

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*.

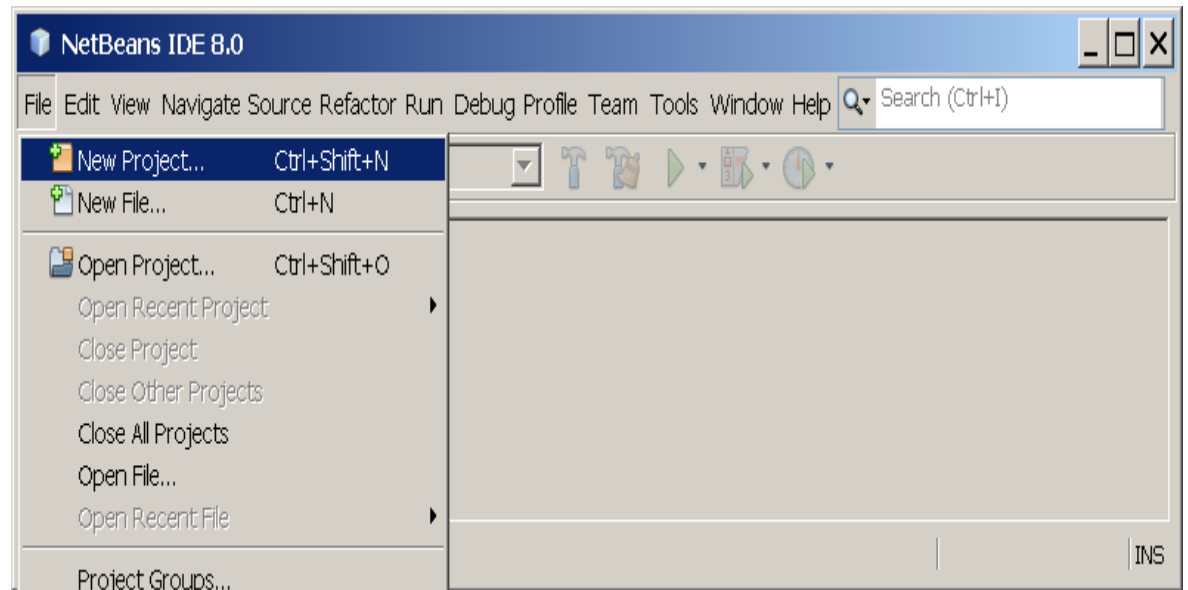
- **Run the program**

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

Create an IDE Project

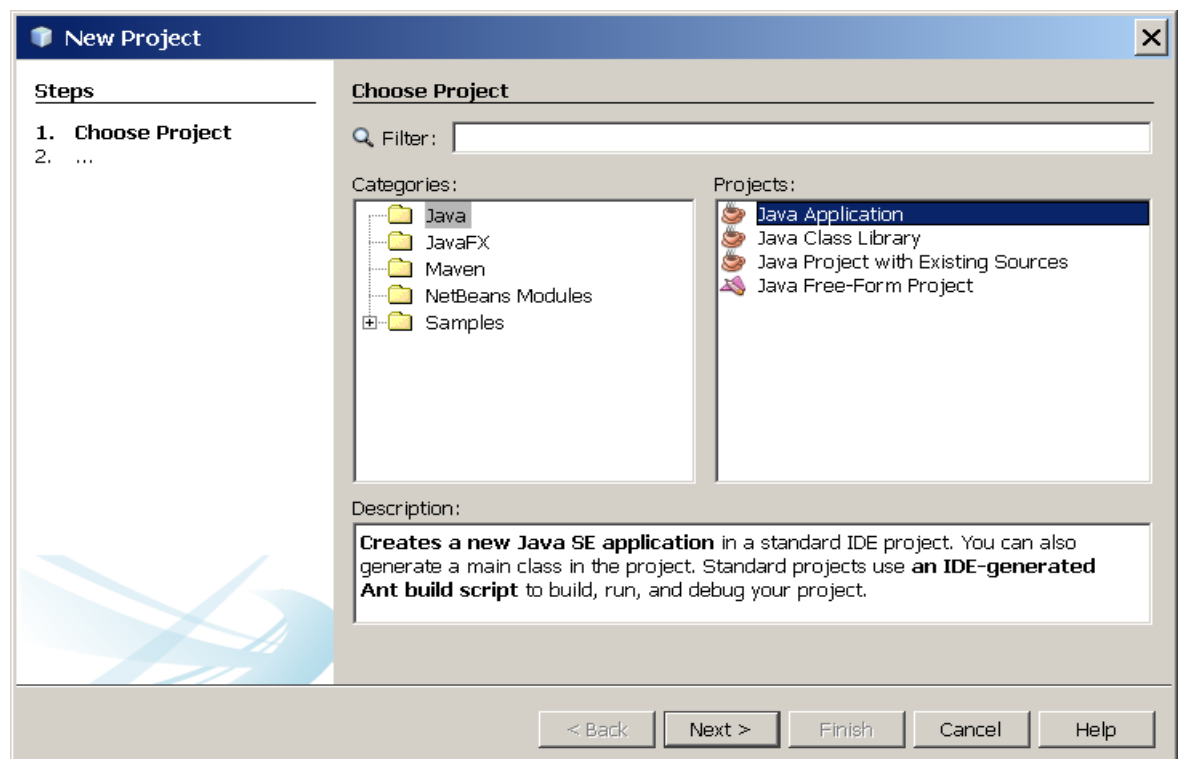
To create an IDE project:

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...**



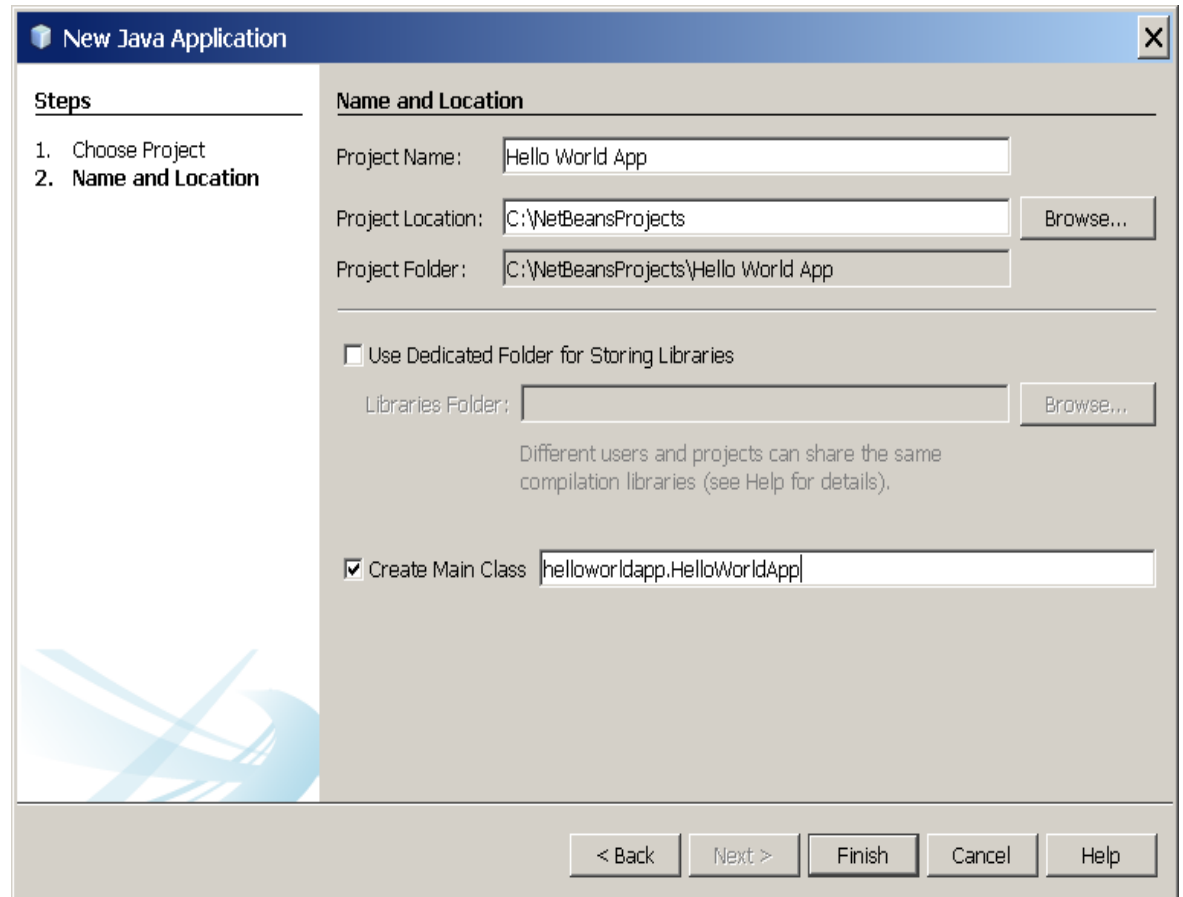
NetBeans IDE with the File | New Project menu item selected.

