (expects no parameters and has an empty body).

Types of Constructors

1. **Default Constructors:** Default constructor is the constructor which doesn't take any argument. It has no parameters.
2. **Parameterized Constructors:** It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object

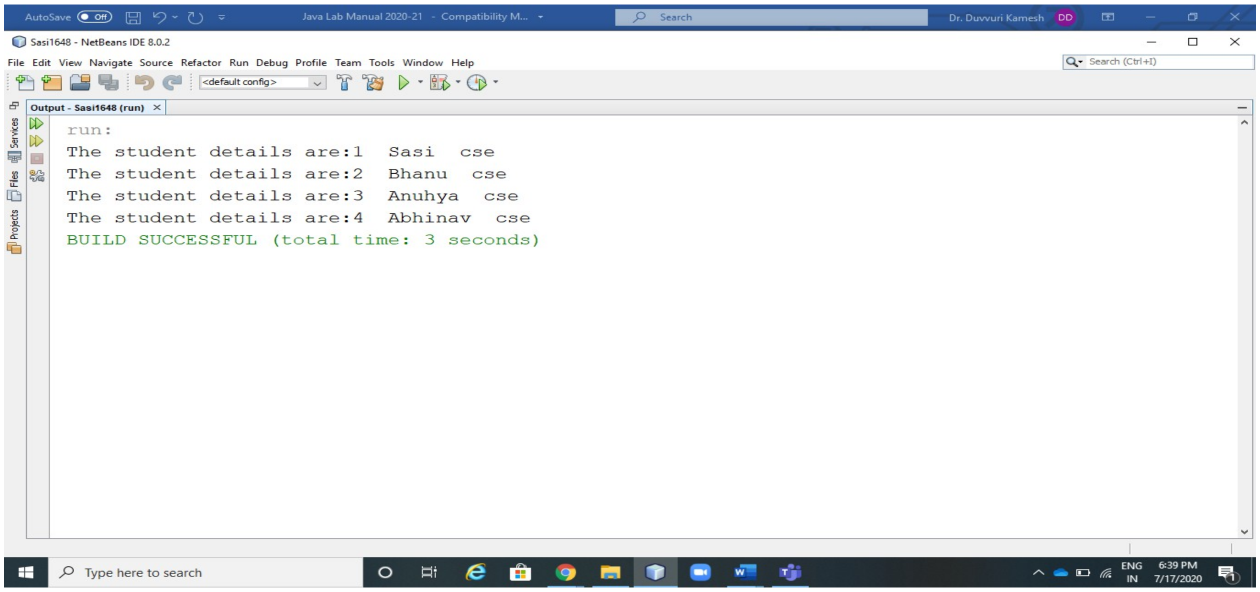
ALGORITHM:

```
STEP 1: START
STEP 2: Create a class called Student
STEP 3: Create 3 instant variables.
STEP 4: Create parameterized constructor which has 3 parameters. Initialize the
instant variables with parameters variables of the constructor.
STEP 5: Create another class which has the main().
STEP 6: Create 'n' number of Student objects where 'n' value is passed as input to
constructor
STEP 7: STOP
```

SOURCE CODE:

```
            class Student
            {
                        int hollticket;
                        String studName;
                        String Department;
            Student(int ht, String sdN, String Dep)
            {
            hollticket=ht;
            studName=sdN;
            Department=Dep;
System.out.print("The student details are:"+hollticket+"  "+studName+"  "+Department+"\n");
             }
            }
          class StudentDetails
            {
                         public static void main(String args[])
                        {
                                    Student obj1 = new Student(1,"Sasi","cse");
                                    Student obj2 = new Student(2,"Bhanu","cse");
                                    Student obj3 = new Student(3,"Anuhya","cse");
                                    Student obj4 = new Student(4,"Abhinav","cse");

                        }
          }
```

OUTPUT:



```
run:
The student details are:1  Sasi  cse
The student details are:2  Bhanu  cse
The student details are:3  Anuhya  cse
The student details are:4  Abhinav  cse
BUILD SUCCESSFUL (total time: 3 seconds)
```

(B)AIM: To Write a Java program to demonstrate string comparison using == and equals method.

Theory: In general, both equals () and "==" operator in Java are used to compare objects to check equality but here are some of the differences between the two:

1. Main difference between. equals () method and == operator is that one is method and other is operator.
2. We can use == operators for reference comparison (**address comparison**) and. equals () method for **content comparison**. In simple words, == checks if both objects point to the same memory location whereas. equals () evaluates to the comparison of values in the objects.

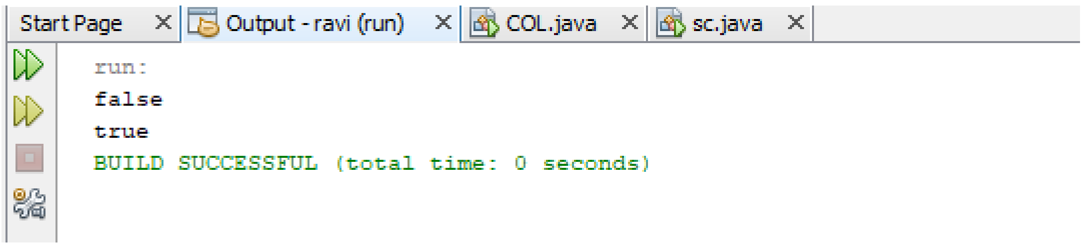
ALGORITHM:

STEP 1: START
STEP 2: Create two objects namely s1 and s2.
STEP 3: Both s1 and s2 refers to different objects.
STEP 4: If s1 == s2 return true as both have same addresses in memory.
STEP 5: else it returns false as both have different addresses in memory.
STEP 6: If Using equals, the result is true because its only comparing the values given in s1 and s2. Otherwise it returns false
STEP 7: STOP

SOURCE CODE

```
public class sc
{
        public static void main(String[] args)
        {
                String s1 = new String("HELLO");
                String s2 = new String("HELLO");
                System.out.println(s1 == s2);
                System.out.println(s1.equals(s2));
        }
}
```

OUTPUT



```
run:
false
true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Explanation

== Operator	**Equals() Method**
It is majorly used to compare the reference values and objects.	It is used to compare the actual content of the object.
We can use the == operator with objects and primitives.	We cannot use the equals method with primitives.
The == operator can't compare conflicting objects, so at that time the compiler surrenders the compile-time error.	The equals() method can compare conflicting objects utilizing the equals() method and returns "false".

VIVA VOCE:

1. Define String class. Write the syntax to find the length of the string.
 A) String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created.
Syntax: int length = StringName.length();

2. What is the difference between equals() and == .
A)
  1. **==** checks if both objects point to the same memory location.
  2. **equals()** evaluates to the comparison of values **in the** objects.

3. Parameterized constructor. Write the Syntax to create a parametrized Constructor.
 A) Parameterized constructor is a constructor with the argument.

```
Class student
{
Student(int a, int b)
{…….}}
```

4. What is Default Constructor? Give its importance in java ,Write the syntax to declare it.
   A) Default constructor contains no arguments for the constructor. .

```
Syntax:  class ClassName{

              ClassName(){ }
               ...
                              }
```

5. Differentiate between Constructor and Function.
   A) Constructors are special type of methods used to initialize objects of the class and to execute instance of a class. Functions are set of instructions that are invoked at any part in a program to return a result and purpose is to execute java code.

WEEK 3

(A)AIM: To Write a Java program that checks whether a given string is a palindrome or not. Example: MADAM is a palindrome.

THEORY: String is a sequence of characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created. A **palindrome** is a word, phrase, number or sequence of words that reads the same backwards as forwards.

Creating a String: There are two ways to create a String in Java
String literal, Using new keyword
String literal : Assigning a String literal to a String instance: Example : String str1 = "Welcome";
Using New Keyword: String str1 = new String("Welcome");

ALGORITHM:

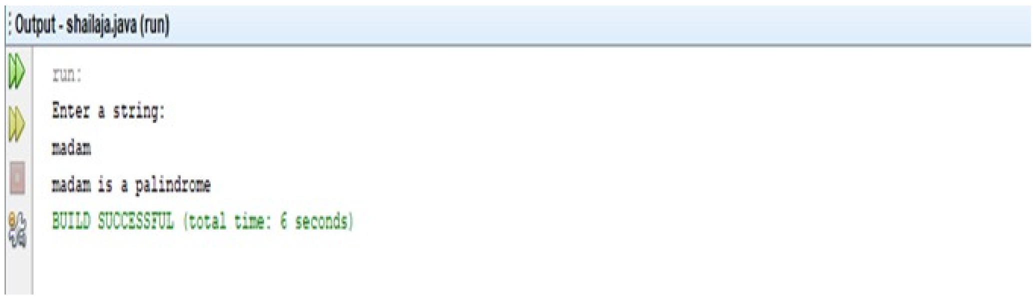
```
STEP 1: START
STEP 2: Read the given string from console using scanner class and store in the variable str
STEP 3: Find length of str.
STEP 4: Reverse the given string and store in the variable 'rev'
STEP 5: If ( str = rev) then it results as 'palindrome', otherwise it results as ' not palindrome '
```

STEP 6**: END**

SOURCE CODE:

```
import java.util.Scanner;
   class ChkPalindrome
   {
     public static void main(String args[])
     {
        String str, rev = "";
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        str = sc.nextLine();

        for ( int i = str.length();- 1; i >= 0; i-- )
           rev = rev + str.charAt(i);

        if (str.equals(rev))
           System.out.println(str+" is a palindrome");
        else
           System.out.println(str+" is not a palindrome");

     }
   }
```

OUTPUT



```
run:
Enter a string:
madam
madam is a palindrome
BUILD SUCCESSFUL (total time: 6 seconds)
```

(B)AIM: To Write a Java program for sorting list of names. Read input from command line.

THEORY: A sorting algorithm is an algorithm that puts elements of a list in a certain order. The most-used orders are numerical order and lexicographical order.

ALGORITHM : Sorting the given Names in Ascending Order

STEP 1**: START**

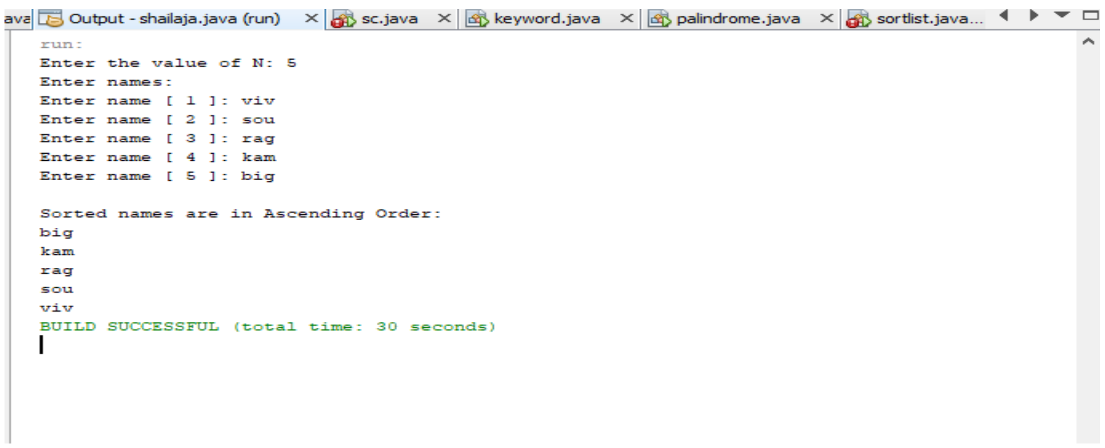
```
STEP 2: Enter the value of n.
STEP 3: Initialize temp to 0
STEP 4: Read the data from user into array A[] until n.
STEP 5: //sort the names in ascending order
STEP 6: for k : 0 to n-1 step 1
STEP 7: begin
STEP 8: for i: 0 to n-k-1 step1
STEP 9: begin
STEP 10: if(A[ i ] > A[ i+1] )
STEP 11: begin
STEP 12: // here swapping of positions is being done.
STEP 13: temp = A[ i ];
STEP 14: A[ i ] = A[ i+1 ];
STEP 15: A[ i + 1] = temp ;
STEP 16: End   //if end
STEP 17: End    //inner for loop
STEP 18: End    //outer for loop
STEP 19: //Print sorted named in ascending order
STEP 20: for k: 0 to n-1 step 1
STEP 21: print A[k]
STEP 22: END
```

SOURCE CODE:

```java
import java.util.Scanner;

class SortStrings
{
   public static void main(String args[])
   {
      String temp;
      Scanner SC = new Scanner(System.in);

      System.out.print("Enter the value of N: ");
      int N= SC.nextInt();
      SC.nextLine();

      String names[] = new String[N];

      System.out.println("Enter names: ");
      for(int i=0; i<N; i++)
      {
         System.out.print("Enter name [ " + (i+1) +" ]: ");
         names[i] = SC.nextLine();
      }
```

```
        //sorting strings
        for(int i=0; i<5; i++)
        {
           for(int j=1; j<5; j++)
           {
              if(names[j-1].compareTo(names[j])>0)
              {
                 temp=names[j-1];
                 names[j-1]=names[j];
                 names[j]=temp;
              }
           }
        }

        System.out.println("\nSorted names are in Ascending Order: ");
        for(int i=0;i<N;i++)
        {
           System.out.println(names[i]);
        }
     }
  }
```

OUTPUT



```
run:
Enter the value of N: 5
Enter names:
Enter name [ 1 ]: viv
Enter name [ 2 ]: sou
Enter name [ 3 ]: rag
Enter name [ 4 ]: kam
Enter name [ 5 ]: big

Sorted names are in Ascending Order:
big
kam
rag
sou
viv
BUILD SUCCESSFUL (total time: 30 seconds)
```

(C)AIM: To write a Java program to make frequency count of words in a given text.

THEORY: Method substring() returns a new string that is a substring of given string. Java String substring() method is used to get the substring of a given string based on the passed indexes. There are two variants of this method. Find the frequency count of words in a given text.
The substring method has two forms:

- substring(int beginIndex)
- substring(int beginIndex, int endIndex)

ALGORITHM:

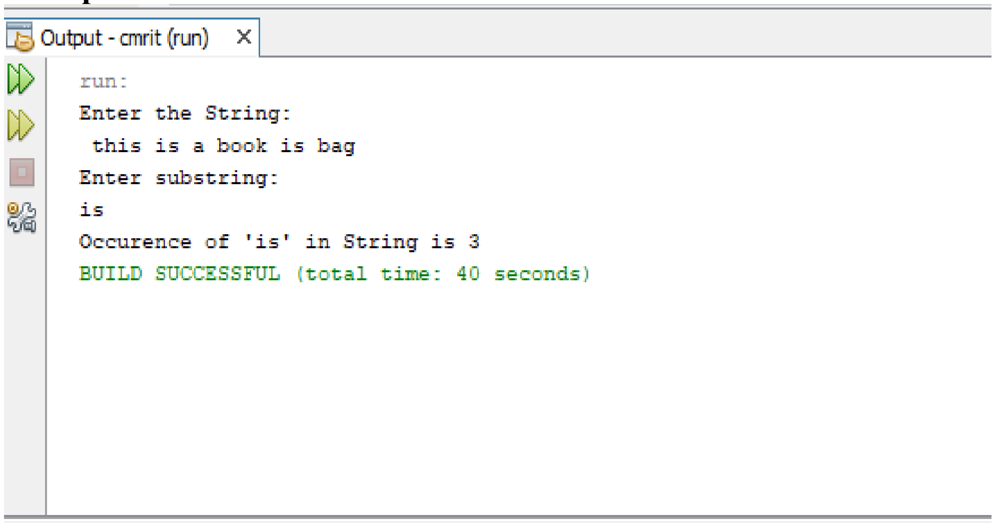
```
STEP 1: START
STEP 2: Read the sting from the console and store in the variable str using scanner class
STEP 3: Read the substring from the console and store in the variable sub using scanner class object
STEP 4: For i : 0 to str.length() do
STEP 5: Search the given substring ‘sub’ in the given string ‘str’ and compute the count of substring if it is found.
STEP 6:  end for loop
STEP 7: END
```

SOURCE CODE