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



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.

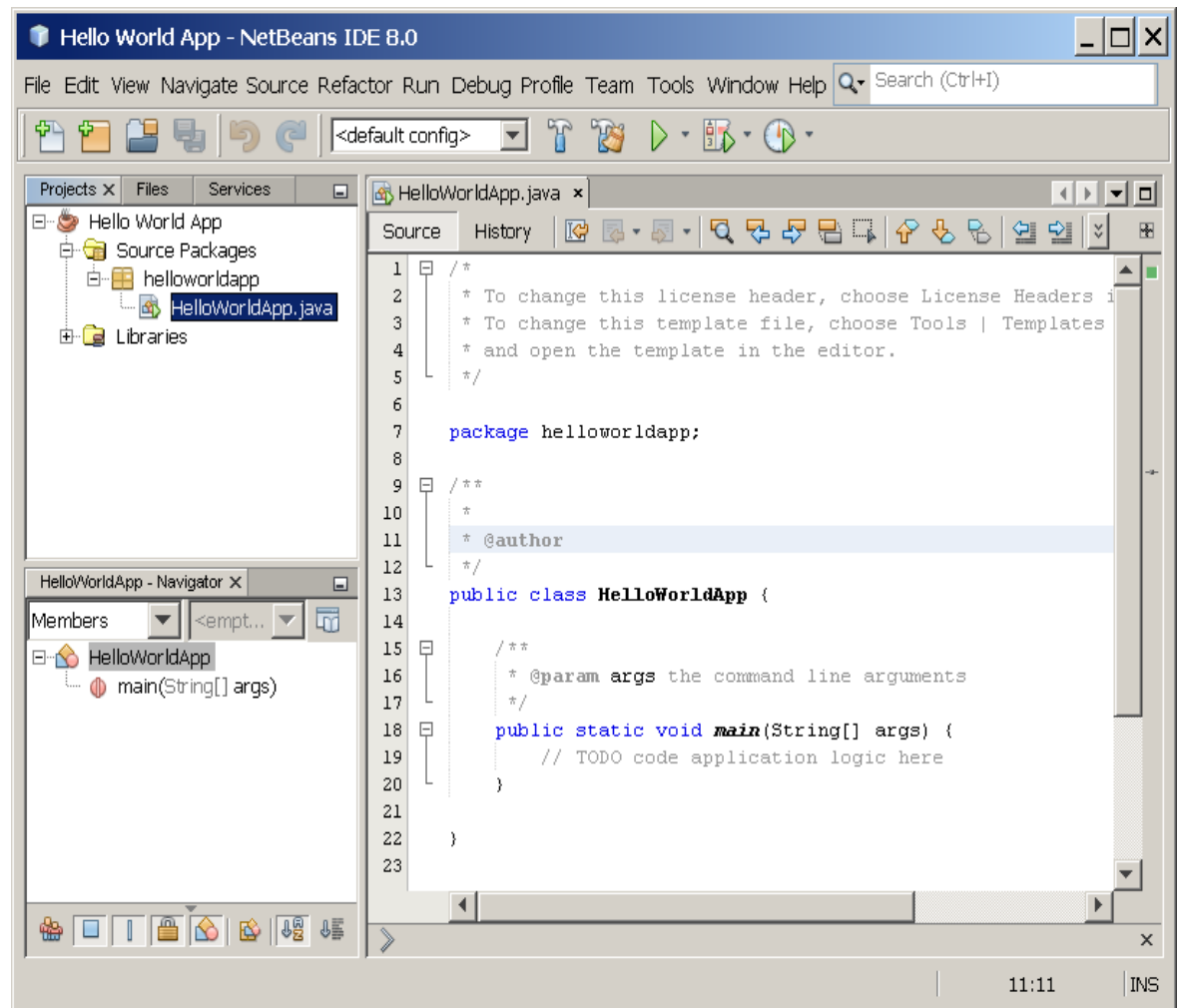


NetBeans IDE, New Project wizard, Name and Location page.

5. Click Finish.

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

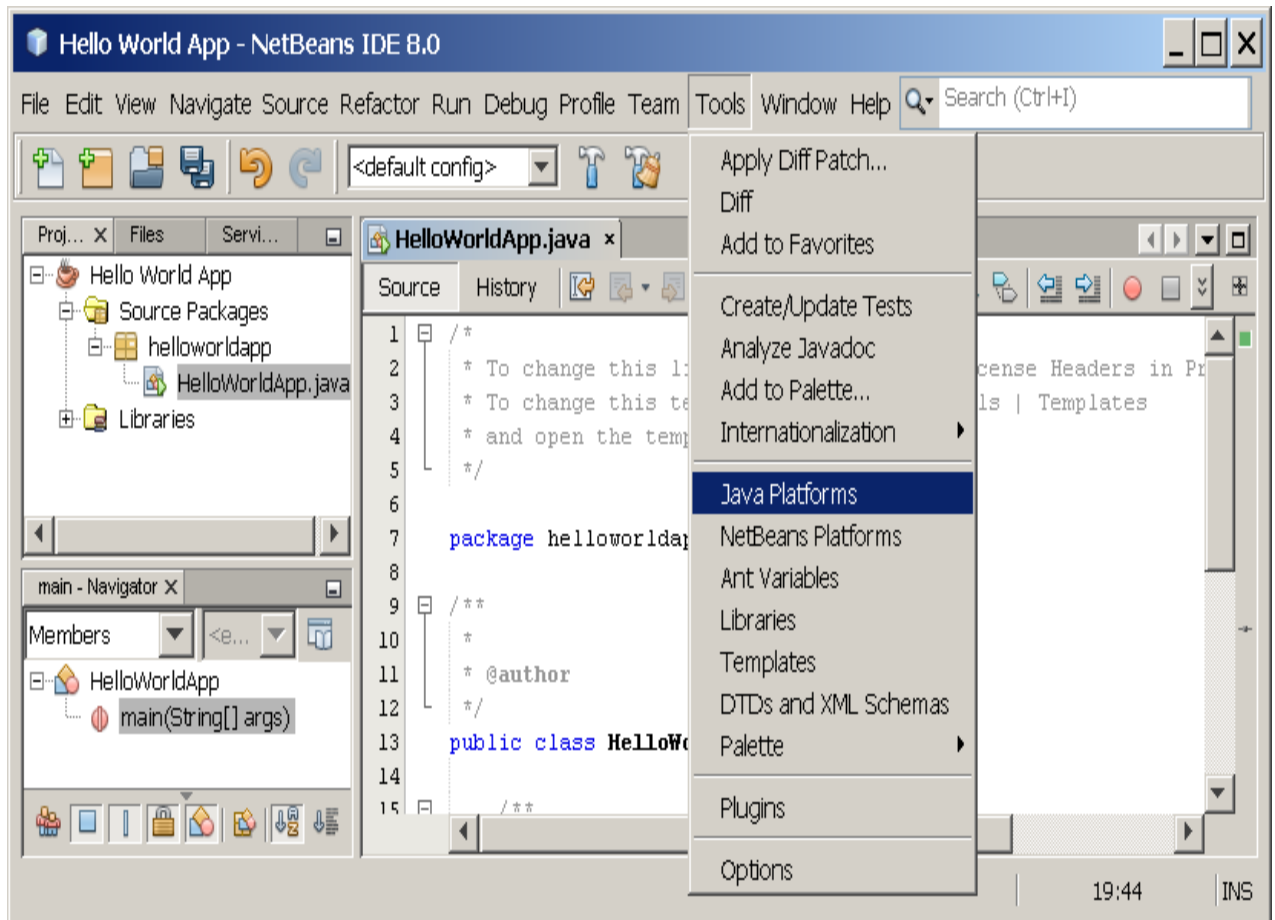
- The **Projects** window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The **Source Editor** window with a file called HelloWorldApp.java open.
- The **Navigator** window, which you can use to quickly navigate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



NetBeans IDE with the HelloWorldApp project open.