```
package cmrit;
import java.util.Scanner;
import java.lang.String;
public class frequency_count {
   public static void main(String args[])
   {
      Scanner SC = new Scanner(System.in);
      System.out.println("Enter the String: ");
      String str=SC.nextLine();
      System.out.println("Enter substring: ");
      String sub=SC.nextLine();
      int index,count=0;
      for(int i=0; i+sub.length()<=str.length(); i++)
      //i+sub.length() is used to reduce comparisions
      {
//System.out.println("i="+i);
//System.out.println("sub.length= "+sub.length()+ " " +
 //      "i+sub.length()= "+ i+sub.length()+""
 //      + "string.length ="+str.length());
      index=str.indexOf(sub,i);
   //System.out.println("index position of substring= "+index);
         if(index>=0)
         {
            count++;
            i=index;
            index=-1;
//System.out.println("count= "+count+"  i=  "+ ""+i+"  ind=  "+ind);
         }
```

```
        }
    System.out.println("Occurence of '"+sub+"' in String is "+
        count);

    }
}
```

Output



```
Output - cmrit (run)
run:
Enter the String:
 this is a book is bag
Enter substring:
is
Occurence of 'is' in String is 3
BUILD SUCCESSFUL (total time: 40 seconds)
```

VIVA VOCE:

1. What is a substring?

Method **substring**() returns a new string that is a**substring** of given string. **Java** String **substring**() method is used to get the **substring** of a given string based on the passed indexes. There are two variants of this method.

2. How many forms of substring in Java?

There are **two** variants of **substring**() method.This article depicts about **all** of them, as follows : 1. **String substring**() : This method has **two** variants and returns a new **string** that is a **substring** of this **string**. The **substring** begins with the character at the specified index and extends to the end of this **string**.

3. What is String copyValueOf() Method.

The method copyValueOf() is used for copying an array of characters to the String. The point to note here is that this method does not append the content in String, instead it replaces the existing string value with the sequence of characters of array.

4. What are the outputs of the following statements?

A) a) String bc = "abc".substring(1); **output:**bc
b) String c = "abc".substring(2,3); **output:**c
c) String d = "cde".substring(1, 2); **output:**d

5. What is the procedure to reverse the given string.

A) StringBuffer d = new StringBuffer("welcome ");
d.reverse();

6. Name the package which support the substrings.

The package which support the substrings is java.lang

Week-4

(A)AIM: To write a java program to demonstrate static member, static method, static class, and static block.

Theory: Static keyword can be used with class, variable, method and block. Static members can access without object.

Static Member: A static variable is common to all the instances (or objects) of the class because it is a class level variable. only a single copy of static variable is created and shared among all the instances of the class. Memory allocation for such variables only happens once when the class is loaded in the memory.

Few Important Points:

- Static variables are also known as Class Variables.
- Unlike non-static variables, such variables can be accessed directly in static and non-static methods.

Static Method: static methods can access class variables(static variables) without using object(instance) of the class, however non-static methods and non-static variables can only be accessed using objects.
Static methods can be accessed directly in static and non-static methods.

Syntax: Static keyword followed by return type, followed by method name.

static return_type method_name();

Static Class:

- A class can be made static only if it is a nested class.
- Nested static class doesn't need reference of Outer class
- A static class cannot access non-static members of the Outer class

Static Block:

Static block is used for initializing the static variables. This block gets executed when the class is loaded in the memory. A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

ALGORITHM:

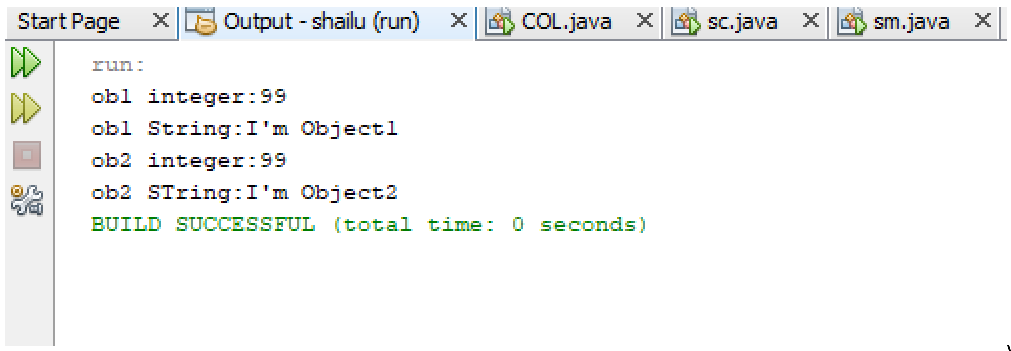
```
STEP1: START
STEP2: Create A class with name static_member_Demo
STEP3: Create a static int variable var1 and string variable var2
STEP4:  Define main () under this class
STEP5:  Create two objects namely obj1, obj2 to this class
STEP6: Assign the value to these static valuable and string variable 2 times.
STEP7:  Print the corresponding variable var1, var2 values
STEP8: STOP
```

SOURCE CODE

```
class static_member_Demo{
  //Static integer variable
  static int var1=77;
  //non-static string variable
  String var2;

  public static void main(String args[])
  {
  static_member_Demo ob1 = new static_member_Demo ();
  static_member_Demo ob2 = new static_member_Demo ();
  /* static variables can be accessed directly without
   * any instances. Just to demonstrate that static variables
   * are shared, I am accessing them using objects so that
   * we can check that the changes made to static variables
   * by one object, reflects when we access them using other
   * objects
   */
      //Assigning the value to static variable using object ob1
  ob1.var1=88;
  ob1.var2="I'm Object1";
```

```
        /* This will overwrite the value of var1 because var1 has a single
         * copy shared among both the objects.
         */
        ob2.var1=99;
   ob2.var2="I'm Object2";
   System.out.println("ob1 integer:"+ob1.var1);
   System.out.println("ob1 String:"+ob1.var2);
   System.out.println("ob2 integer:"+ob2.var1);
   System.out.println("ob2 STring:"+ob2.var2);
   }
}
```

OUTPUT:



```
run:
ob1 integer:99
ob1 String:I'm Object1
ob2 integer:99
ob2 STring:I'm Object2
BUILD SUCCESSFUL (total time: 0 seconds)
```

\

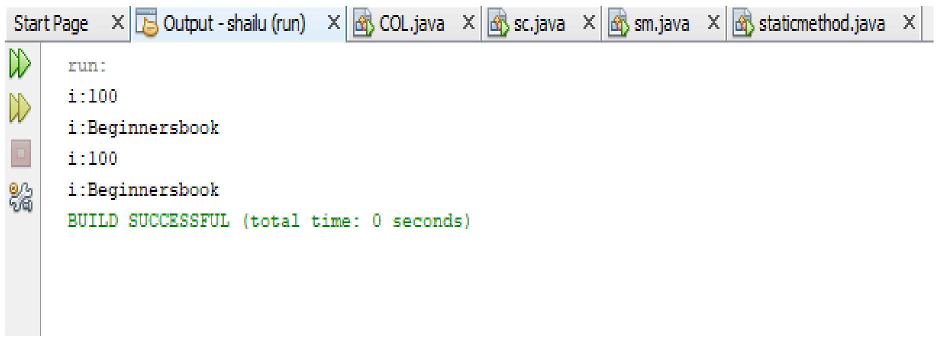
ALGORITHM: Static Method

```
STEP1: START
STEP2: Create A class with name static_Method_Demo
STEP3: Create a static int variable i and static string variable s
STEP4:  Define static method display () to print the static variables i ,s values
STEP5:  Define non-static method func () to call the static method display ()
STEP6: Define main () under this class
STEP7:  Create two objects namely obj1 to this class
STEP8:  call the non-static method func () using object obj1
STEP6:  call static method display () without any object
STEP7:  STOP
```

SOURCE CODE

```
class static_method_Demo{
  static int i = 100;
  static String s = "Beginnersbook";
  //Static method
  static void display()
  {
     System.out.println("i:"+i);
     System.out.println("i:"+s);
  }

  //non-static method
  void funcn()
  {
     //Static method called in non-static method
     display();
  }
  //static method
  public static void main(String args[])
  {
    static_method_Demo obj = new static_method_Demo ();
    //You need to have object to call this non-static method
    obj.funcn();

     //Static method called in another static method
     display();
   }
}
```

OUTPUT



```
run:
i:100
i:Beginnersbook
i:100
i:Beginnersbook
BUILD SUCCESSFUL (total time: 0 seconds)
```

ALGORITHM : Static Class

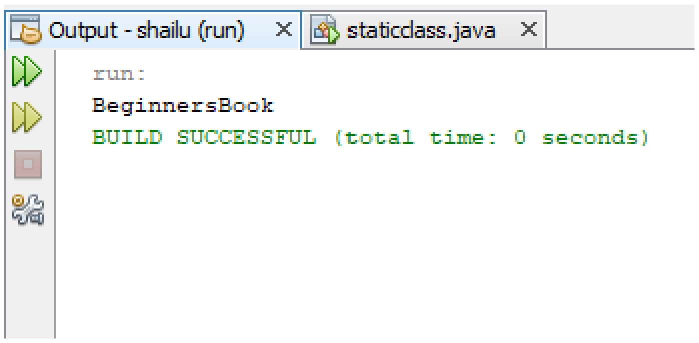
STEP1: **START**
STEP2: Create a outer class with name static_outer_class_Demo
STEP3: Create a static string variable str
STEP4: Create a static nested class with name 'MyNestedClass' under outer class
STEP5: Define non static method display () under this nested class to print static variable 'str' value
STEP6: Define main () under outer class
STEP7: for a regular nested class need to create an instance of outer class first with name

'obj1'
STEP8: call the non-static method display () using object obj1
STEP9: **STOP**

SOURCE CODE

```
class static_outer_class_Demo{
  private static String str = "BeginnersBook";

  //Static class
  static class MyNestedClass{
  //non-static method
  public void display() {

    /* If you make the str variable of outer class
     * non-static then you will get compilation error
     * because: a nested static class cannot access non-
     * static members of the outer class.
     */
    System.out.println(str);
  }

  }
  public static void main(String args[])
  {
     /* To create instance of nested class we didn't need the outer
  * class instance but for a regular nested class you would need
  * to create an instance of outer class first
      */
static_outer_class_Demo.MyNestedClass obj = new
                         static_outer_class_Demo.MyNestedClass();
  obj.display();
  }
}
```

OUTPUT:



```
run:
BeginnersBook
BUILD SUCCESSFUL (total time: 0 seconds)
```

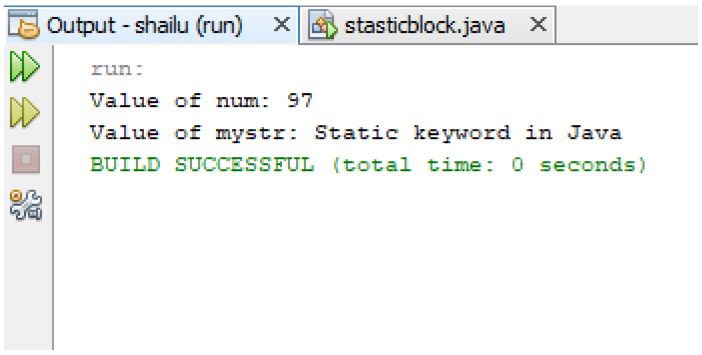
ALORITHM: Static Block

```
STEP1: START
STEP2: Create a outer class with name static_block_Demo
STEP3: define static int variable num, static string instance str
STEP4: define static block  to print the message (‘static block1’) and initialize static variables as
           num = 68 and str =”Block1”.
STEP5:  define static block  to print the message (‘static block2’) and initialize static variables as
           num = 98 and str =”Block2”.
STEP6: Define main () under outer class
STEP7:  print the static variables num and str  without object
STEP8:  STOP
```

SOURCE CODE

```
class JavaExample{
  static int num;
  static String mystr;
  static{
    num = 97;
    mystr = "Static keyword in Java";
  }
  public static void main(String args[])
  {
    System.out.println("Value of num: "+num);
    System.out.println("Value of mystr: "+mystr);
  }
}
```

OUTPUT:



```
run:
Value of num: 97
Value of mystr: Static keyword in Java
BUILD SUCCESSFUL (total time: 0 seconds)
```

(B)AIM: To write a java program to demonstrate method overloading and method Overriding

THEORY: Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different

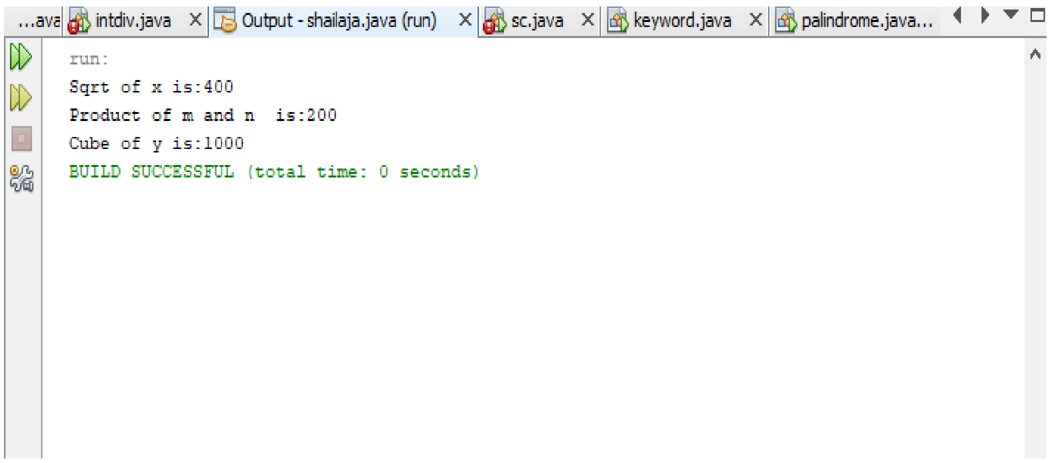
ALGORITHM:

```
STEP 1: START
STEP 2: Create a class with name ‘Overloading_Demo’
STEP 3: Define three user defined ‘display ()’ methods with different parameter lists
STEP 4: Define main ()
STEP 5: Create the object ‘obj’ to the class ‘Overloading_Demo’
STEP 6:  Using object obj call the display () with corresponding parameter values
STEP 7:  STOP.
```

SOURCE CODE

```
class MethodOverLoad
  {
 void calValue()
 {
 int x=20;
 x=x*x;
 System.out.println("Sqrt of x is:"+x);
 }
 void calValue(int y)
 {
 y=y*y*y;
 System.out.println("Cube of y is:"+y);
 }
 void calValue(int m,int n)
 {
  int z=m*n;
 System.out.println("Product of m and n  is:"+z);
 } }
 class MOL
 {
 public static void main(String args[])
 {
 MethodOverLoad m=new MethodOverLoad();
 m.calValue();
 m.calValue(10,20);
 m.calValue(10);
 }
 }
```

OUTPUT:


```
run:
Sqrt of x is:400
Product of m and n  is:200
Cube of y is:1000
BUILD SUCCESSFUL (total time: 0 seconds)
```

THEORY: Declaring a method in **sub class** which is already present in **parent class** is known as **method overriding**. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. In this case the method in parent class is called overridden method and the method in child class is called overriding method.

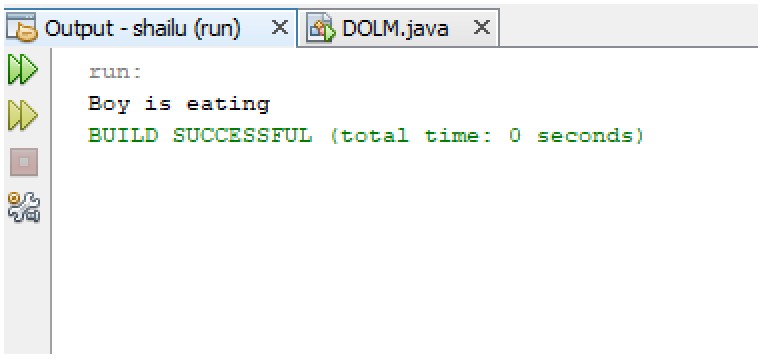
ALGORITHM:

```
STEP1: START
STEP2: Create a parent class with name 'Human'
STEP3: Define a user defined method eat() to print the message "Human is eating" under this parent
          or super class
STEP4: Create a child class with name 'Boy' inherited from the parent class 'Human'
STEP5:  Define  the same user defined method eat() to print the message "Boy is eating" under this
           Child  or Derived  class
STEP6: Define main () under this child class 'Boy'
STEP7:  Create an object  'obj1' to this child class 'Boy'
STEP8:  call user defined eat() method  using the child class object obj1  to print the message
          'Boy is eating'        //which overrides the parent class
STEP9:  STOP
```

SOURCE CODE

```
class Human{
  //Overridden method
  public void eat()
  {
    System.out.println("Human is eating");
  }
}
class Boy extends Human{
  //Overriding method
  public void eat(){
    System.out.println("Boy is eating");
  }
  public static void main( String args[]) {
    Boy obj = new Boy();
    //This will call the child class version of eat()
    obj.eat();
  }
}
```

OUTPUT



```
run:
Boy is eating
BUILD SUCCESSFUL (total time: 0 seconds)
```

(C)AIM: To write a java program to demonstrate finals, blank finals, final methods, and final classes

THEORY: The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

- variable
- method
- class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only.

1) Java final variable: If we make any variable as final, you cannot change the value of final variable (It will be constant).

ALGORITHM: Java final variable

STEP1: **START**
STEP2: create a class with name ‘Bike’ .
STEP3: declare a variable final int speedlimit and initialize with value 90.
STEP4: Define a user defined method run() to initialize the variable ‘speedlimit’ with 400
STEP5: Define main() under this class ‘Bike’
STEP6: create an object ‘obj’ to this class ‘Bike’
STEP7: call the run() using obj to display the value of speed limit
STEP8: **STOP**

SOURCE CODE

```
class Bike9{
     final int speedlimit=90;//final variable
     //void run(){
      //int speedlimit=400;
        //System.out.println(speedlimit);
     //}
      }
      class finalvar1{

     public static void main(String args[]){
     Bike9 obj=new  Bike9();
     obj.speedlimit=100;
       //cannot be accessed
       //out side as it is final variable
       //obj.run();
       }
      }
```

OUTPUT: Compile Time Error.

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

ALGORITHM

STEP1: START
STEP2: Create a super class with name ‘Bike’ .
STEP3: Define a final method run() to print the message ‘running’
STEP4: Create a derived class with name ‘Honda’ from super class ‘Bike’
STEP5: Define run() to print the message ‘running safely with 100kmph’

STEP6: Define main() under this class
STEP7: Create an object 'honda' to the derived class 'Honda'
STEP8: Call run() using an derived class object 'honda' to print the message
STEP9: STOP

SOURCE CODE

```
class Bike
{
   final void run()
   {
      System.out.println("running");
   }
}
class Honda extends Bike
{
   void run()
   // method cannot be overriden
   //as it is final method
   {
      System.out.println("running safely with 100kmph");
   }


public static void main(String args[])
        {
      Honda honda= new Honda();
      honda.run();
      }
    }
```

OUTPUT: Compile Time Error.

3) Java final class: If you make any class as final, you cannot extend it.

ALGORITHM:

STEP1: START

STEP2: Create a final class with name ‘Bike’ .
STEP3: Define user defined run() to print the message ‘Final class’
STEP4: Create a derived class with name ‘Honda’ from super class ‘Bike’
STEP5: Define run() to print the message ‘Derived class’
STEP6: Define main() under this class
STEP7: Create an object ‘honda’ to the derived class ‘Honda’
STEP8: Call run() using an derived class object ‘honda’ to print the message
STEP9: STOP

SOURCE CODE

```
   final class Bike1{
      void run()
        {
           System.out.println("final class");
        }
        }
      class Honda1 extends Bike1
        // class cannot be inherited
         //as it is final class
        {
        void run()
          {
             System.out.println("derived class");
          }
        }
```

OUTPUT: Compile Time Error.

(D)AIM: To write a java program to demonstrate synchronized keyword.

Theory: If a method is declared as synchronized, it is known as synchronized method. Synchronized method is used to lock an object for any shared resource. When a thread invokes a synchronized method, it automatically acquires the lock for that object and releases it when the thread completes its task

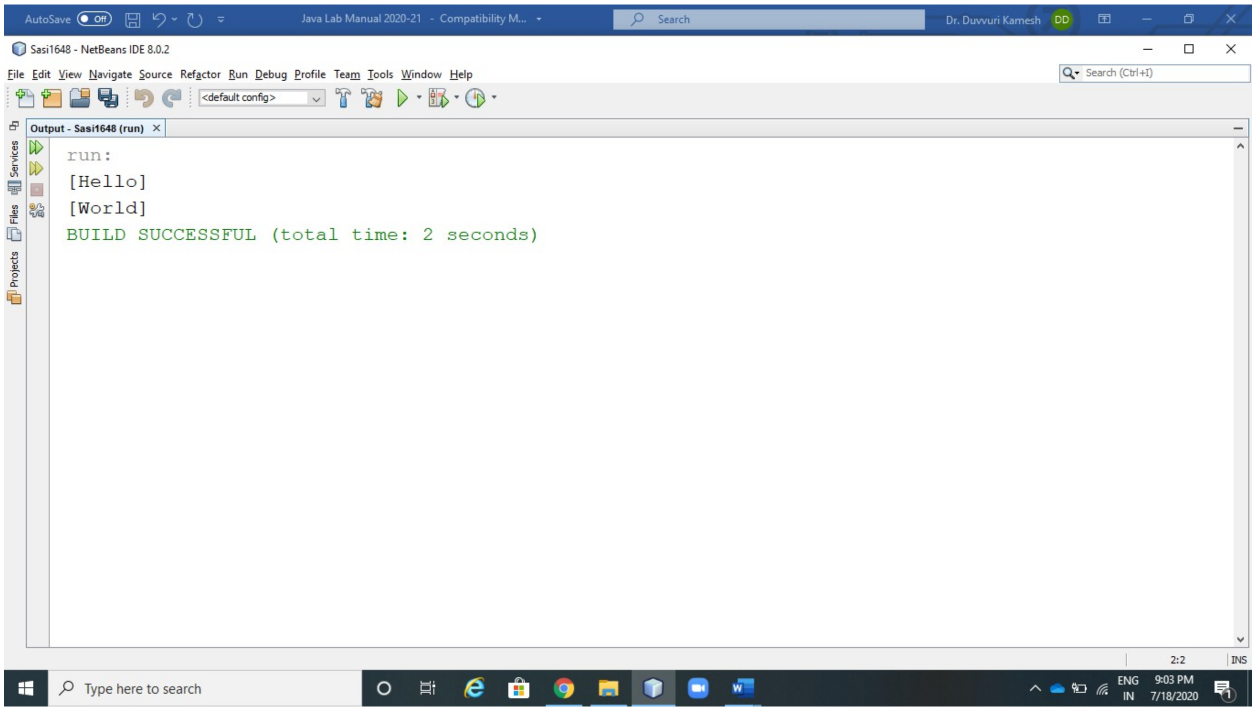
ALGORITHM:

STEP1: Create a thread class by implementing the Runnable interface.
STEP 2: Start the thread in the constructor.
STEP 3: Implement the run() method. Inside this method create a synchronize()
STEP 4: Use sleep() method to pause the thread for one second.
STEP 5: Create multiple threads in the main() method.
STEP 7. Invoke the threads.

SOURCE CODE

```
/ This program is not synchronized.
class Callme
{
   void call(String msg)
   {
      System.out.print("[" + msg);
      try
      {
         Thread.sleep(1000);
      }
      catch(InterruptedException e)
      {
         System.out.println("Interrupted");
      }
      System.out.println("]");
   }
}
class Caller implements Runnable
{
   String msg;
   Callme target;
   Thread t;
   public Caller(Callme targ, String s)
   {
      target = targ;
      msg = s;
      t = new Thread(this,"demo");
      t.start();
   }
public void run()
{
   synchronized(target){
         target.call(msg);
   }
}
}
class Synch
{
   public static void main(String args[])
   {
      Callme target = new Callme();
      Caller ob1 = new Caller(target, "Hello");
      Caller ob3 = new Caller(target, "World");
}
}
```

OUTPUT



```
run:
[Hello]
[World]
BUILD SUCCESSFUL (total time: 2 seconds)
```

VIVA VOCE:

1. Difference between Static variable and non-Static variable.

   1. A static variable is shared among all instances of a class.

   2. A non-static variable is specific to a single instance of that class.

2. Difference between method overloading and method overriding

method overloading	method overriding
Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its

	super class.

3. Define java final keyword? write the syntax to define final variable

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

Syntax:

```
          public class Test {
        public static void main(String args[]) {
                final int i = 10;
                  i = 30; // Error because i is final.
                }
                        }
```

4. List the forms of java final keyword?

*variable

*method

*class

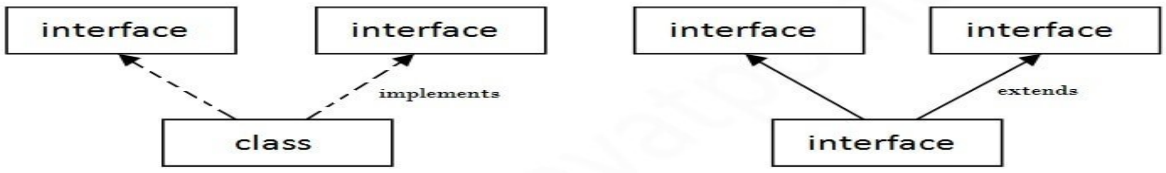
5. Give an example of synchronization method.

synchronized void printTable(**int** n)

WEEK 5

(A)AIM: To write a java program to implement multiple inheritance using interface.

THEORY: If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



Multiple Inheritance in Java

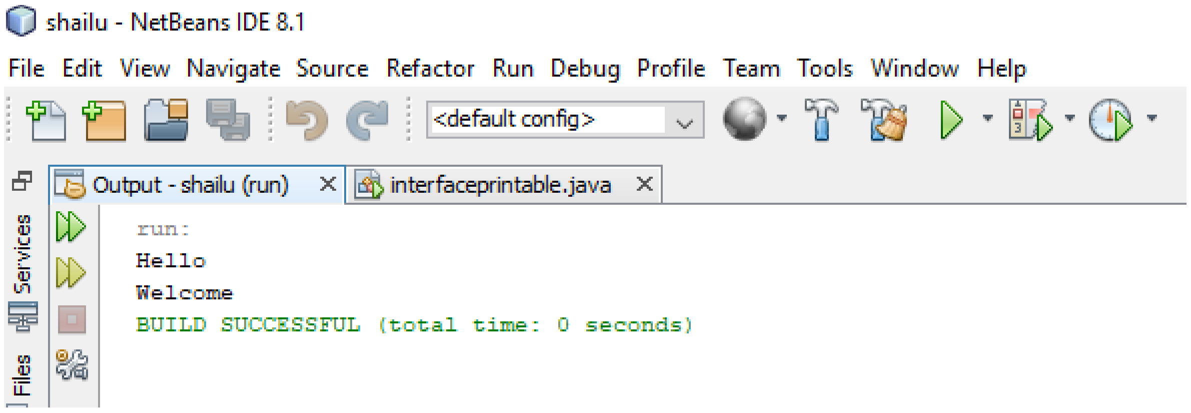
ALGORITHM:

STEP1: START
STEP2: Define an interface with name ‘ Printable’
STEP3: Define print() method without body .
STEP4: Define an interface with name ‘ Showable’
STEP5: Define show() method without body .
STEP6: Create a class with name multiple_inheritance implements interfaces Printable,Showable
STEP7: Define print() to display a message ‘Hello’
STEP8: Define show() to display a message ‘Welcome’
STEP9: Define main() under this class
STEP10: Create an object ‘obj’
STEP11: call print() and show() using the object ‘obj’ to print the corresponding Messages.
STEP12: STOP

SOURCE CODE

```
interface Printable{
    void print();
    }
    interface Showable{
    void show();
    }
    class multiple_inheritance implements Printable, Showable{
    public void print(){System.out.println("Hello");}
    public void show(){System.out.println("Welcome");}

    public static void main(String args[]){
    multiple_inheritance obj = new multiple_inheritance ();
    obj.print();
    obj.show();      }
}
```

OUTPUT


shailu - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - shailu (run)
interfaceprintable.java
Services
Files

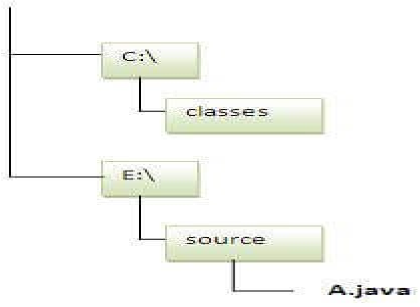
```
run:
Hello
Welcome
BUILD SUCCESSFUL (total time: 0 seconds)
```

(B)AIM: To write a program to demonstrate packages.

THEORY: A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and 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
1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
2) Java package provides access protection.
3) Java package removes naming collision.

Send the class file to another directory or drive. There is a scenario, I want to put the class file of A.java source file in classes folder of c: drive.



ALGORITHM:

STEP1: START
STEP2: Create a package with name 'mypack' in the current directory
STEP3: Write a simple java program to print the message "welcome to package" ,import the package 'mypack' and save the file with name Simple.java
STEP4: set the class path to send the class file of A.java source file into classes folder of c: drive.
STEP5: STOP

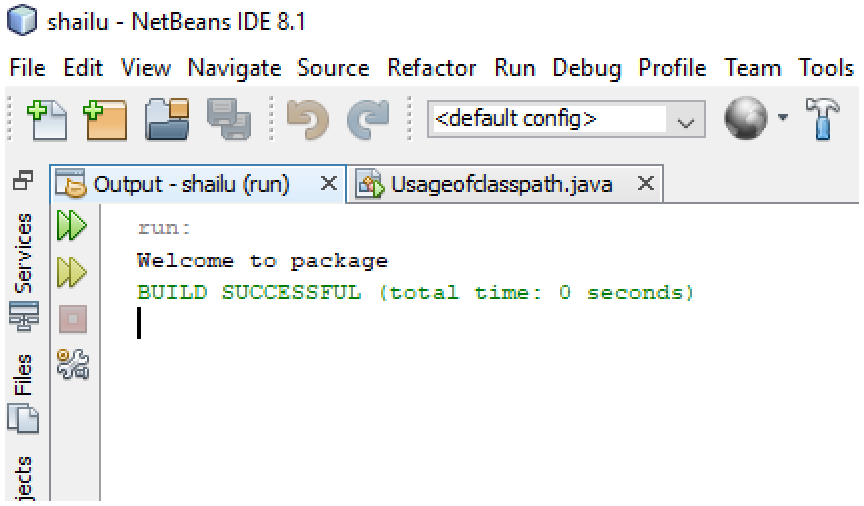
To Compile:
e:\sources> javac -d c:\classes Simple.java

To Run:
To run this program from e:\source directory, you need to set classpath of the directory where the class file resides.
e:\sources> set classpath=c:\classes;.;
e:\sources> java mypack.Simple

SOURCE CODE