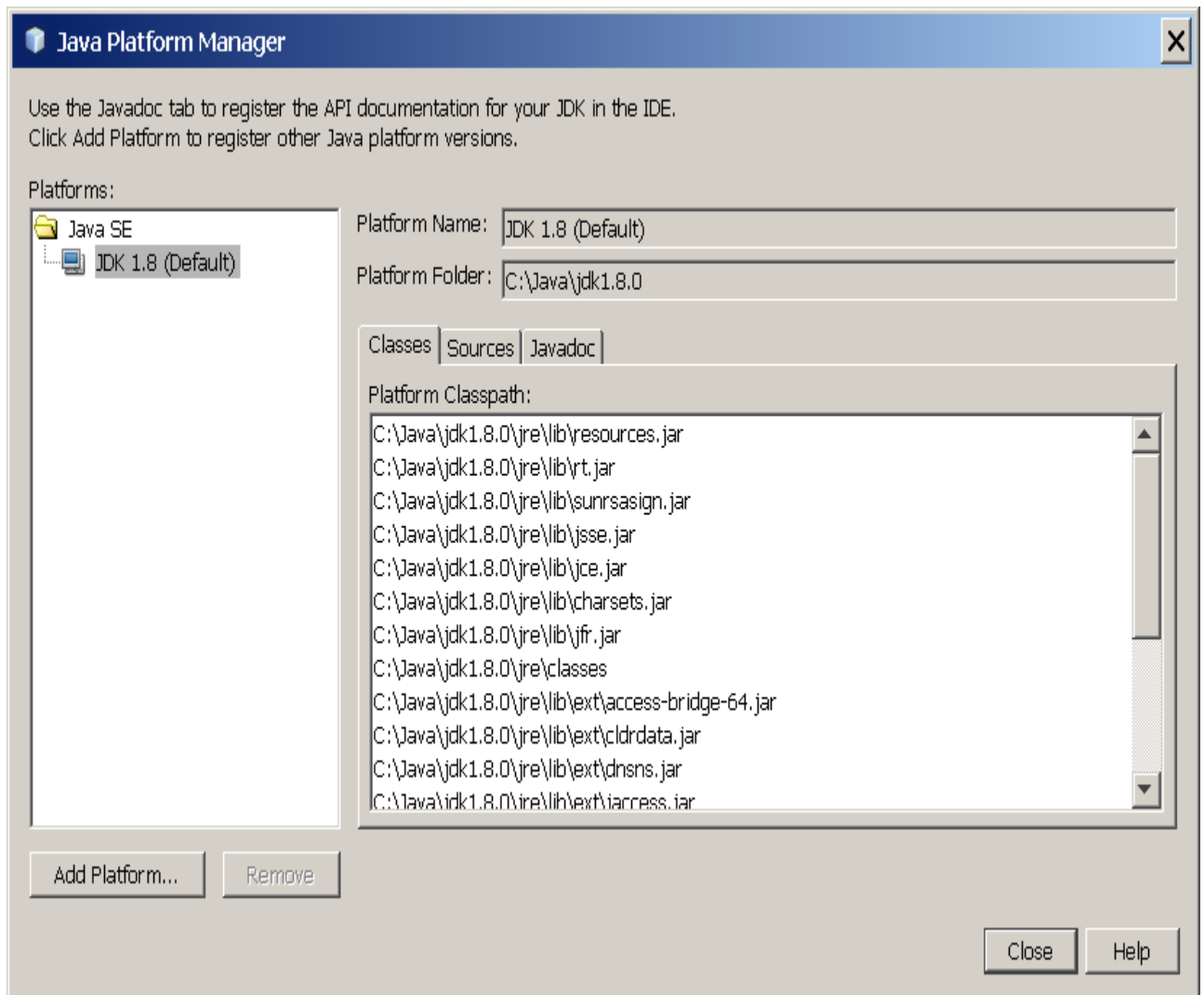
Add JDK 8 to the Platform List (if necessary)

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:



Selecting the Java Platform Manager from the Tools Menu

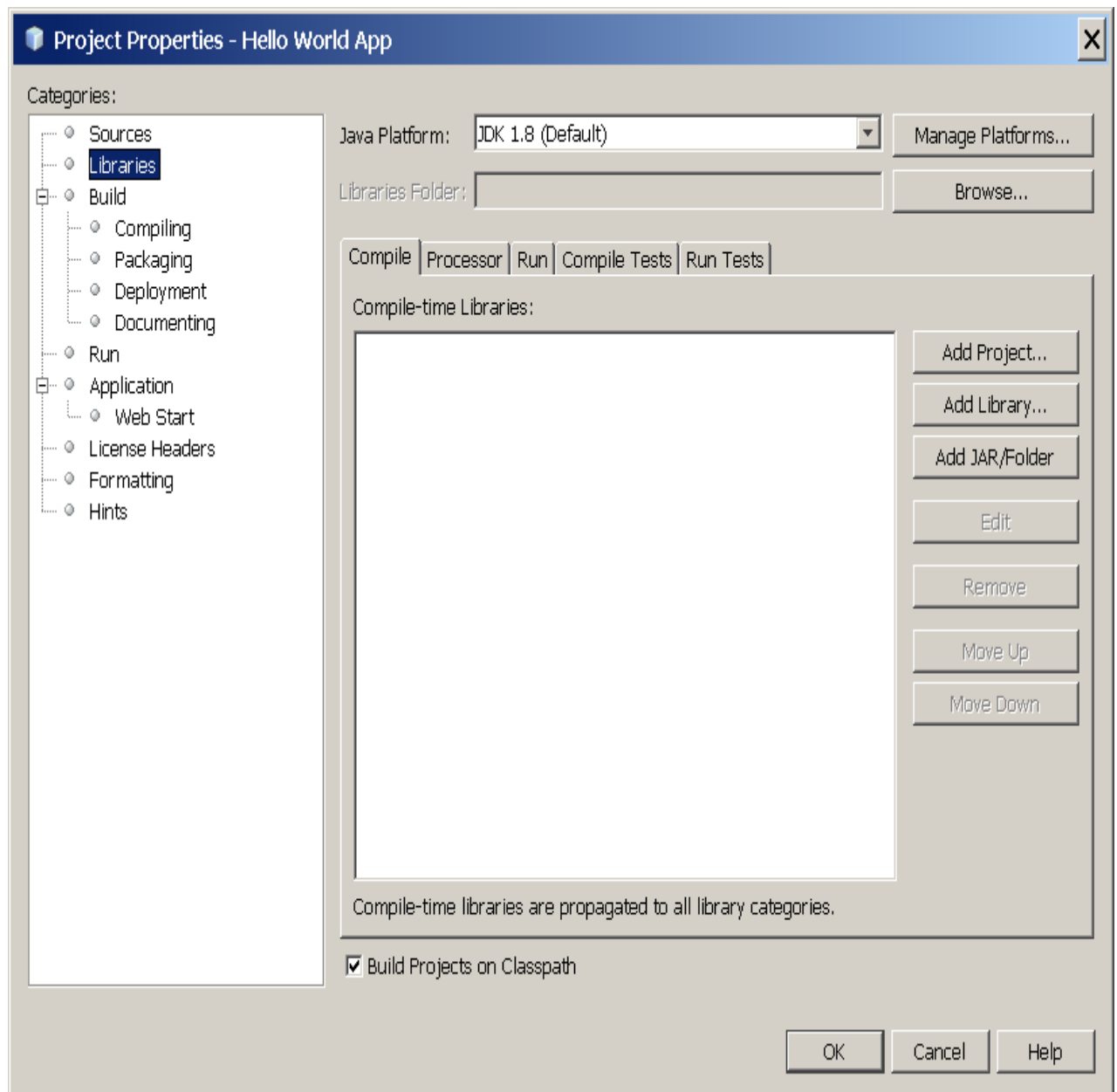
If you don't see JDK 8 (which might appear as 1.8 or 1.8.0) in the list of installed platforms, click **Add Platform**, navigate to your JDK 8 install directory, and click **Finish**. You should now see this newly added platform:



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:



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
```

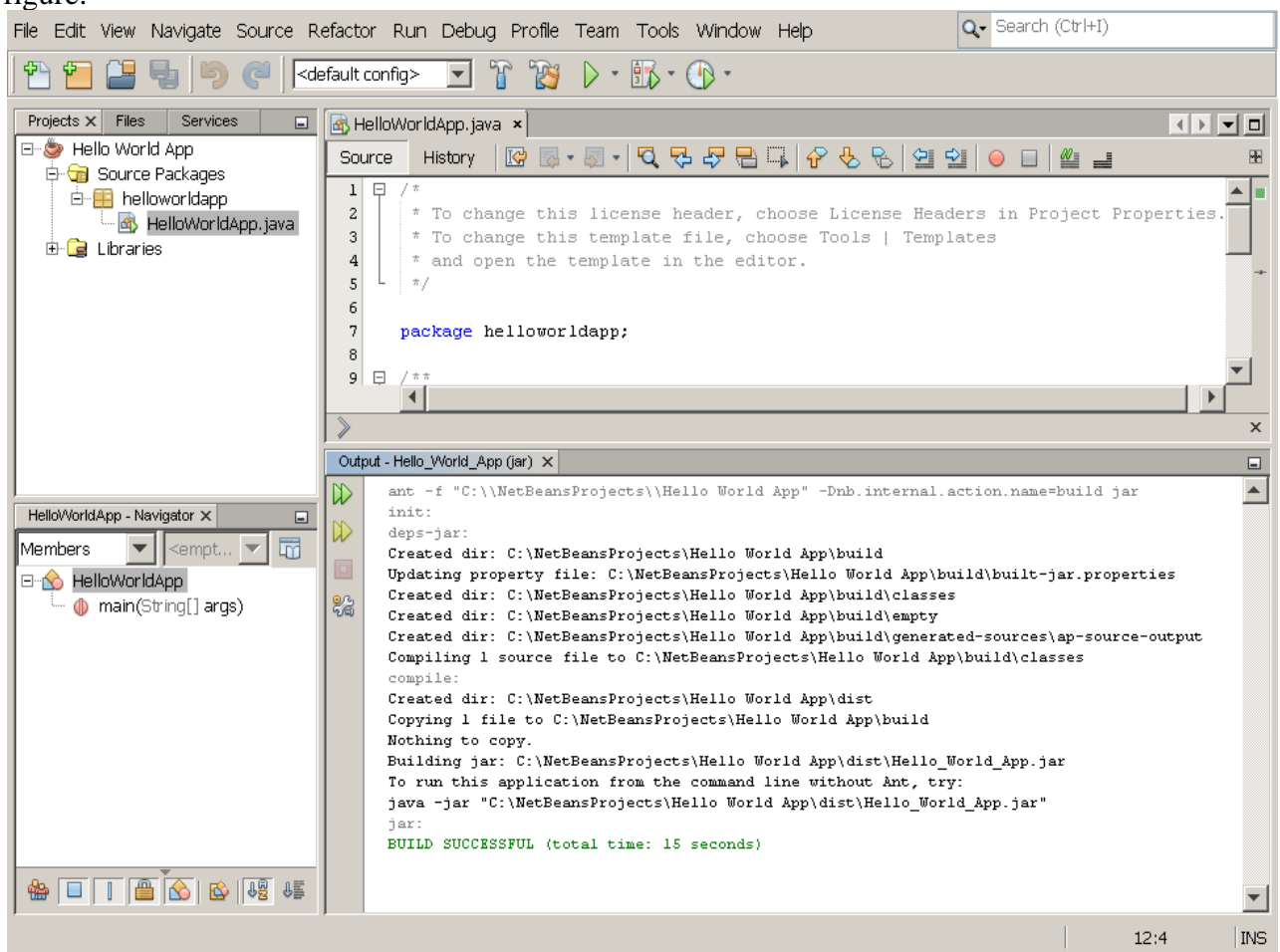


```
*/  
public static void main(String[] args) {  
    System.out.println("Hello World!"); // Display the string.  
}  
  
}
```

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:

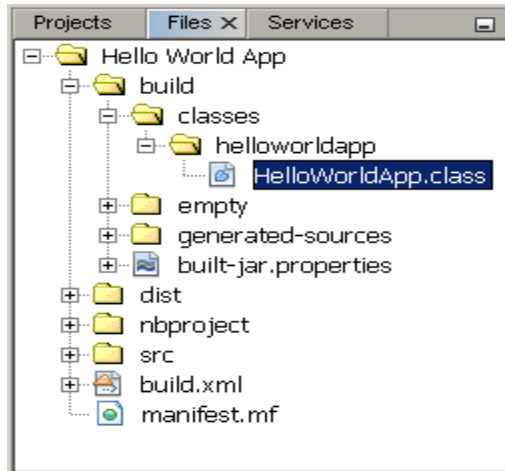


Output window showing results of building the HelloWorld project.

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.



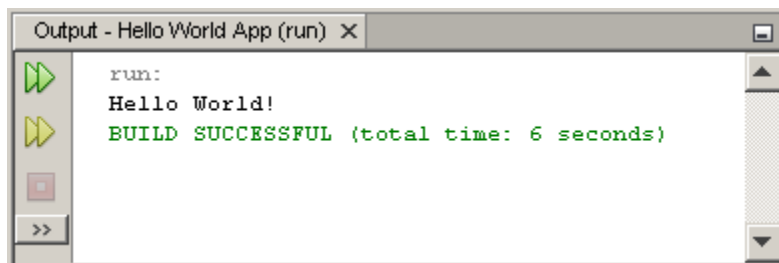
Files window, showing the generated .class file.

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.



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 discriminant $b^2 - 4ac$ is negative, display a message stating that there are no real solutions.

THEORY: The Quadratic $ax^2 + bx + c$, 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 $\text{root1} = -b + \sqrt{D}/2a$

STEP6: Calculate $\text{root2} = -b - \sqrt{D}/2a$

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 $\text{root1} = \text{root2} = -b/2*a$

STEP11: Print value of root1 and root2 the roots are real and equal.

STEP12: GOTO END

STEP13: Calculate $\text{RealPart} = -b/2a$

STEP14: Calculate $\text{imaginaryPart} = \sqrt{(-D)}/2a$

STEP15: Print value of $\text{root1} = \text{RealPart} + i \text{imaginaryPart}$

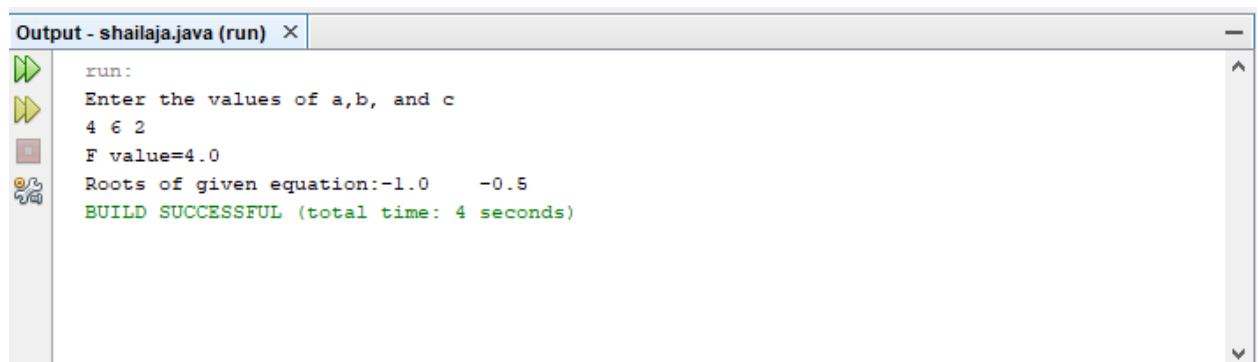
STEP16: Print value of $\text{root2} = \text{RealPart} - i \text{imaginaryPart}$ the roots are complex and different

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:

The screenshot shows a Java IDE window titled "Output - shailaja.java (run)". The output text is as follows:

```
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 NumbersGenerations

STEP1: START

STEP2: Enter the value of n.

STEP3: for i := 2 to n do

STEP4: Initialize p to 0

STEP5: for j := 2 to i / 2 do

STEP6: if j divides i

STEP7: proceed to the next j

STEP8: p =1,break;

STEP9: If (p=0) print i

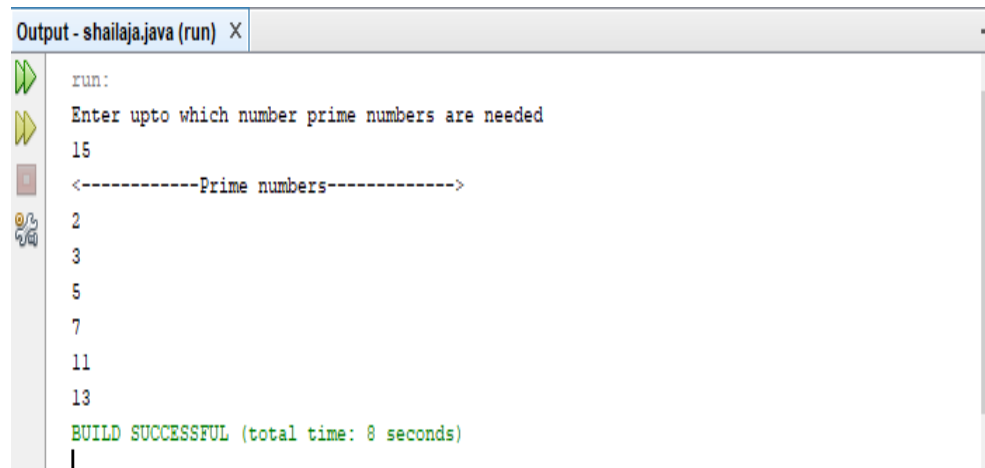
STEP10: proceed to the next i

STEP11: END

SOURCE CODE:

```
class Prime
{
    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:

```
Output - shailaja.java (run) X  
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. What are Object-oriented programming principles?**

Abstraction, encapsulation, inheritance, and polymorphism are four of the main principles of object-oriented programming.