```
package shailu;
    public class Usageofclasspath{
     public static void main(String args[]){
        System.out.println("Welcome to package");
       }
    }
```

OUTPUT:


shailu - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools
<default config>
Output - shailu (run)
Usageofclasspath.java
Services
Files
jects

```
run:
Welcome to package
BUILD SUCCESSFUL (total time: 0 seconds)
```

(C)AIM: To write a Java program to create an abstract class

THEORY: We can require that some methods be overridden by sub classes by specifying the abstract type modifier. These methods are sometimes referred to as sub classer responsibility as they have no implementation specified in the super class. Thus a sub class must override them. Any class that contains one or more abstract methods must also be declared abstract. Such types of classes are known as abstract classes. Abstract classes can contain both abstract and non-abstract methods.

ALGORITHM:

```
STEP1: START
STEP2: Create an abstract class named Shape.
STEP3: Declare integer variables for height, width and radius in the abstract class Shape.
STEP4: Declare abstract method printArea() in the abstract class Shape.
STEP5: Create sub classes Rectangle, Triangle and Circle that extends Shape.
STEP6: Implement the printArea() method in all three classes.
STEP7: Invoke the methods in the main class by the respective objects.
STEP8: END
```

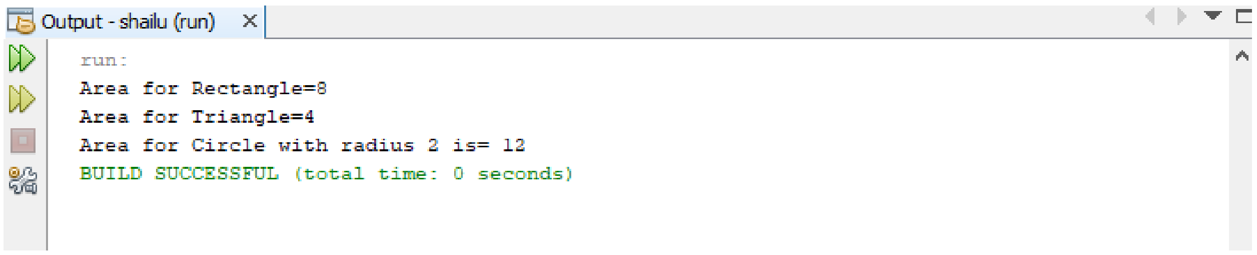
SOURCE CODE:

```
abstract class Shape
{
        int a=2;
        int b=4;
        abstract void printArea();
}
class Rectangle extends Shape
{

        void printArea()
        {

                int r=a*b;
                System.out.println("Area for Rectangle="+r);
```

```
            }
    }
    class Triangle extends Shape
    {

            void printArea()
            {
           int t=(a*b)/2;
               System.out.println("Area for Triangle="+t);

            }
    }
    class Circle extends Shape
    {

            void printArea()
            {
            int area=(int)(3.14*a*a);
      System.out.println("Area for Circle with radius "+a+" is= "+area);

            }
       }
    public class AbstractDemo
    {
            public static void main(String args[])
            {
               Rectangle r= new Rectangle();
               Triangle t= new Triangle();
               Circle c=new Circle();
               Shape s;
```

```
        s=r;
        s.printArea();
        s=t;
        s.printArea();
        s=c;
        s.printArea();
        }
}
```

OUTPUT:



Output - shailu (run)

```
run:
Area for Rectangle=8
Area for Triangle=4
Area for Circle with radius 2 is= 12
BUILD SUCCESSFUL (total time: 0 seconds)
```

(D)AIM: To write a Java program that creates a user interface to perform integer divisions.

THEORY: Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a Web document. After an applet arrives on the client, it has limited access to resources, so that it can produce an arbitrary multimedia user interface and run complex computations without introducing the risk of viruses or breaching data integrity. The Applet class is contained in the java.applet package. All applets are subclasses of Applet. Thus, all applets must import java.applet

ALGORITHM:

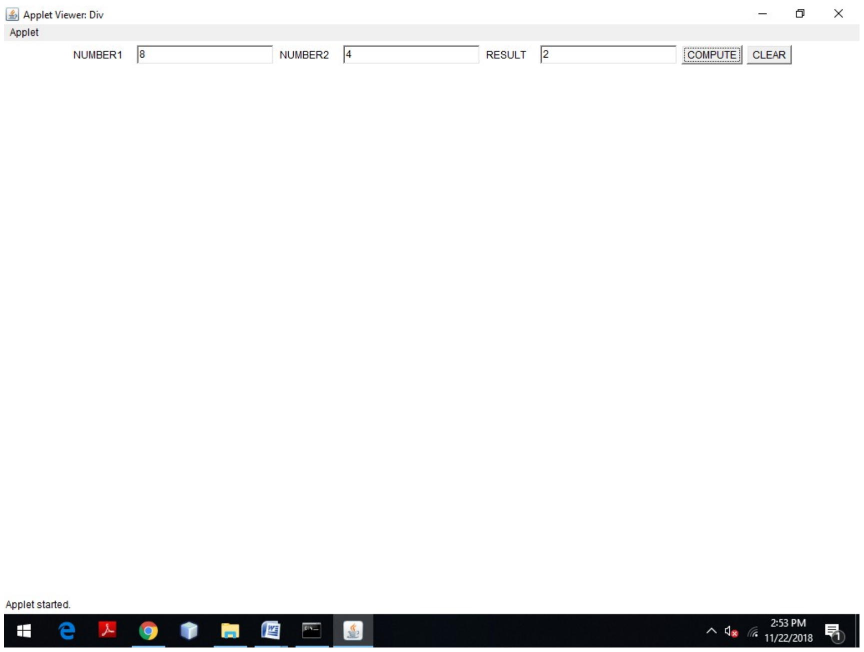
STEP1: START
STEP2: Create an applet using extends Applet class.
STEP3: Create three text fields to accept Num1, Num2 and Result field.
STEP4: Create two buttons named “Divide” to generate result and “Clear” to clear the three text fields.
STEP5: Use JoptionPane class to create message dialog box.
STEP6: Use actionPerformed() method of ActionListener interface to handle button events.
START7: END

SOURCE CODE:

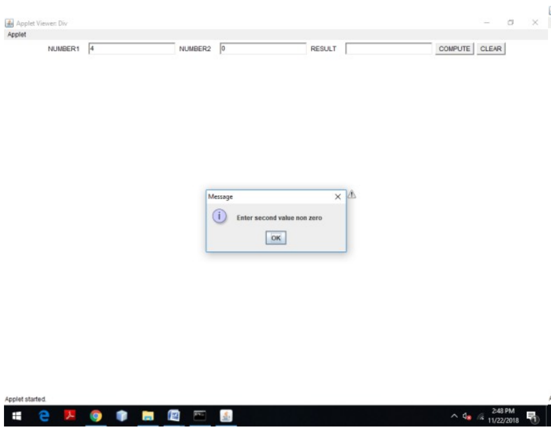
```
import java.applet.*;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;


/*<applet code=Div width=500 height=500>

</applet>*/

public class Div extends Applet implements ActionListener

{

        Button b1,b2;

        Label l1,l2,l3;

        TextField tf1,tf2,tf3;

        String msg;

        public void init()

        {

                b1=new Button("COMPUTE");

                b1.addActionListener(this);

                b2=new Button("CLEAR");

                b2.addActionListener(this);

                tf1=new TextField(20);

                tf2=new TextField(20);

                 tf3=new TextField(20);
```

```
            l1=new Label("NUMBER1");
            l2=new Label("NUMBER2");
            l3=new Label("RESULT");
            add(l1);
            add(tf1);
            add(l2);
            add(tf2);
            add(l3);
            add(tf3);
            add(b1);
            add(b2);
}
public void actionPerformed(ActionEvent ae)
{
 if(ae.getSource()==b1)
 {

try
{
int a=Integer.parseInt(tf1.getText());
int b=Integer.parseInt(tf2.getText());
int c=a/b;
tf3.setText(""+c);
}
catch(NumberFormatException ex)
{
tf3.setText(" ");
JFrame f=new JFrame();
JOptionPane.showMessageDialog(f,"Enter only numbers");
repaint();
}
catch(ArithmeticException ex)
{
tf3.setText(" ");
JFrame f=new JFrame();
JOptionPane.showMessageDialog(f,"Enter second value non zero");
repaint();
}
}
else
{
   tf1.setText("");
   tf2.setText("");
   tf3.setText("");
   msg="";
   repaint();
}
}
public void paint(Graphics g)
{
g.drawString(msg,30,70);
}
}
```

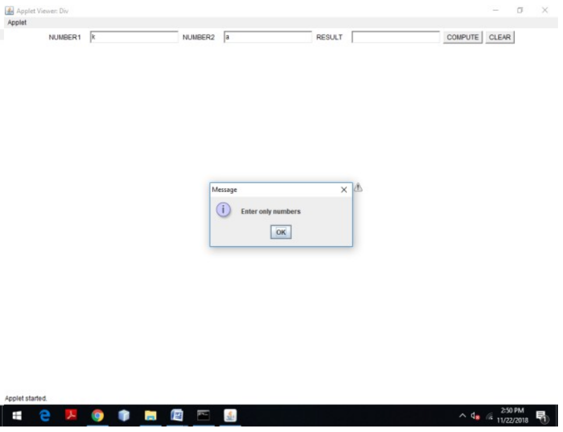
OUTPUT:



```
Applet Viewer: Div
Applet
NUMBER1 8    NUMBER2 4    RESULT 2    COMPUTE  CLEAR
Applet started.
```



```
Applet Viewer: Div
Applet
NUMBER1 4    NUMBER2 0    RESULT     COMPUTE  CLEAR
Message
Enter second value non zero
OK
Applet started.
```



```
Applet Viewer: Div
Applet
NUMBER1 k    NUMBER2 a    RESULT     COMPUTE  CLEAR
Message
Enter only numbers
OK
Applet started.
```

VIVA - VOCE:

1. Define java Inheritance.

   Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

2. Define java package

   A **java package** is a group of similar types of classes, interfaces and sub-packages.

3. How many ways access package from outside package.

There are three ways to access the package from outside the package.

- ✔ import package.*;
- ✔ import package.classname;
- ✔ fully qualified name

4. What is an abstract class?

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body). We can require that some methods be overridden by sub classes by specifying the abstract type modifier.

5. What is the purpose of init() method and start() method?

init() method is called at the time of starting the execution. This is called only once in the life cycle. start() method is called by the init() method. This method is called a number of times in the life cycle; whenever the applet is deiconifed, to make the applet active.

WEEK-6

(A)AIM: To write a java program to crate and test user defined exception class.

THEORY: Creating our own Exception known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need. By the help of custom exception, we can have your own exception and message.The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

Java try block
Java try block is used to enclose the code that might throw an exception. It must be used within the method.
Java catch block
Java catch block is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent class exception (i.e., Exception) or the generated exception type. The catch block must be used after the try block only. You can use multiple catch block with a single try block.

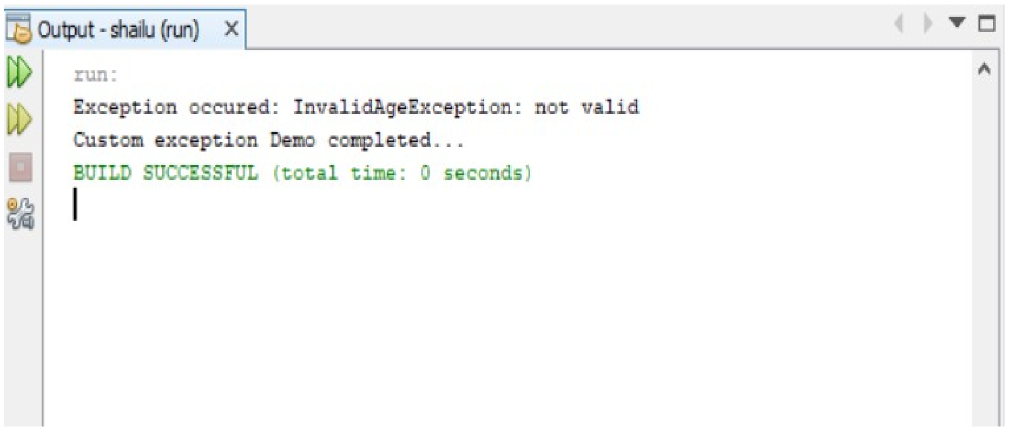
ALGORITHM :

```
STEP1: START
STEP2:  Create a class with name ‘InvalidAgeException’ extends exception
STEP3: Define a parameterized constructor InvalidAgeException(string s)
STEP4. Call parent constructor in the derived class constructor  using super(s) keyword.
STEP5: Close the custom exception class ‘InvalidAgeException’
STEP6: Create a class with name ‘TestCustomException1’
STEP7: Define static method validate( int age )  throws custom exception ‘InvalidAgeException’
STEP8: Validate age: if age<18 then throw new InvalidAgeException("not valid")
STEP9:  else print ("welcome to vote")
STEP10: Define main()
STEP11:  call validate() static method with one parameter value in the try and catch block . Trt and catch block will  throws an exception if any
STEP12: print message ‘Custom exception Demo completed..’
STEP13: STOP
```

SOURCE CODE:

```
class InvalidAgeException extends Exception{
         InvalidAgeException(String s){
          super(s);
         }
        }
        class TestCustomException1{


          static void validate(int age)throws InvalidAgeException{
            if(age<18)
             throw new InvalidAgeException("not valid");
            else
             System.out.println("welcome to vote");
          }

          public static void main(String args[]){
             try{
             validate(13);
             }catch(Exception m){System.out.println("Exception occured: "+m);}

             System.out.println("Custom exception Demo completed...");
          }
        }
```

OUTPUT



Output - shailu (run)

```
run:
Exception occured: InvalidAgeException: not valid
Custom exception Demo completed...
BUILD SUCCESSFUL (total time: 0 seconds)
```

(B)AIM: To write a Java program that implements a multi-thread application that has three threads.

THEORY: Multithreading replaces event loop programming by dividing your tasks into discrete and logical units. Threads also provide a secondary benefit: they do away with polling. Pollin is usually implemented by a loop that is used to check some condition repeatedly. To avoid polling, Java includes an elegant inter-process communication mechanism via the **wait( )**, **notify( )**, and **notifyAll( )** methods. These methods are implemented as **final** methods in **Object**, so all classes have them. All three methods can be called only from within a **synchronized** context.

ALGORITHM:

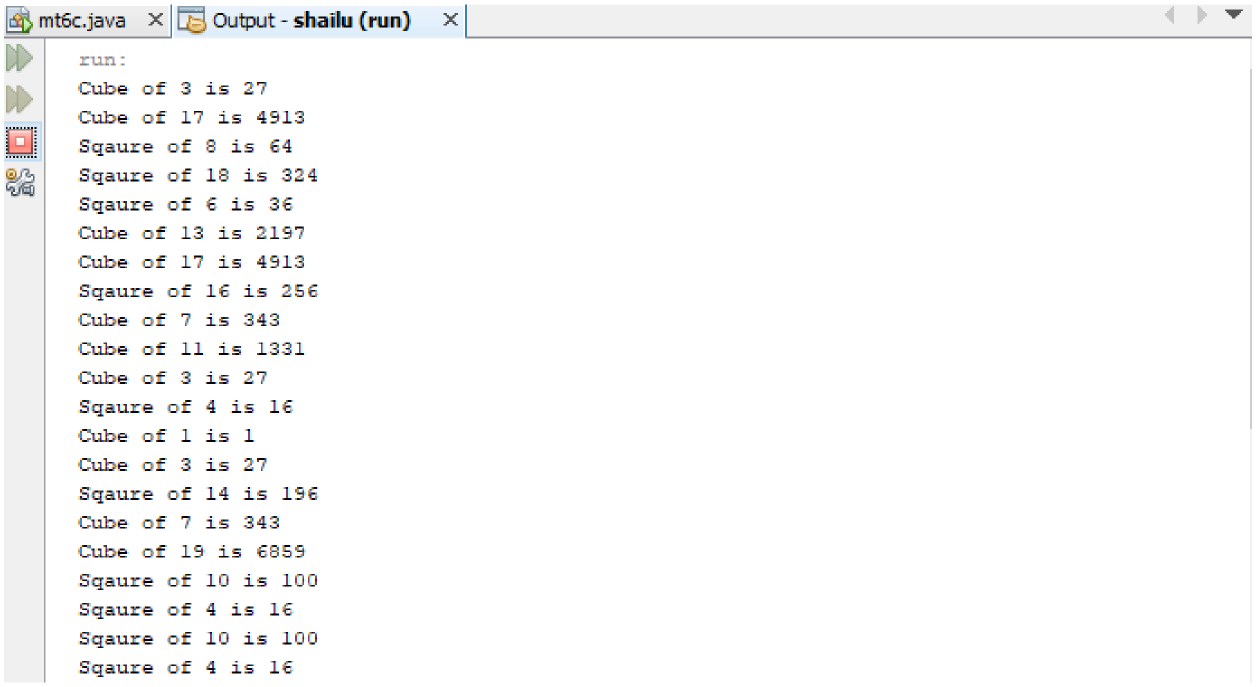
```
STEP1: START
STEP2: Create a thread class by implementing the Runnable interface.
STEP3: Start the thread in the constructor.
STEP4: Implement the run() method.
STEP5: Use sleep() method to pause the thread for one second.
STEP6: Create multiple threads in the main() method.
STEP7: Invoke the threads.
STEP8: END
```

SOURCE CODE:

```
import java.util.*;
class Odd implements Runnable
{
	int i;
	Odd(int i)
	{
		this.i = i;
	}
	public void run()
	{
		System.out.println("Cube of "+i+" is "+(i*i*i));
	}
}
class Even implements Runnable
{
	int i;
	Even(int i)
	{
		this.i = i;
	}
	public void run()
```

```
        {
                System.out.println("Sqaure of "+i+" is "+(i*i));
        }
}
class Random1 extends Thread
{
        public void run()
        {
                int j=0;
                Random  rand = new Random();
                while(true){
                        j = rand.nextInt(20);
                        if(j%2!=0)
                        {
                                Odd o = new Odd(j);
                        Thread t1 = new Thread(o);
                                t1.start();
                                try
                                {
                                Thread.sleep(2000);
                                }
                                catch(InterruptedException ie)
                                { }
                        }
                        else
                        {
                                Even e = new Even(j);
                        Thread t2 = new Thread(e);
                                t2.start();
                                try
                                {
                                        Thread.sleep(1000);
                                }
                                catch(InterruptedException ie)
                                { }
                        }
                }
        }
}
public class mt6c
{
        public static void main(String[] a)
        {
                Random1 r = new Random1();
                Thread t = new Thread(r);
                t.start();
        }
}
```

OUTPUT:



```
run:
Cube of 3 is 27
Cube of 17 is 4913
Sqaure of 8 is 64
Sqaure of 18 is 324
Sqaure of 6 is 36
Cube of 13 is 2197
Cube of 17 is 4913
Sqaure of 16 is 256
Cube of 7 is 343
Cube of 11 is 1331
Cube of 3 is 27
Sqaure of 4 is 16
Cube of 1 is 1
Cube of 3 is 27
Sqaure of 14 is 196
Cube of 7 is 343
Cube of 19 is 6859
Sqaure of 10 is 100
Sqaure of 4 is 16
Sqaure of 10 is 100
Sqaure of 4 is 16
```

VIVA - VOCE:

1. What is an Exception? Define Try and catch Block

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.

Java try block is used to enclose the code that might throw an exception. The try **block** contains set of statements where an exception can occur. A try **block** is always followed by a**catch block**, which handles the exception that occurs in associated try **block**. A try **block** must be followed by**catch blocks** or finally **block** or both.

2. What is the importance of Exception handling in java?

The core advantage of **exception handling** is to maintain the normal flow of the application. An **exception** normally disrupts the normal flow of the application that is why we use **exception handling**.

3. What is the Difference between Throw and throws ?

Throw	**Throws**
Throw keyword is used **in the** method body to **throw**an	**Throws** clause is used to declare an exception, which means it works

exception, while **throws** is used in method signature to declare the exceptions that can occur **in the** statements present **in the** method.	similar to the try-catch block.

4. What is a thread?

A thread, in the context of Java, is the path followed when executing a program. A thread is a light-weight process. All Java programs have at least one thread, known as the main thread, which is created by the Java Virtual Machine (JVM) at the program's start.

5. What is the life cycle of thread?

The following are the thread life cycle methods:

New − A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

Runnable − After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

Waiting − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

Timed Waiting − A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

Terminated (Dead) − A runnable thread enters the terminated state when it completes its task or otherwise terminates.

WEEK-7

AIM: To write a Java program to demonstrate MouseListener, MouseMotionListener and KeyListener.

THEORY: A listener is an object that is notified when an event occurs. It has two major requirements. First, it must have been registered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications. The methods that receive and process events are defined in a set of interfaces found in java.awt.event.

ALGORITHM:

```
STEP1: START
STEP2: Create an applet extending from an Applet class.
STEP3: In the init()method add both MouseListener and MouseMotionListener.
STEP4: Extend the MouseAdapter and MouseMotionAdapter classes.
STEP5: Override the following methods
        a) mouseClicked()
        b) mouseEntered()
        c) mouseExited()
        d) mousePressed()
        e) mouseReleased()
        f) mouseDragged ()
        g) mouseMoved()
STEP6: Using showStatus() method display the message.
STEP7: Display the necessary information on the screen.
STEP8: END
```

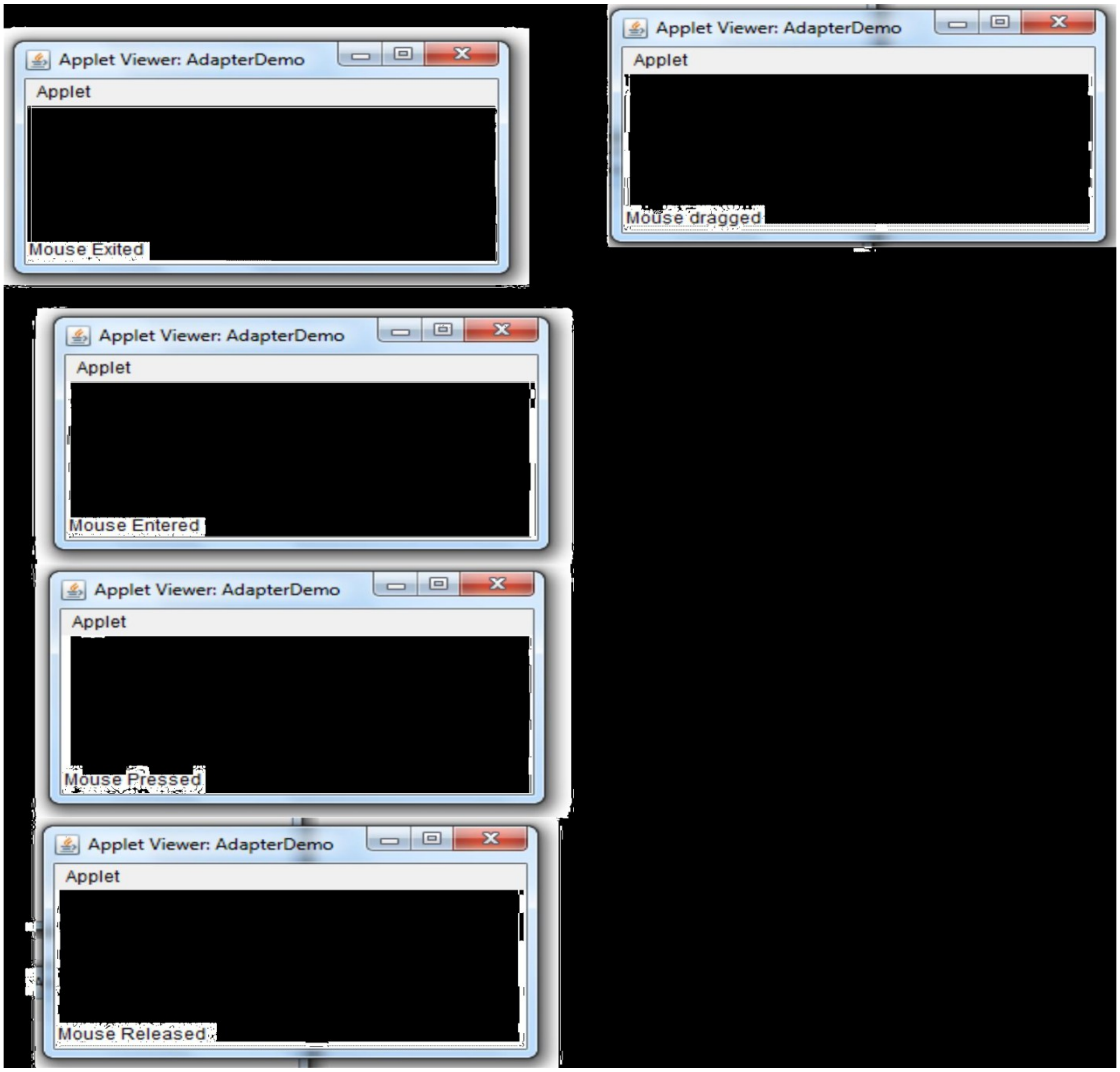
SOURCE CODE:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="MouseEvents" width=300 height=300>
</applet>*/
public class MouseEvents extends Applet implements MouseListener,
MouseMotionListener
{
        String msg = "";
        int mouseX = 0, mouseY = 0; // coordinates of mouse
        public void init()
        {
```

```
        addMouseListener(this);
        addMouseMotionListener(this);
    }
    public void mouseClicked(MouseEvent me)
    {
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse clicked.";
        repaint();
    }
    public void mouseEntered(MouseEvent me)
    {
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse entered.";
        repaint();
    }
    public void mouseExited(MouseEvent me)
    {
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse exited.";
        repaint();
    }
    public void mousePressed(MouseEvent me)
    {
        mouseX = me.getX();
        mouseY = me.getY();
        msg = "Down";
        repaint();
    }
    public void mouseReleased(MouseEvent me)
    {
```

```
                mouseX = me.getX();
                mouseY = me.getY();
                msg = "Up";
                repaint();
        }
        public void mouseDragged(MouseEvent me)
        {
                mouseX = me.getX();
                mouseY = me.getY();
                msg = "*";
                showStatus("Dragging mouse at " + mouseX + ", " + mouseY);
                repaint();
        }
        public void mouseMoved(MouseEvent me)
        {
                showStatus("Moving mouse at " + me.getX() + ", " + me.getY());
        }
        public void paint(Graphics g)
        {
                g.drawString(msg, mouseX, mouseY);
        }
}
```

OUTPUT:



```
Mouse Exited
Mouse dragged
Mouse Entered
Mouse Pressed
Mouse Released
```

ALGORITHM:

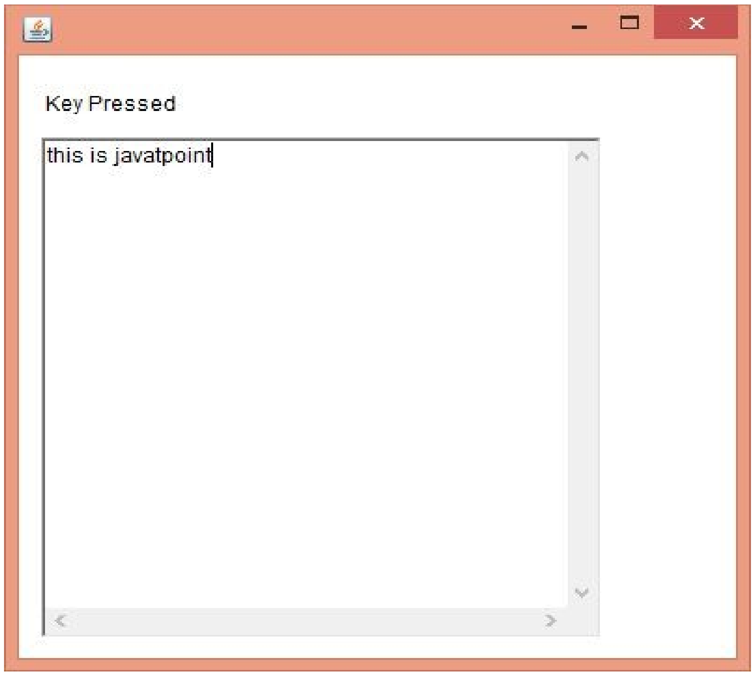
```
STEP1: START
STEP2: Create a class extending from a Frame and implementing KeyListener.
STEP3: Create a TextArea and add KeyListener to it.
STEP4: Set bounds of the text area using setBounds() method.
STEP5: Set size to the frame using setSize() method.
STEP6: Override the following methods
        a) keyPressed()
        b) keyReleased()
        c) keyTyped()
STEP7: Using setText() method display the message in the text area.
```

STEP8: Display the necessary information on the screen.
STEP9: END

SOURCE CODE:

```
import java.awt.*;
import java.awt.event.*;
public class KeyListenerExample extends Frame implements KeyListener{
    Label l;
    TextArea area;
    KeyListenerExample(){
        l=new Label();
        l.setBounds(20,50,100,20);
        area=new TextArea();
        area.setBounds(20,80,300, 300);
        area.addKeyListener(this);
        add(l);
         add(area);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
    public void keyPressed(KeyEvent e) {
        l.setText("Key Pressed");
    }
    public void keyReleased(KeyEvent e) {
        l.setText("Key Released");
    }
    public void keyTyped(KeyEvent e) {
        l.setText("Key Typed");
    }

    public static void main(String[] args) {
        new KeyListenerExample();
        }
    }
```

OUTPUT:



Key Pressed

```
this is javatpoint
```

VIVA - VOCE:

1. What is the purpose of adapter classes?

Java adapter classes provide the default implementation of listener interfaces. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. The adapter classes are found in java.awt.event and javax.swing.event packages.