2. Define class?

A Java class is the template from which objects are created. It is a blueprint for an object. A class is created using the keyword `class`. Classes can contain methods, which are actions that the class can perform, and fields, which are variables that contain data used by the class.

3. Define object?

A Java object is a member (also called an instance) of a Java class. Each object has an identity, a behavior and a state. The state of an object is stored in fields (variables), while methods (functions) display the object's behavior.

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 that checks whether a given string is a palindrome or not.

Ex: 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 not 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();

        int length = str.length();

        for ( int i = length - 1; i >= 0; i-- )
```

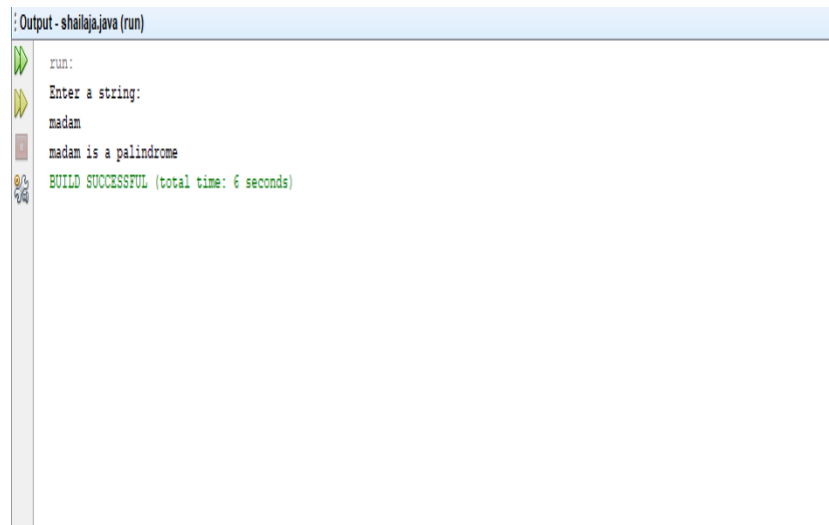


```
        rev = rev + str.charAt(i);

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

    }
}
```

OUTPUT



```
Output - shailaja.java (run)

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:

```
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(); //ignore next line character
```

```
String names[] = newString[N];

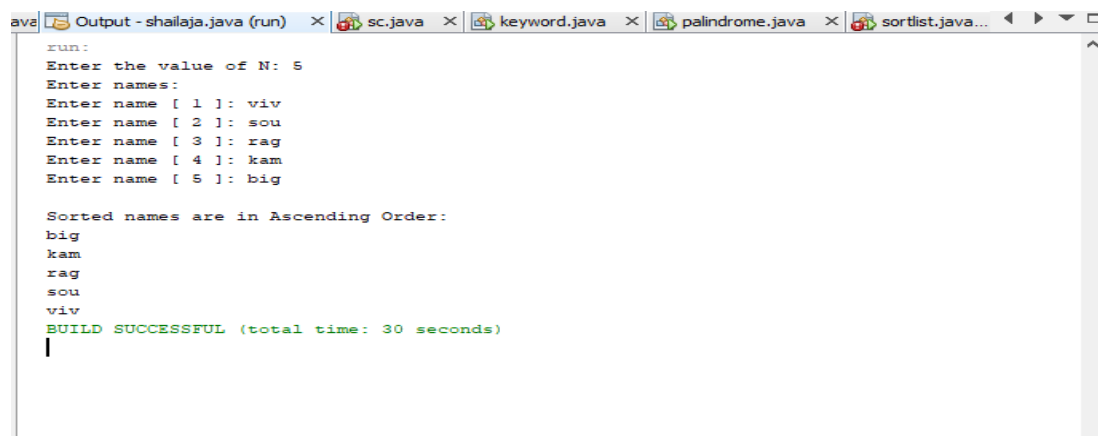
System.out.println("Enter names: ");
for(int i=0; i<N; i++)
{
    System.out.print("Enter name [ " + (i+1) + " ]: ");
    names[i] = SC.nextLine();
}

//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)
```

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. What is purpose of indexOf()?

The **indexOf()** method returns the position of the first occurrence of specified character(s) in a string. Tip: Use the **lastIndexOf** method to return the position of the last occurrence of specified character(s) in a 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 is the procedure to reverse the given string.

```
StringBufferd = new StringBuffer("welcome ");  
  
d.reverse();
```

5. Name the package which support the substrings.

The package which support the substrings is **java.lang**

WEEK 3

(A)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'

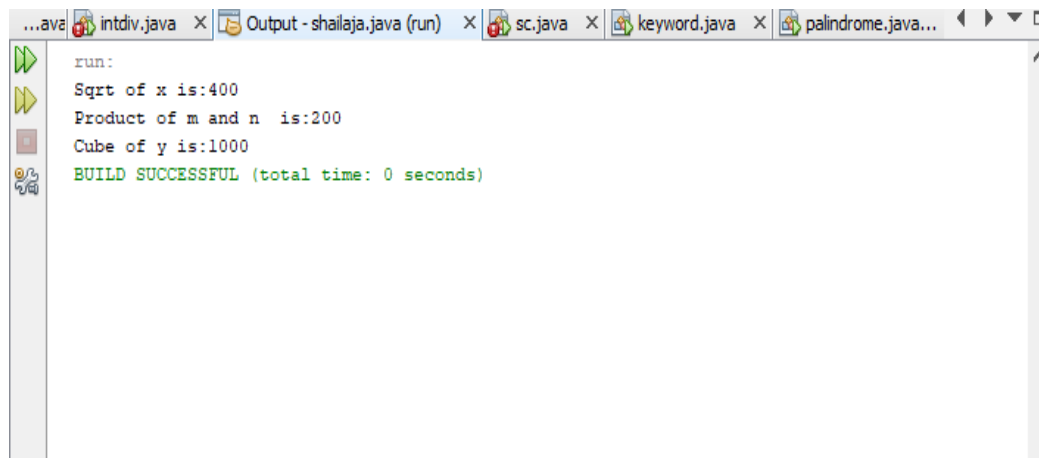
STEP 6: Using object obj call the display () with corresponding parameter values

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

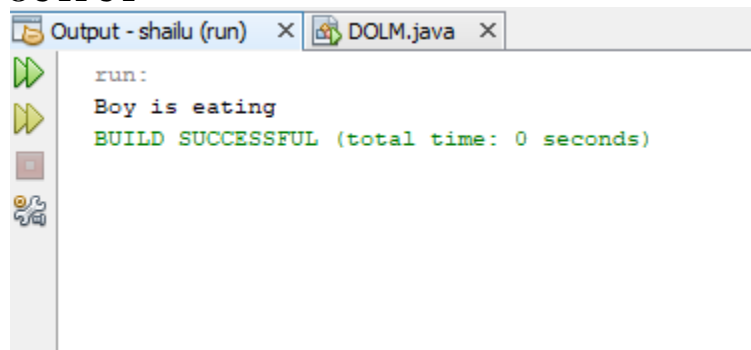
'Boy is eating' //which overrides the parent class

STEP9: STOP

SOURCE CODE

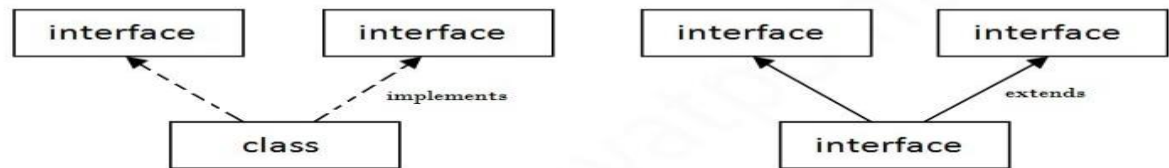
```
class Human{  
    //Overridden method  
    public void eat()  
    {  
        System.out.println("Human is eating");  
    }  
}  
class Boy extends Human{  
    //Overriding method  
    public void eat(){  
        System.out.println("Boy is eating");  
    }  
    public static void main(String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

OUTPUT



(B)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");}

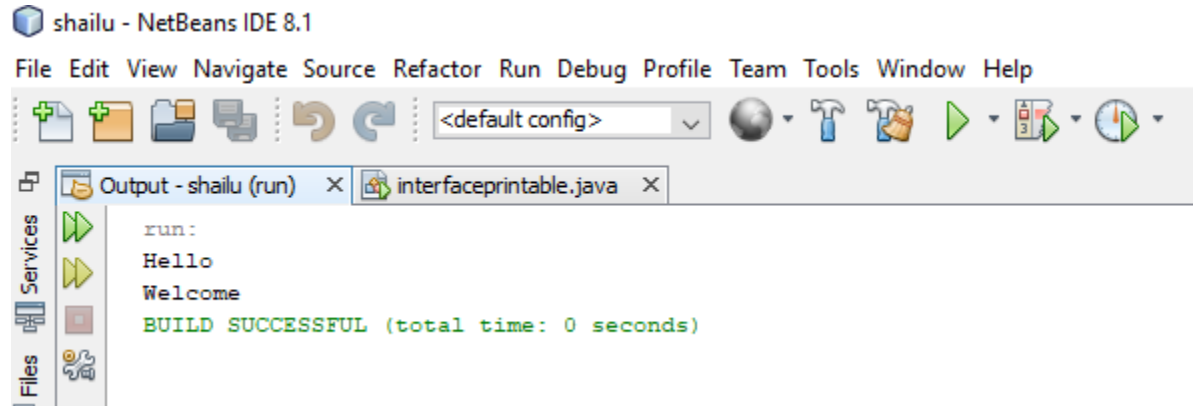
    public static void main(String args[]){
        multiple_inheritance obj = new multiple_inheritance ();
        obj.print();
    }
}