2. What is the purpose of MouseListener?

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods. Methods of MouseListener interface are given below:

public abstract void mouseClicked(MouseEvent e);

public abstract void mouseEntered(MouseEvent e);

public abstract void mouseExited(MouseEvent e);

public abstract void mousePressed(MouseEvent e);

public abstract void mouseReleased(MouseEvent e);.

3. What is a the purpose of MouseMotionListener?

The Java MouseMotionListener is notified whenever you move or drag mouse. It is notified against MouseEvent. The MouseMotionListener interface is found in java.awt.event package. It has two methods.Methods of MouseMotionListener interface are given below:

public abstract void mouseDragged(MouseEvent e);

public abstract void mouseMoved(MouseEvent e);

4. What is a the purpose of KeyListener?

The Java KeyListener is notified whenever you change the state of key. It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package. It has three methods.

Methods of KeyListener interface are given below:

public abstract void keyPressed(KeyEvent e);

public abstract void keyReleased(KeyEvent e);

public abstract void keyTyped(KeyEvent e);

5. What is the purpose of showStatus() method and repaint() method?

The **showStatus( )** method is used to display the information in applet windows status bar of the browser or Appletviewer. This is used for debugging, because it gives you an easy way to output message.

The **repaint( )** method causes the AWT runtime system to execute the update () method of the Component class which clears the window with the background color of the applet and then calls the paint () method.

6. What is the difference between Frame and JFrame?

A) Frame is part of java.awt package and exists since JDK1.0. JFrame is part of javax.swing package and exists since JDK1.1.3 or something. Frame extends Window. JFrame extends Frame. You can directly add components to Frame. You add components to JFrame.getContentPane(). JFrame implements javax.swing.RootPane Container. Frame does not.

WEEK 8:

(A)AIM: To develop an applet in Java that displays a simple message.

THEORY: Applets are designed to bring the web alive. They function to add animation sound and eventually complete multi-media into HTML documents. Java is also part of the future of interfacing with virtual reality environments. At present, java is limited only by the capabilities of the internet itself. Applets are java programs that are specialized for use over the Web.

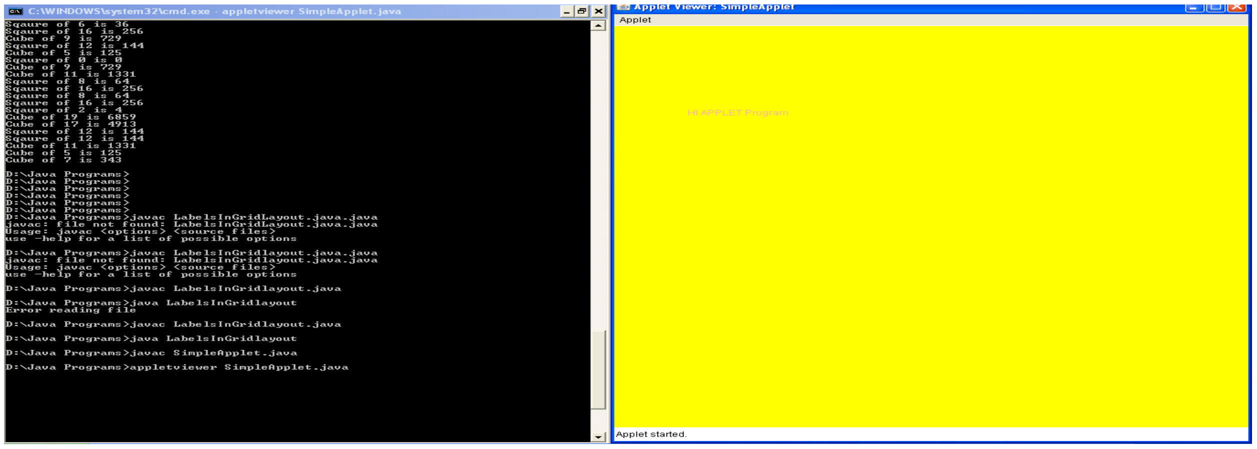
ALGORITHM:

```
STEP1: START
STEP2: Create an applet using extends Applet class.
STEP3: Using drawString() method in the paint() method, display the simple message in required coordinates.
STEP4: Include the applet tag in the HTML code
STEP5: Execute an applet using the web-browser.
STEP6: END
```

SOURCE CODE:

```
import java.applet.*;
import java.awt.*;
/*<applet code="SimpleApplet" height=300 width=300>
</applet>*/
public class SimpleApplet extends Applet{
        public void paint(Graphics g)
        {
                g.setColor(Color.pink);
                setBackground(Color.yellow);
                g.drawString("HI APPLET Program",80,150);
}
}
```

OUTPUT:


C:\WINDOWS\system32\cmd.exe - appletviewer SimpleApplet.java
Applet Viewer: SimpleApplet
Applet
HI APPLET Program
Applet started.

```
Sqaure of 6 is 36
Sqaure of 16 is 256
Cube of 9 is 729
Sqaure of 12 is 144
Cube of 5 is 125
Sqaure of 0 is 0
Cube of 9 is 729
Cube of 11 is 1331
Sqaure of 8 is 64
Sqaure of 16 is 256
Sqaure of 8 is 64
Sqaure of 16 is 256
Sqaure of 2 is 4
Cube of 19 is 6859
Cube of 17 is 4913
Sqaure of 12 is 144
Sqaure of 12 is 144
Cube of 11 is 1331
Cube of 5 is 125
Cube of 7 is 343

D:\Java Programs>
D:\Java Programs>
D:\Java Programs>
D:\Java Programs>
D:\Java Programs>
D:\Java Programs>
D:\Java Programs>javac LabelsInGridLayout.java.java
javac: file not found: LabelsInGridLayout.java.java
Usage: javac <options> <source files>
use -help for a list of possible options

D:\Java Programs>javac LabelsInGridlayout.java.java
javac: file not found: LabelsInGridlayout.java.java
Usage: javac <options> <source files>
use -help for a list of possible options

D:\Java Programs>javac LabelsInGridlayout.java

D:\Java Programs>java LabelsInGridlayout
Error reading file

D:\Java Programs>javac LabelsInGridlayout.java

D:\Java Programs>java LabelsInGridlayout

D:\Java Programs>javac SimpleApplet.java

D:\Java Programs>appletviewer SimpleApplet.java
```

(B)AIM: To applet to compute factorial value

THEORY: The Applet life cycle: The init()Method: The init()method is where your applet does much of its setup such as defined its layout, parsing parameters or setting the background colors. The start() Method: The start()method is used mainly when implementing threads in java. The stop() Method: The stop() method is used to do what its name suggests: stop what is going on. The destroy() method: when it is called, the applet is told to free up system resources.

ALGORITHM:

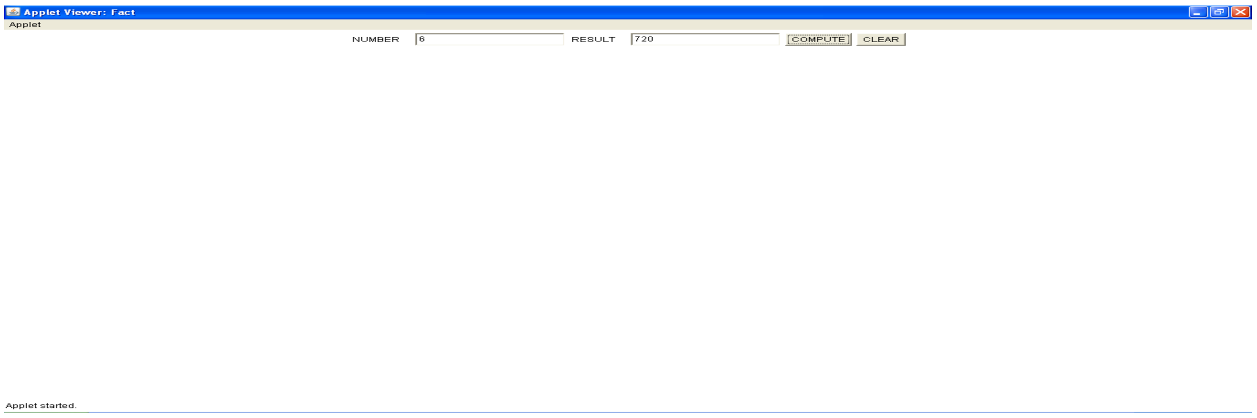
```
STEP1: START
STEP2: Create an applet using extends Applet class.
STEP3: Create two textfields , one for to enter the number and another for result purpose.
STEP4: Create two buttons, one for compute and another button to clear the textfields data.
STEP5: Get the values from the textfield using getText() method.
STEP6: Compute the factorial value and set the result in the Result textfield.
STEP7: Include the applet tag in html and run the applet in the browser.
STEP8: END
```

SOURCE CODE:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*<applet code=Fact width=500 height=500> </applet>*/
public  class Fact extends Applet implements ActionListener
{
        Button b1,b2;
```

```
Label l1,l2;
TextField tf1,tf2;
public void init()
{
        b1=new Button("COMPUTE");
        b1.addActionListener(this);
        b2=new Button("CLEAR");
        b2.addActionListener(this);
        tf1=new TextField(20);
        tf2=new TextField(20);
        l1=new Label("NUMBER");
        l2=new Label("RESULT");
        add(l1);
        add(tf1);
        add(l2);
        add(tf2);
        add(b1);
        add(b2);
}
public void actionPerformed(ActionEvent e)
{
        if(e.getSource()==b1)
        {
                int a=Integer.parseInt(tf1.getText());
                int fact=1;
                for(int i=1;i<=a;i++)
                fact*=i;
                tf2.setText(""+fact);
        }
        else
        {
                tf1.setText("");
                tf2.setText("");
} } }
```

OUTPUT:



VIVA - VOCE:

1. What is the purpose of setText() and getText() methods?

setText() method is used to set text into the TextField specified through object. It's return type is void. getText() method is used to get text from the TextField specified through object. It's return type is String.

2. What is the purpose of getSource() method?

getSource() method returns the command string associated with this action. getSource() returns the object on which the Event initially occurred.

3. What is a the purpose of parseInt()?

The method generally used to convert String to Integer in Java is parseInt(). public static int parseInt(String s) throws NumberFormatException - This function parses the string argument as a signed decimal integer.

4. What is the use of actionListener()?

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().

5. What is an applet?

Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. Applet is embedded in a HTML page using the APPLET or OBJECT tag and hosted on a web server.

6. What are the Life cycle methods of Applet?

When an applet begins, the following methods are called, in this sequence:

1. init( )
2. start( )
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:
1. stop( )
2. destroy( )

WEEK 9

(A)AIM: To write a Java program that simulates a traffic light.

THEORY: The AWT supports a rich assortment of graphics methods. All graphics are drawn relative to a window. These can the main window of an applet, a child window of an applet, or a stand-alone application window. The origin of each window is at the top-left corner and is 0,0 coordinates are specified in pixels. All output to a window takes place through graphics context.

ALGORITHM:

```
STEP1: START
STEP2: Create an applet using extends Applet class.
STEP3: Create three checkboxes for red, orange and green and make them into one group.
STEP4: Implement the itemStateChanged() method whenever the checkbox selection is changed.
STEP5: Draw three ovals initially filled with black.
STEP6: Fill the ovals with appropriate colors on the selection of checkbox.
STEP7: Display the appropriate message beside the filled oval.
STEP8: END
```

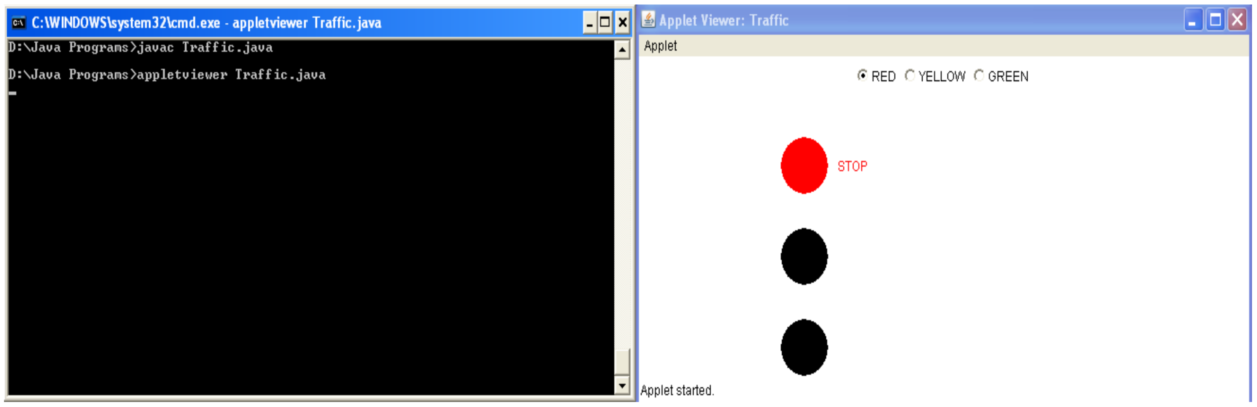
SOURCE CODE:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="Traffic" width=400 height=400>
</applet>*/
public class Traffic extends Applet implements ItemListener
{
   int colourNum;
   CheckboxGroup cbg;
   Checkbox red,yellow,green;
   String msg=" ";
   public void init()
   {
         cbg=new CheckboxGroup();
         red=new Checkbox("RED",cbg,true);
         yellow=new Checkbox("YELLOW",cbg,true);
         green=new Checkbox("GREEN",cbg,true);
         add(red);
         add(yellow);
         add(green);
         red.addItemListener(this);
```

```
        yellow.addItemListener(this);
        green.addItemListener(this);
    }
     public void itemStateChanged(ItemEvent ie)
     {
              if (ie.getSource()==red)
              colourNum=1;
        else if (ie.getSource()==yellow)
              colourNum=2;
        else
              colourNum=3;
        repaint();
     }
    public void paint (Graphics g)
    {
      g.setColor(Color.BLACK);
      g.fillOval (150, 70, 50, 50); // red light
      g.fillOval (150, 150, 50, 50); // yellow light
      g.fillOval (150, 230, 50, 50); // green light
      switch (colourNum)
      {
        case 1:g.setColor (Color.RED);
          g.fillOval (150,70,50,50);
              msg="STOP";
              g.drawString(msg,210,100);
              break;
        case 2:g.setColor(Color.YELLOW);
              g.fillOval (150,150,50,50);
              g.setColor (Color.red);
              msg="READY";
              g.drawString(msg,210,180);
              break;
        case 3:g.setColor(Color.GREEN);
```

```
                g.fillOval (150,230,50,50);
                g.setColor (Color.red);
                msg="GO";
                g.drawString(msg,210,260);
                break;
        }
    }
}
```

OUTPUT:



```
D:\Java Programs>javac Traffic.java

D:\Java Programs>appletviewer Traffic.java
```

VIVA - VOCE:

1. What is a Checkbox?

The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

2. What is a CheckboxGroup?

The object of CheckboxGroup class is used to group together a set of Checkbox. At a time only one check box button is allowed to be in "on" state and remaining check box button in "off" state. It inherits the object class.

3. What is the use of this keyword?

THIS is a reference variable in Java that refers to the current object. It can be used to refer instance variable of current class. It can be used to invoke or initiate current class constructor. It can be passed as an argument in the method call.

4. What is the purpose of Graphics class?

It's essential that programmers understand the Graphics class before they attempt to draw images via Java. The Graphics class provides the framework for all graphics operations within the AWT. Because the Graphics class is an abstract base class, it cannot be instantiated directly.

5. What is the purpose of AWT?

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS. he java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

6. What does java.awt.event package caontain?

The java.awt.event package defines classes and interfaces used for event handling in the AWT and Swing. Events are fired by event sources. An event listener registers with an event source to receive notifications about the events of a particular type. This package defines events and event listeners, as well as event adapter classes, which are convenience classes to make easier the process of writing event listeners.

(B)AIM: To write a java program to demonstrate Hashtable usage

THEORY: It is sub class of Dictionary. It allows data to be stored in the form of key/value pairs. Java Hashtable class implements a hashtable, which maps keys to values. It inherits Dictionary class and implements the Map interface.

Points to remember

- 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.
- The initial default capacity of Hashtable class is 11 whereas loadFactor is 0.75.

ALGORITHM:

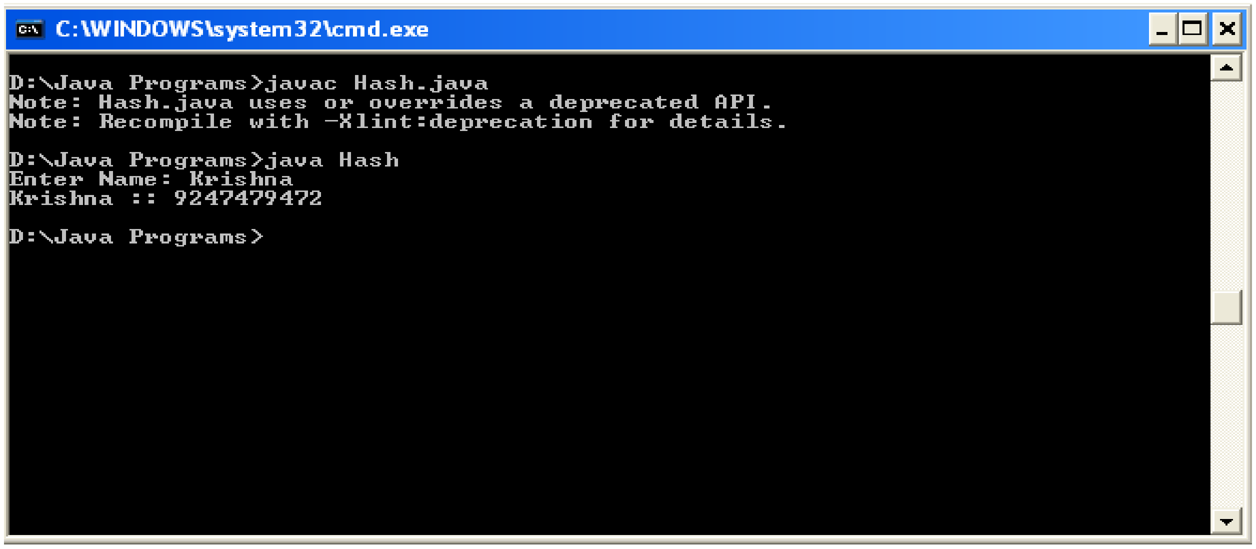
```
STEP1: START
STEP2: Create a class named Hash.
STEP3: Create a text file which contains names and phone numbers are separated by a
tab.
STEP4: Using DataInputStream/Scanner, the file has to be read and the content has to be
moved to hash  table using put() method.
STEP5: Whenever a name is given the corresponding phone number from hash table has
to be displayed else not found to be displayed.
STEP6: END
```

SOURCE CODE:

```
import java.io.*;
import java.util.*;
public class Hash
{
   public static void main(String args[]) throws IOException
   {
     Hashtable<String,String> ht=new Hashtable<String,String>();
         String s1,s2;
         Scanner s = new Scanner(new FileReader("test.txt"));
         while(s.hasNext())
         {
         s1=s.next();
         s2=s.next();
         ht.put(s1,s2);
         }
         DataInputStream dis=new DataInputStream(System.in);
         System.out.print("Enter Name: ");
         String name=dis.readLine();
         name=(String)name.trim();
         String phoneNo=ht.get(name);
         if(phoneNo!=null)
         {
```

```
System.out.println(name+" :: "+phoneNo);
}
else
{
        System.out.println("Not Found");
}}}
```

OUTPUT:



```
D:\Java Programs>javac Hash.java
Note: Hash.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

D:\Java Programs>java Hash
Enter Name: Krishna
Krishna :: 9247479472

D:\Java Programs>
```

VIVA VOCE:

1.What is the purpose of Hashtable?

Java Hashtable class implements a hashtable, which maps keys to values. It inherits Dictionary class and implements Map interface. The important points about Java Hashtable are:

1. A Hashtable is an array of list. Each list is known as a bucket. The position of bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key.
2. It contains only unique elements.
3. It may have not have any null key or value.
4. It is synchronized.

2. What is the purpose of DataInputStream class?

A data input stream enable an application read primitive Java data types from an underlying input stream in a machine-independent way (instead of raw bytes). That is why it is called DataInputStream – because it reads data (numbers) instead of just bytes.

3. What is a the purpose of trim() method?

The java string trim() method eliminates leading and trailing spaces. The unicode value of space character is '\u0020'. The trim() method in java string checks this unicode value before and after the string, if it exists then removes the spaces and returns the omitted string.

4. What is the use of Generics?

Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively. Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time. Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

5. What is the use of put() and get()?

put() method inserts a key and a value into the hash table. Returns null if the key isn't already in the hash table; returns the previous value associated with the key if the key is already in the hash table.

get() method returns the object that contains the value associated with the key. If the key is not in the hash table, a null object is returned.

6. What does the java.util package contain?

Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (stringtokenizer, random-number generator, arraylist, vector, dictionary, hashtable, calendar, map, iterator and enumerator etc).

WEEK 10

(A)AIM: To write a Java program to display the table using Labels in Grid Layout.

THEORY: GridLayout is one of the Layout managers. A layout manager automatically arranges your controls with in a window by using some type of algorithm. Grid Layout lays out component in a two-dimensional grid. When you instantiate a GridLayout, you define the number of rows and columns.

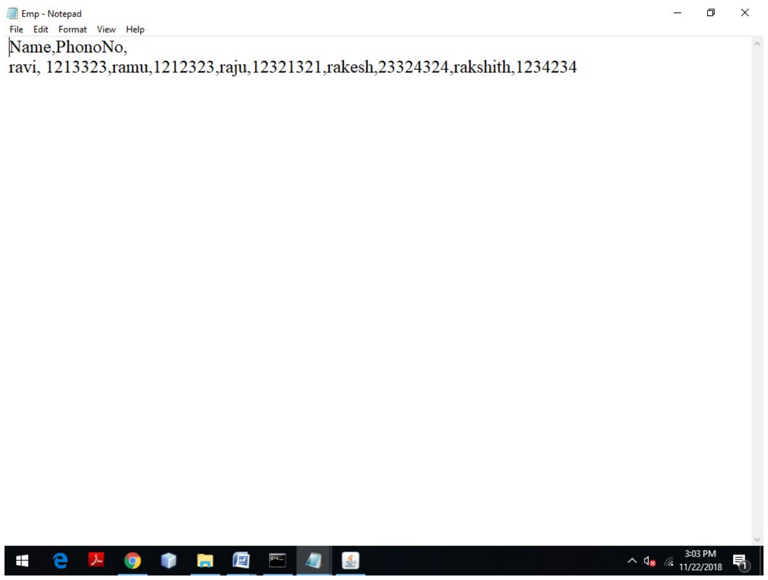
ALGORITHM:

```
STEP1: START
STEP2: Create a class named LabelsInGridLayout.
STEP3: Create a file named table.txt.
STEP4: Create a Frame with a header.
STEP5: Set GridLayout and add data read from a table named Table.txt file.
STEP6: Set the setSize() and setVisible() methods.
STEP7: Set the default close operation to close the frame and execute.
STEP8: END
```

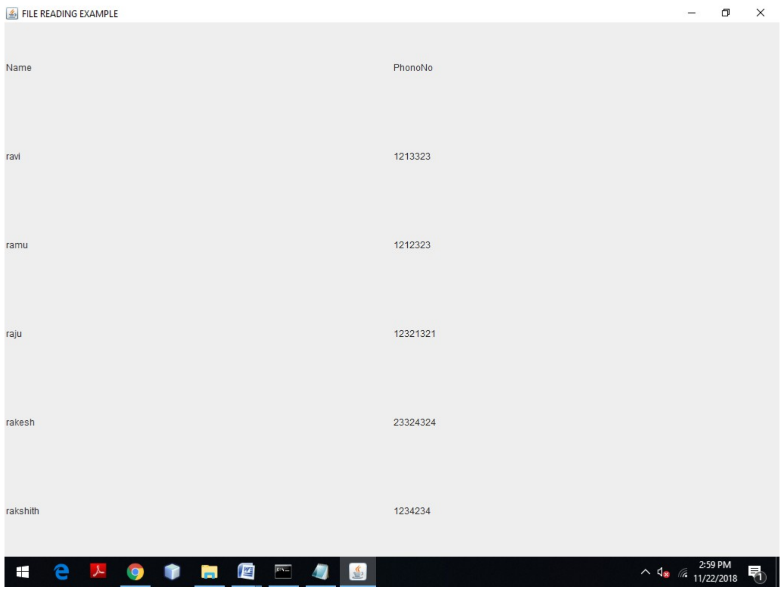
SOURCE CODE:

```
import java.awt.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
public class DisplayTable
{
        public static void main(String[] args)
        {
        JFrame f = new JFrame("FILE READING EXAMPLE");
                try
                {
                Scanner st=new Scanner(new FileReader("Emp.txt"));
                st.useDelimiter("\\s*,\\s*");
                        while(st.hasNext())
                        {
                                f.add(new Label(st.next()));
                        f.setLayout(new GridLayout(6,2));
                                f.setSize(400,200);
                                f.setVisible(true);
                        }
                }
                 catch (Exception ex)
                {
                        System.out.println("Error reading file ");
                }
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```

OUTPUT:



```
Name,PhonoNo,
ravi, 1213323,ramu,1212323,raju,12321321,rakesh,23324324,rakshith,1234234
```



```
Name        PhonoNo
ravi        1213323
ramu        1212323
raju        12321321
rakesh      23324324
rakshith    1234234
```

(B)AIM: To write a Java program that works as a simple calculator.

THEORY: GridLayout is one of the Layout managers. A layout manager automatically arranges your controls with in a window by using some type of algorithm. Grid Layout lays out component in a two dimensional grid. When you instantiate a GridLayout, you define the number of rows and columns.

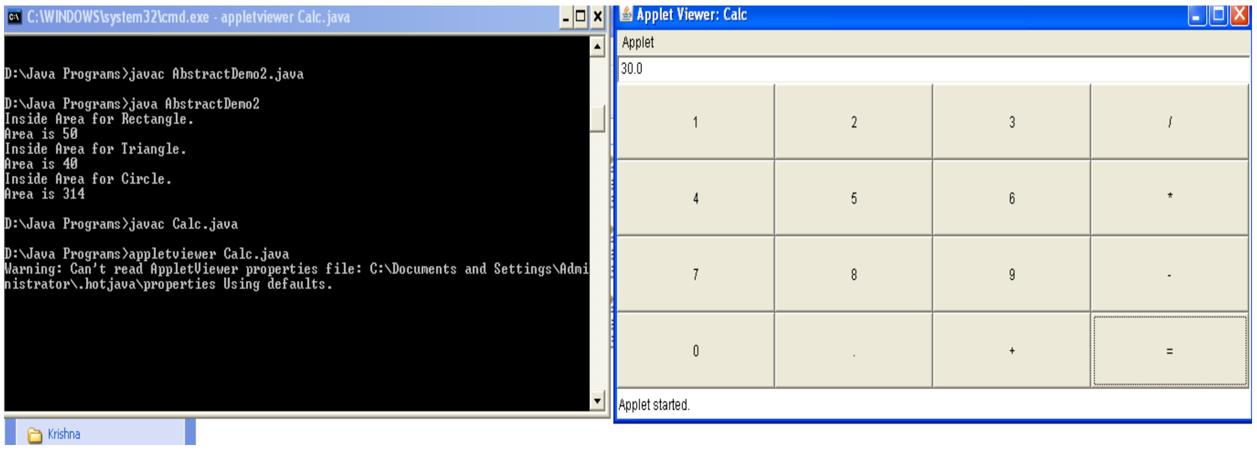
ALGORITHM:

STEP1: START
STEP2: Create an applet using extends Applet class.
STEP3: Create buttons using Buttons() and text field using TextField() classes.
STEP4: Add all the buttons in the required order to a panel.
STEP5: Use GridLayout and place the buttons in this layout by using add().
STEP6: Finally add the TextField() and Panel to the window.
STEP7: Implement the actionPerformed() method.
STEP8: END

SOURCE CODE:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="Calc" height=300 width=300>
</applet>
*/
public class Calc extends Applet implements ActionListener
{
        TextField tf;
        double arg=0;
        String op="=";
        boolean start=true;
        public void init()
        {
                setLayout(new BorderLayout());
                tf=new TextField("0");
                add(tf,BorderLayout.NORTH);
                Panel p=new Panel();
                p.setLayout(new GridLayout(4,4));
                String buttons="123/456*789-0.+=";
                for(int i=0;i<buttons.length();i++)
                {
                    Button b=new Button(buttons.substring(i,i+1));
                    p.add(b);
                    b.addActionListener(this);
                }
                add(p);
        }
        public void actionPerformed(ActionEvent ae)
        {
                String s=ae.getActionCommand();
                if('0'<=s.charAt(0)&&s.charAt(0)<='9'||s.equals("."))
                {
                    if(start)
                    tf.setText(s);
                    else
                    tf.setText(tf.getText()+s);
                    start=false;
                }
                else
                {
                    calcu(Double.parseDouble(tf.getText()));
                    op=s;
                    start=true;
                }
        }
        public void calcu(double n)
        {
                if(op.equals("+"))
                arg+=n;
                else
                if(op.equals("-"))
                arg-=n;
                else
                if(op.equals("*"))
                arg*=n;
                else
                if(op.equals("/"))
                {
                try{
                      arg/=n;
                   }
                  catch(ArithmeticException e)
                 {
                  tf.setText("Arithmetic Exception");
                 }
                }
                else
                if(op.equals("="))
                arg=n;
                tf.setText(""+arg);
        }}
```

OUTPUT:



```
C:\WINDOWS\system32\cmd.exe - appletviewer Calc.java

D:\Java Programs>javac AbstractDemo2.java

D:\Java Programs>java AbstractDemo2
Inside Area for Rectangle.
Area is 50
Inside Area for Triangle.
Area is 40
Inside Area for Circle.
Area is 314

D:\Java Programs>javac Calc.java

D:\Java Programs>appletviewer Calc.java
Warning: Can't read AppletViewer properties file: C:\Documents and Settings\Admi
nistrator\.hotjava\properties Using defaults.
```

```
Applet Viewer: Calc
Applet
30.0
Applet started.
```

VIVA VOCE:

1. What is the difference between TextField and TextArea?

The object of a TextField class is a text component that allows the editing of a single line text. It inherits TextComponent class. On the other hand, the object of a TextArea class is a multi line region that displays text. It allows the editing of multi-line text. It inherits TextComponent class.