```



```
    obj.show();    }  
}
```

OUTPUT



VIVA VOCE:

1. Define method overloading?

Method overloading in java is a feature that allows a class to have more than one method with the same name, but with different parameters. Java supports method overloading through two mechanisms: By changing the number of parameters.

2. Define method overriding?

In Java, method overriding occurs when a subclass (child class) has the same method as the parent class. In other words, method overriding occurs when a subclass provides a particular implementation of a method declared by one of its parent classes.

3. What is interface?

An interface in Java is a **blueprint of a class**. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction.

4. What is the difference between 'extends' and 'implements' key words?

extends keyword is used to inherit a class or interface, while implements keyword is used to implement the interfaces. A class can extend only one class but can implement any number of interfaces. A subclass that extends a superclass may override some of the methods from superclass.

5. How to achieve multiple inheritance in Java.

In Java, we can achieve multiple inheritance through the concept of interface. An interface is like a class that has variables and methods, however, unlike a class, the methods in an interface are abstract by default.

WEEK 4

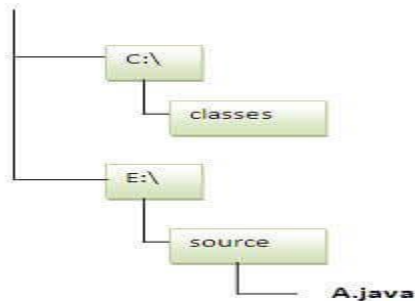
(A)AIM: To write a program to demonstrate packages.

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:

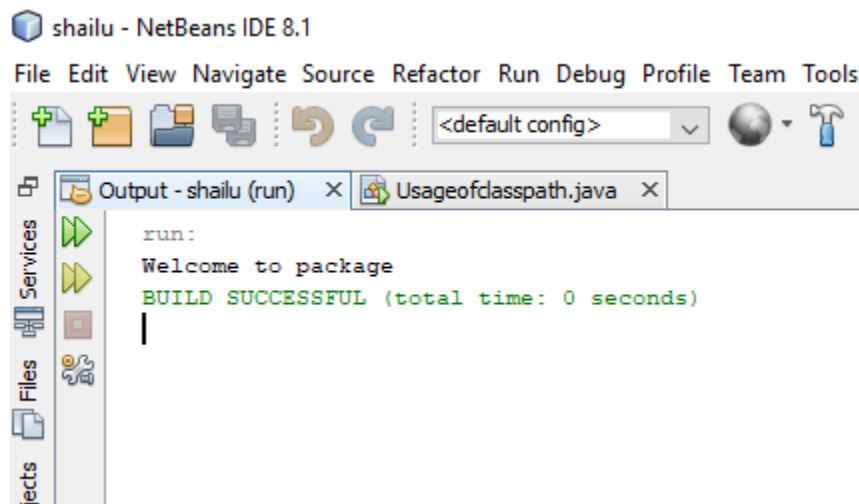
To run this program from e:\source directory, you need to set classpath of the directory where the class file resides.

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:

(B)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
{

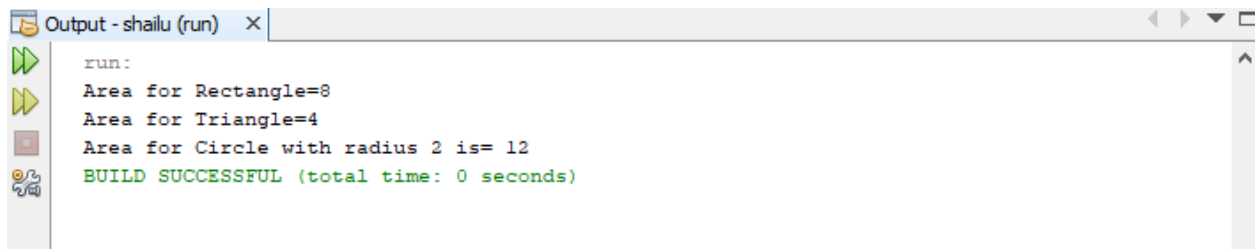
    void printArea()
    {
        int area=(int)(3.14*a*a);
        System.out.println("Area for Circle with radius "+a+" is= "+area);

    }
}

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:

The screenshot shows an IDE output window titled "Output - shailu (run)". On the left, there are icons for running (green play button), stepping through (yellow play button), and debugging (red bug icon). The output text is as follows:

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

VIVA - VOCE**1. Define java Inheritance.**

Inheritance in Java is a concept that acquires the properties from one class to other classes; for example, the relationship between father and son. Inheritance in Java is a process of acquiring all the behaviours of a parent object.

2. Define java package

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.

3. What is the purpose of CLASSPATH?

Classpath is a parameter in the Java Virtual Machine or the Java compiler that specifies the location of user-defined classes and packages. The parameter may be set either on the command-line, or through an environment variable.

4. What is an abstract class?

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class). Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

5. What is the purpose of 'import' keyword?

The use of keyword 'import' in Java programming is used to import the built-in and user-defined packages, class or interface in Java programming.

WEEK-5**(A) AIM**

To write a java program to demonstrate exception handling mechanism.

THEORY**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

- STEP 1 : START
- STEP 2 : Create a class with name 'MultipleCatchBlocks' and write main()
- STEP 3 : Start try block
- STEP 4 : Read two parameters
- STEP 5 : Evaluate Division of two parameters
- STEP 6 : End try block
- STEP 7 : Write catch block to handle ArithmeticException
- STEP 8 : Write catch block to handle ArrayIndexOutOfBoundsException
- STEP 9 : Write catch block to handle NumberFormatException
- STEP 10 : Write Finally block
- STEP 11 : Print message 'Exception Handling completed..'
- STEP 12 : STOP

SOURCE CODE

```
public class MultipleCatchBlock1 {  
  
    public static void main(String[] args) {  
  
        try{  
            int a[]=new int[5];  
            a[5]=30/0;
```

```
    }  
    catch(ArithmeticException e)  
    {  
        System.out.println("Arithmetic Exception occurs");  
    }  
    catch(ArrayIndexOutOfBoundsException e)  
    {  
        System.out.println("ArrayIndexOutOfBoundsException occurs");  
    }  
    catch(Exception e)  
    {  
        System.out.println("Parent Exception occurs");  
    }  
    System.out.println("rest of the code");  
}  
}
```