2. What is a Button and a Label?

The button class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. The object of Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.

3. What is the use of Panel?

The Panel is a simplest container class. It provides space in which an application can attach any other component. It inherits the Container class. It doesn't have title bar.

4. What is the purpose of Layout Manager?

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 following are the commonly used layout managers: BorderLayout, FlowLayout, GridLayout, CardLayout and

GridBagLayout.

5. What is the use of GridLayout?

The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle.

6. What is an Exception and tell about ArithmeticException?

Exception is an abnormal condition. In java, Exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime. Thrown when an exceptional arithmetic condition has occurred. For example, an integer "divide by zero" throws an instance of this class. ***ArithmeticException*** objects may be constructed by the virtual machine as if suppression was disabled and/or the stack trace was not writable.

Week 11

AIM: To develop Swing application which uses JList, JTree, JTable, JTabbedPane and JScrollPane

THEORY: Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT. In addition to the familiar components, such as buttons, check boxes, and labels, Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables. Even familiar components such as buttons have more capabilities in Swing. For example, a button may have both an image and a text string associated with it. Also, the image can be changed as the state of the button changes. Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and, therefore, are platform-independent.

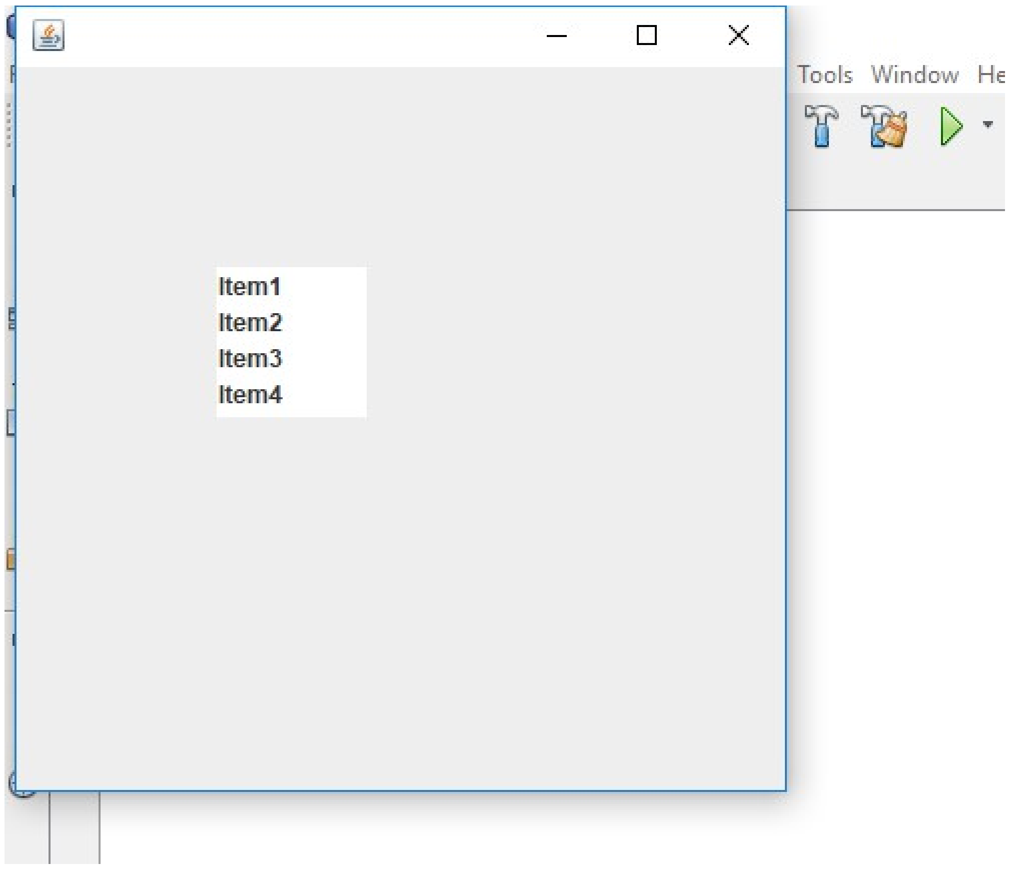
ALGORITHM:JList

```
STEP1: START
STEP2: Create the List of elements using DefaultListModel.
STEP3: Add above created  list to JList.
STEP4: Create a JFrame and add JList to the JFrame.
STEP5: Set the JFrame size and set setVisible(true) to make the JList visible.
STEP6: END
```

SOURCE CODE: JList

```
import javax.swing.*;
   public class ListExmp
   {
       ListExmp(){
         JFrame f= new JFrame();
         DefaultListModel<String> l1 = new DefaultListModel<>();
           l1.addElement("Item1");
           l1.addElement("Item2");
           l1.addElement("Item3");
           l1.addElement("Item4");
           JList<String> list = new JList<>(l1);
           list.setBounds(100,100, 75,75);
           f.add(list);
           f.setSize(400,400);
           f.setLayout(null);
           f.setVisible(true);
           f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       }

 public static void main(String args[])
      {
     new ListExmp();
      }
}
```

OUTPUT: JList



```
Item1
Item2
Item3
Item4
```

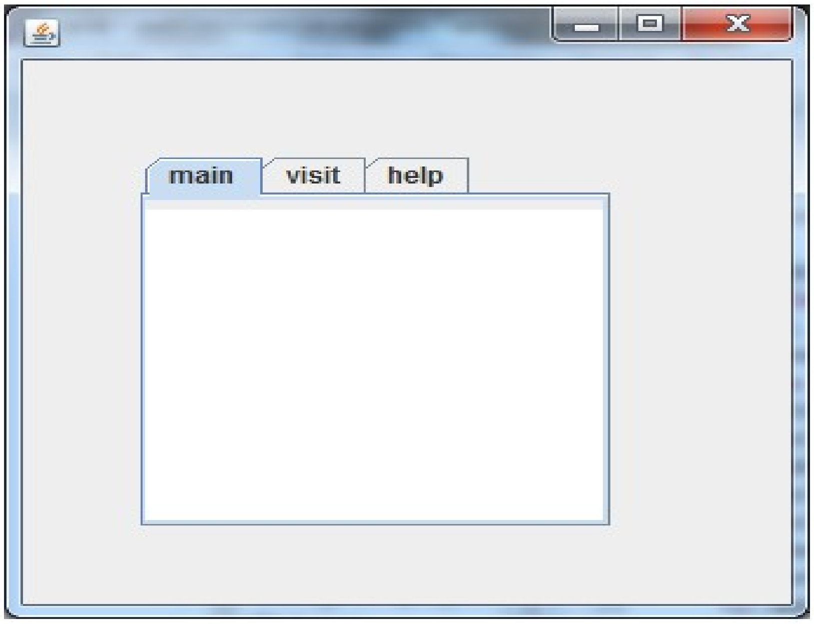
ALGORITHM: JTabbedPane
STEP1: START
STEP2: Create a TextArea with required bounds and add it to a panel.
STEP3: Create other two panels with some information to distinguish from one another.
STEP4: Create a JTabbedPane with required bounds and add the three panels to JTabbedPane object.
STEP5: Create a JFrame and add JTabbedPane to the JFrame.
STEP6: Set the JFrame size and set setVisible(true) to make JTabbedPane visible.
STEP7: END

SOURCE CODE: JTabbedPane

```
 import javax.swing.*;
  public class TabbedPaneExample {
  JFrame f;
  TabbedPaneExample(){
      f=new JFrame();
      JTextArea ta=new JTextArea(200,200);
      JPanel p1=new JPanel();
```

```
        p1.add(ta);
        JPanel p2=new JPanel();
        JPanel p3=new JPanel();
        JTabbedPane tp=new JTabbedPane();
        tp.setBounds(50,50,200,200);
        tp.add("main",p1);
        tp.add("visit",p2);
        tp.add("help",p3);
        f.add(tp);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new TabbedPaneExample();
    }  }
```

JTabbedPane OUTPUT:



ALGORITHM: JScrollPane

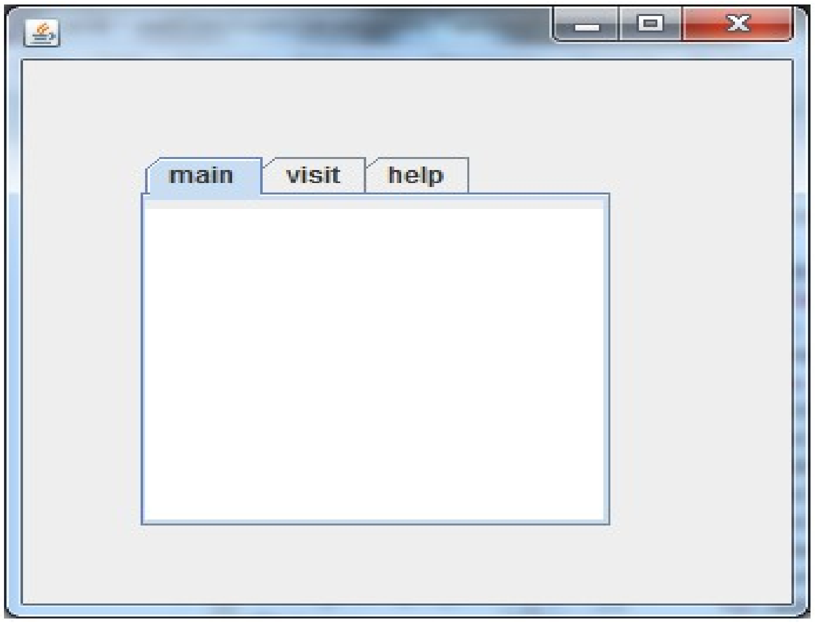
STEP1: START
STEP2: Create a Panel and set panel Layout to the GridLayout.
STEP3: Create array of buttons and add them to the Panel.
STEP4: Create horizontal and vertical ScrollPaneConstants.
STEP5: Create a JScrollPane and set panel, horizontal and vertical ScrollPaneConstants.
STEP6: Get the contentPane object and add the JScrollPane object to it with required layout.
STEP7: END

SOURCE CODE: JScrollPane

```
import javax.swing.*;
public class TabbedPaneExample {
JFrame f;
TabbedPaneExample(){
    f=new JFrame();
    JTextArea ta=new JTextArea(200,200);
    JPanel p1=new JPanel();
    p1.add(ta);
    JPanel p2=new JPanel();
    JPanel p3=new JPanel();
    JTabbedPane tp=new JTabbedPane();
```

```
        tp.setBounds(50,50,200,200);
        tp.add("main",p1);
        tp.add("visit",p2);
        tp.add("help",p3);
        f.add(tp);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new TabbedPaneExample();
    } }
```

OUTPUT: JScrollPane



s

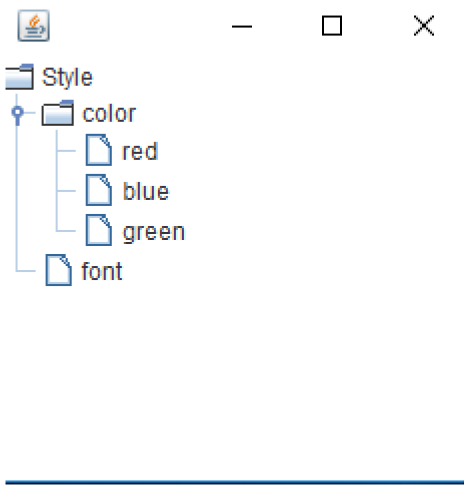
ALGORITHM: JTree
STEP1: START

STEP2: Create a Panel and set panel Layout.
STEP3: Create JTree to the Panel.
STEP4: set the properties of the JTree class
STEP5: END

SOURCE CODE: JTree

```
  import javax.swing.*;
  import javax.swing.tree.DefaultMutableTreeNode;
  public class TreeEx
{
  JFrame f;
  TreeEx(){
    f=new JFrame();
    DefaultMutableTreeNode style=new DefaultMutableTreeNode("Style");
    DefaultMutableTreeNode color=new DefaultMutableTreeNode("color");
    DefaultMutableTreeNode font=new DefaultMutableTreeNode("font");
    style.add(color);
    style.add(font);
    DefaultMutableTreeNode red=new DefaultMutableTreeNode("red");
    DefaultMutableTreeNode blue=new DefaultMutableTreeNode("blue");
    DefaultMutableTreeNode green=new DefaultMutableTreeNode("green");
    color.add(red);
    color.add(blue);
    color.add(green);
    JTree jt=new JTree(style);
    f.add(jt);
    f.setSize(250,250);
    f.setVisible(true);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public static void main(String[] args) {
        new TreeEx();
    }
}
```

OUTPUT: JTree



ALGORITHM: JTable

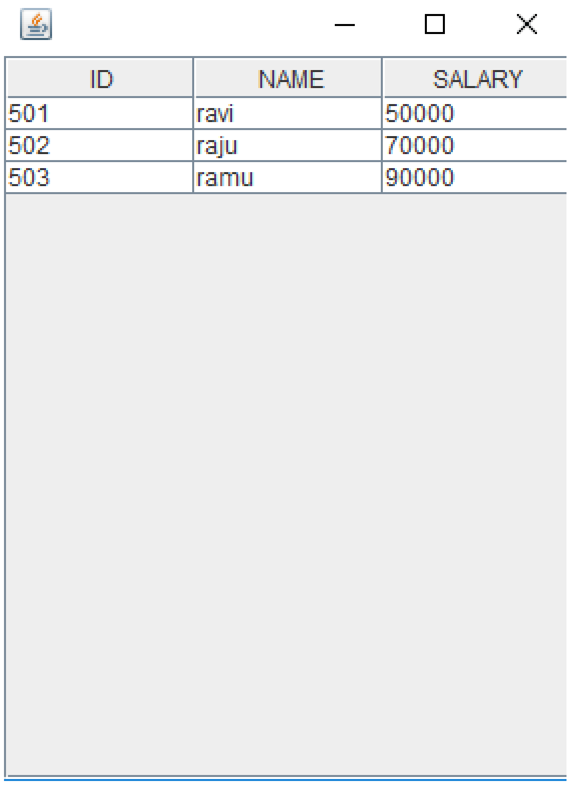
```
STEP1: START
STEP2: Create a Panel and set panel Layout.
STEP3: Add JTable to the Panel.
STEP4: Set the JTable rows and columns .
STEP5: Load the data into the JTable
STEP6: END
```

SOURCE CODE: JTable

```
import javax.swing.*;
public class TableEx
{
    JFrame f;
    TableEx()
    {
    f=new JFrame();
    String data[][]={ {"501","ravi","50000"},
                      {"502","raju","70000"},
                      {"503","ramu","90000"}};
```

```
        String column[]={"ID","NAME","SALARY"};
        JTable jt=new JTable(data,column);
        JScrollPane sp=new JScrollPane(jt);
        f.add(sp);
        f.setSize(300,400);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        public static void main(String[] args)
       {
        new TableEx();
       }
 }
```

OUTPUT: JTable



ID	NAME	SALARY
501	ravi	50000
502	raju	70000
503	ramu	90000

VIVA VOCE:

1. What is the purpose of Swings in Java?

Java Swing 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.

2. What is the purpose of JList class?

The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.

3. What is the purpose of JTabbedPane class?

The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon. It inherits JComponent class.

4. What is the purpose of JScrollPane class?

A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.