OUTPUT

```
Arithmetic Exception occurs  
rest of the code
```

(B) 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);
    }
}

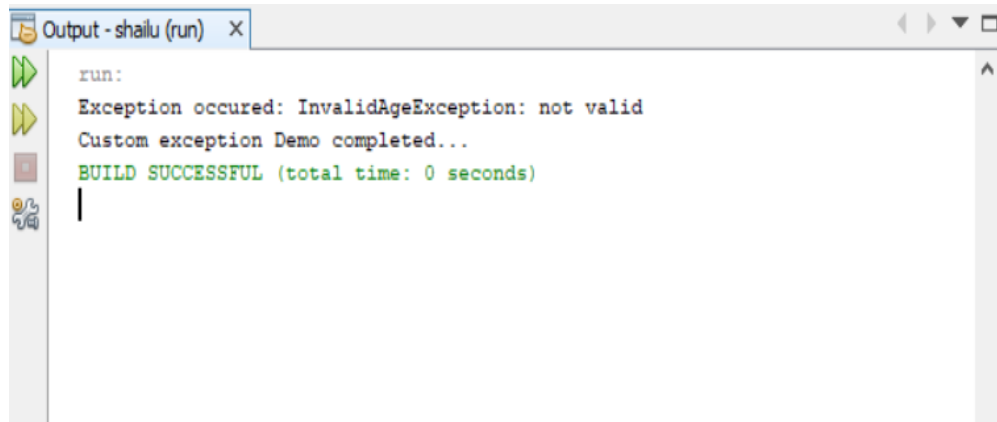
class TestCustomException1 {

    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 occurred: "+m);}
    }
}
```

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

OUTPUT



VIVA - VOCE:

1. What is an Exception?

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.

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. Define Try and catch Blocks?

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 **acatch 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.

4. What is the purpose of 'finally' block?

The finally block in java is used to put important codes such as clean up code e.g. closing the file or closing the connection. The finally block executes whether exception rise or not and whether exception handled or not. A finally contains all the crucial statements regardless of the exception occurs or not.

5. What is the Difference between Throw and throws?

Throw	Throws
Throw keyword is used in the method body to throw an exception, while throws is used in method signature to declare the exceptions that can occur in the statements present in the method.	Throws clause is used to declare an exception, which means it works similar to the try-catch block.

WEEK 6**(A). AIM**

Write a Java program that implements producer-consumer problem.

THEORY

Producer is producing some items, whereas there is one Consumer that is consuming the items produced by the Producer. The same memory buffer is shared by both producers and consumers which is of fixed-size.

The task of the Producer is to produce the item, put it into the memory buffer, and again start producing items. Where as the task of the Consumer is to consume the item from the memory buffer.

ALGORITHM

STEP 1 : START

STEP 2 : Create the Producer and Consumer classes and create as Threads.

STEP 3 : Create shared resource Q class which is having synchronized methods


STEP 4 : Start the Threads.

STEP 5 : Use notify() and wait() alternatively access the shared data.

STEP 6 : Producer produce the data.

STEP 7 : Consumer consume the data.

STEP 8 : STOP

SOURCE CODE

```
class Q {  
    int n;  
    boolean valueSet = false;  
    synchronized int get() {  
        while(!valueSet)  
            try {
```

```
Thread.sleep(2000);
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
Thread.sleep(2000);
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
notify();
}
}
class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
```

```
public void run() {  
    int i = 0;  
    while(true) {  
        q.put(i++);  
    }  
}  
  
class Consumer implements Runnable {  
    Q q;  
    Consumer(Q q) {  
        this.q = q;  
        new Thread(this, "Consumer").start();  
    }  
    public void run() {  
        while(true) {  
            q.get();  
        }  
    }  
}  
  
class PC {  
    public static void main(String args[]) {  
        Q q = new Q();  
        new Producer(q);  
        new Consumer(q);  
        System.out.println("Press Control-C to stop.");  
    }  
}  
  
OUTPUT:  
Put: 0  
Got: 0
```


Put: 1

Got: 1

Put: 2

Got: 2

Put: 3

Got: 3

Put: 4

Got: 4

.

.

.

.

etc

VIVA VOCE

1. What is inter thread communication?

Inter-thread communication in Java is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed.

2. What is the use of wait()?

In Java, the wait() method is used to pause the execution of a thread until another thread signals that it can resume. When a thread calls wait() on an object, it releases the lock on the object and waits until another thread calls notify() or notifyAll() on the same object.

3. What is the use of notify()?

The notify() method is defined in the Object class, which is Java's top-level class. It's used to wake up only one thread that's waiting for an object, and that thread then begins execution. The thread class notify() method is used to wake up a single thread.

4. What is the purpose of notifyAll()?

notifyAll() wakes up all threads that are waiting on this object's monitor. A thread waits on an object's monitor by calling one of the wait methods. The awakened threads will not be able to proceed until the current thread relinquishes the lock on this object.

5. What is Synchronization?

Synchronization in java is the capability to control the access of multiple threads to any shared resource. In the Multithreading concept, multiple threads try to access the shared resources at a time to produce inconsistent results. The synchronization is necessary for reliable communication between threads.

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);
}

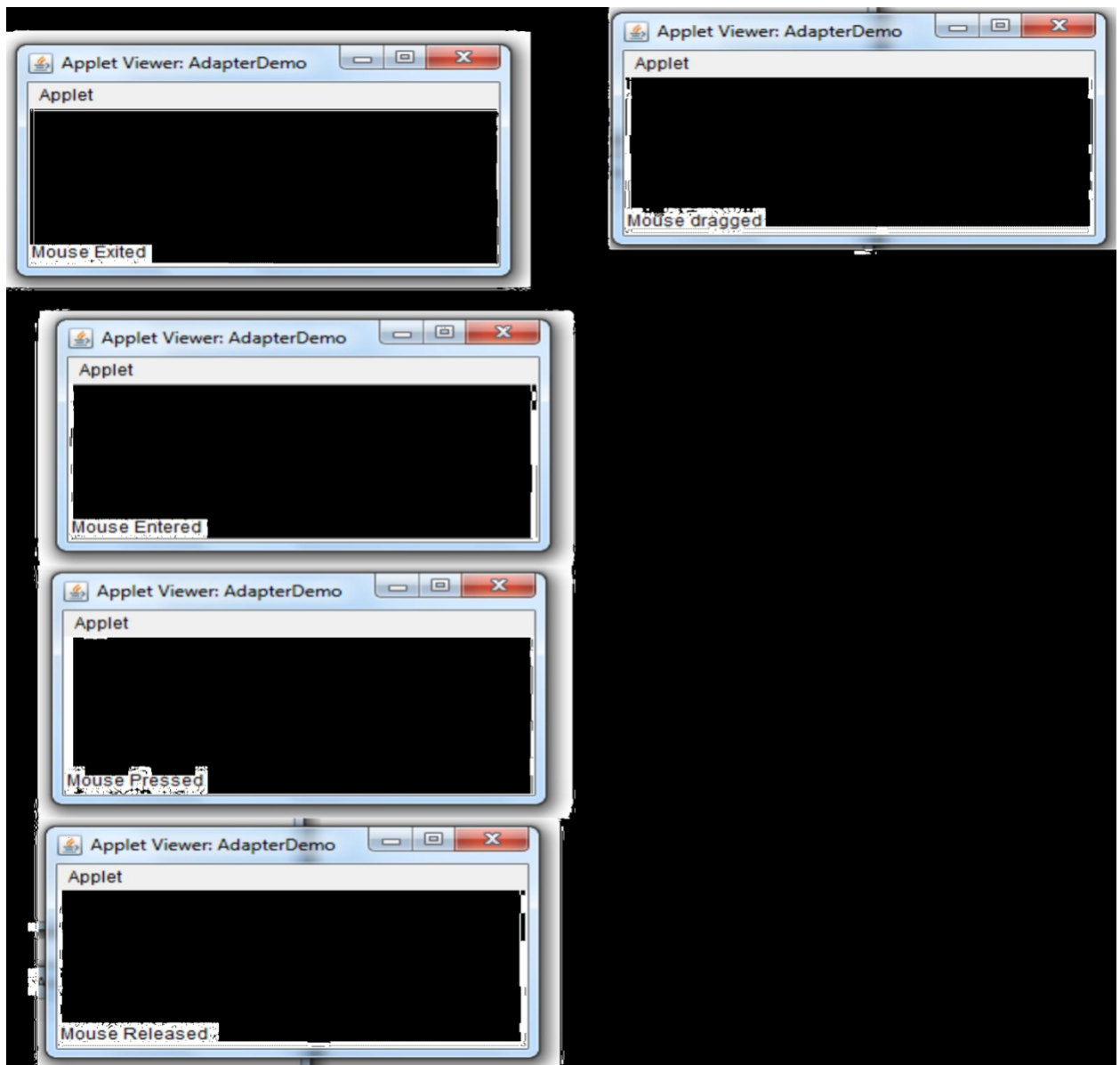
public void mouseClicked(MouseEvent me)
{
    mouseX = 0;
    mouseY = 10;
    msg = "Mouse clicked.";
    repaint();
}

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:

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.

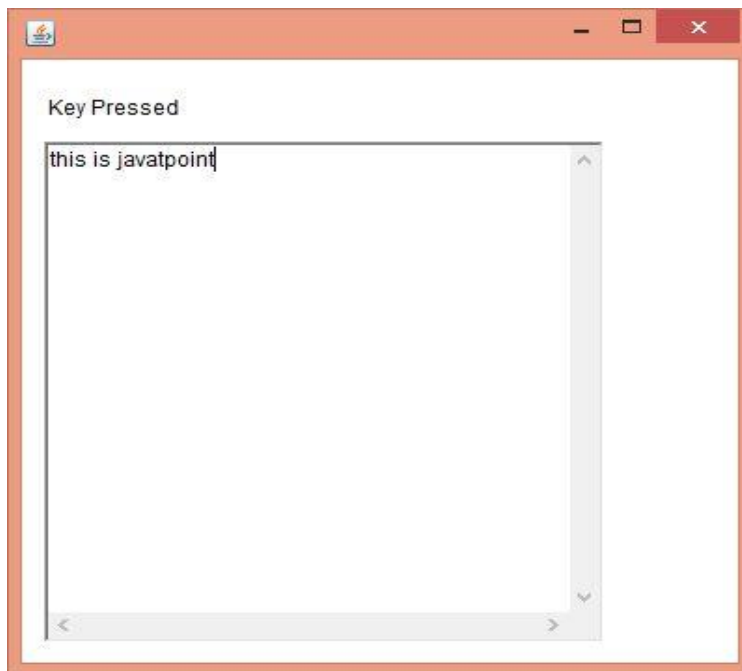
STEP9: END

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:

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:



(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

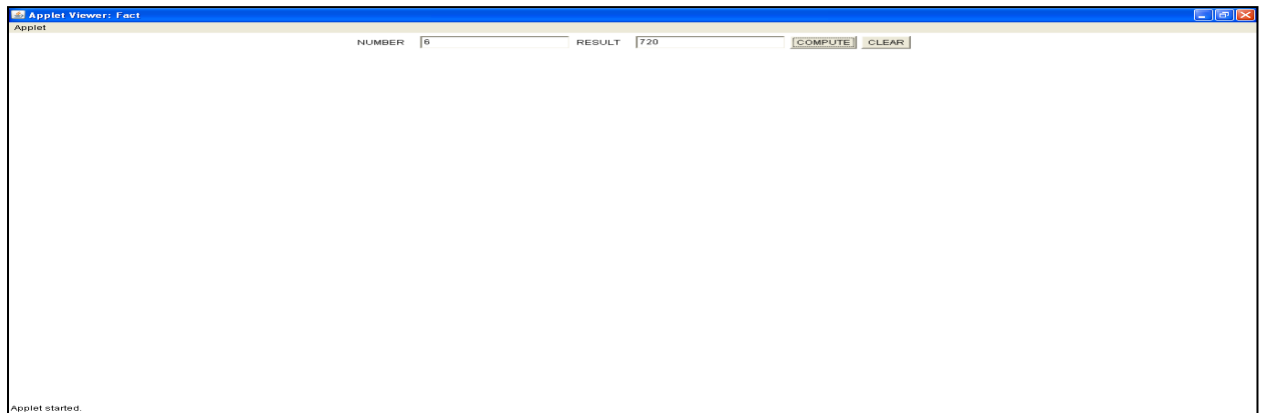
SOURCE CODE:

```
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. Its return type is void. getText() method is used to get text from the TextField specified through object. Its 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 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().

4. 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.

5. 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

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());
```

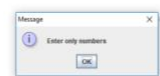
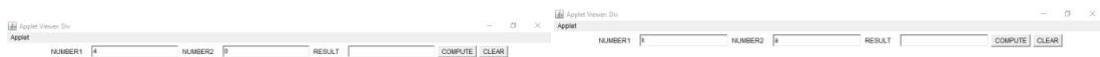
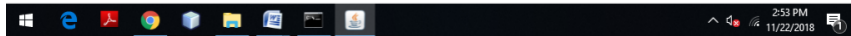
```
int b=Integer.parseInt(tf2.getText());
int c=a/b;
tf3.setText(""+c);
}
catch(NumberFormatException ex)
{
tf3.setText(" ");
JFrame f=new JFrame();
JOptionPane.showMessageDialog(f,"Enter only numbers");
repaint();
}
catch(ArithmeticException ex)
{
tf3.setText(" ");
JFrame f=new JFrame();
JOptionPane.showMessageDialog(f,"Enter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

public void paint(Graphics g)
{
```

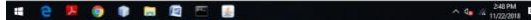
```
g.drawString(msg,30,70);  
}  
}
```

OUTPUT:

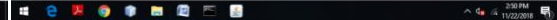
Applet started.



Applet started.



Applet started.



VIVA - VOCE:**1. What is the purpose of ArithmeticException?**

An arithmetic exception in java is a Runtime exception present in the java. lang package. JVM throws Arithmetic Exception when a wrong mathematical expression occurs in a java program.

2. What is the purpose of NumberFormatException?

The NumberFormatException occurs when an attempt is made to convert a string with improper format into a numeric value. That means, when it is not possible to convert a string in any numeric type (float, int, etc), this exception is thrown.

3. What is the use of JOptionPane?

JOptionPane is generally used to access the standard dialog boxes like confirm dialog box, message dialog box, and input dialog box. These dialog windows are used to present information to the user or to solicit input from them.

4. What is dialog box?

Dialog boxes are graphical components that are usually used to display errors or give some other information to the user. They are part of the three top-level containers that each Java graphical user interface (GUI) application must have as a root. Dialogs are created as part of a frame.

5. What is the use of ActionListener?

Java ActionListener is an interface in java. awt. event package. It is an type of class in Java that receives a notification whenever any action is performed in the application.

WEEK 10

(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 be 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

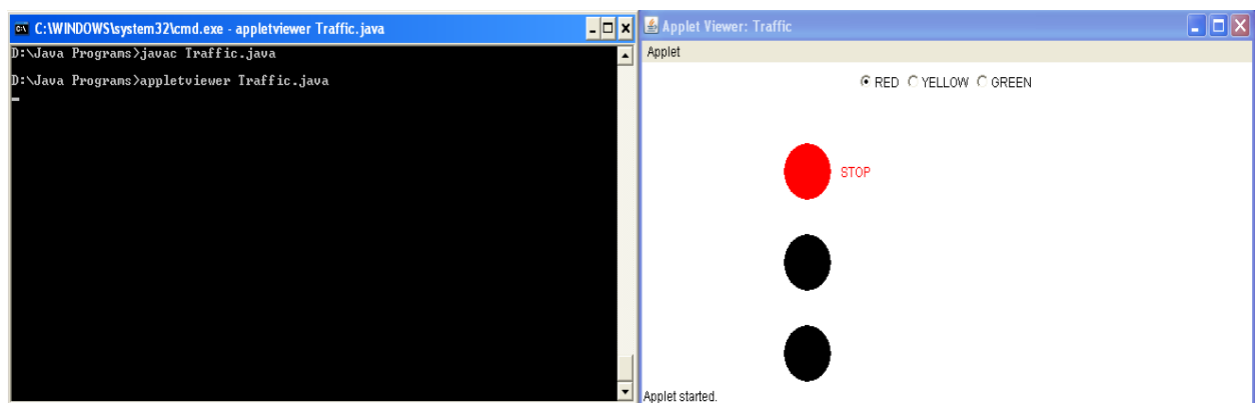
SOURCE CODE:

```
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:



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 itemStateChanged()?

The itemStateChanged() method is invoked automatically whenever you click or unclick on the registered checkbox component.

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. The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

WEEK 11

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 within 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.

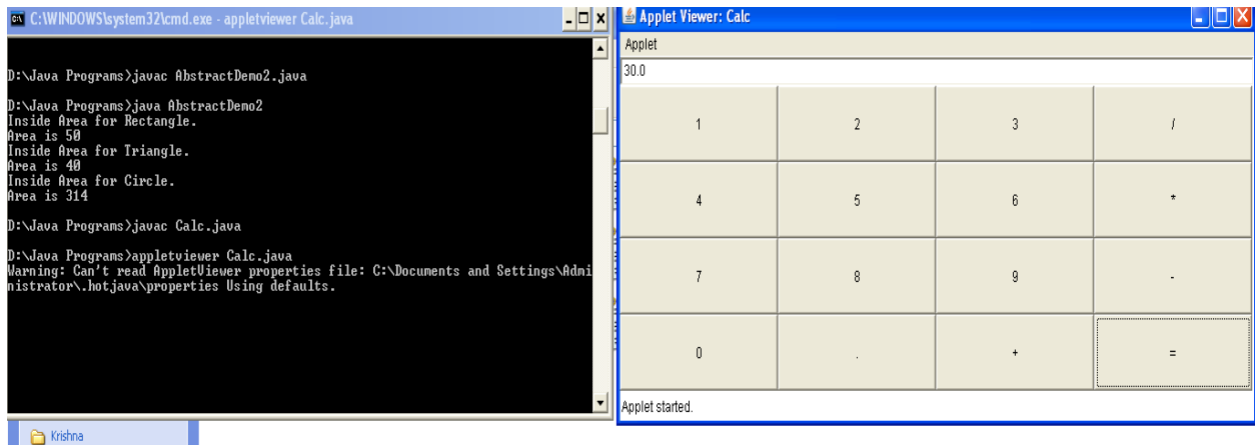
STEP8: END

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  
    {  
        calcul(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:

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 getActionCommand()?

getActionCommand() Returns the command string associated with this action. This string allows a "modal" component to specify one of several commands, depending on its state. For example, a single button might toggle between "show details" and "hide details".

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.

WEEK 12

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

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.

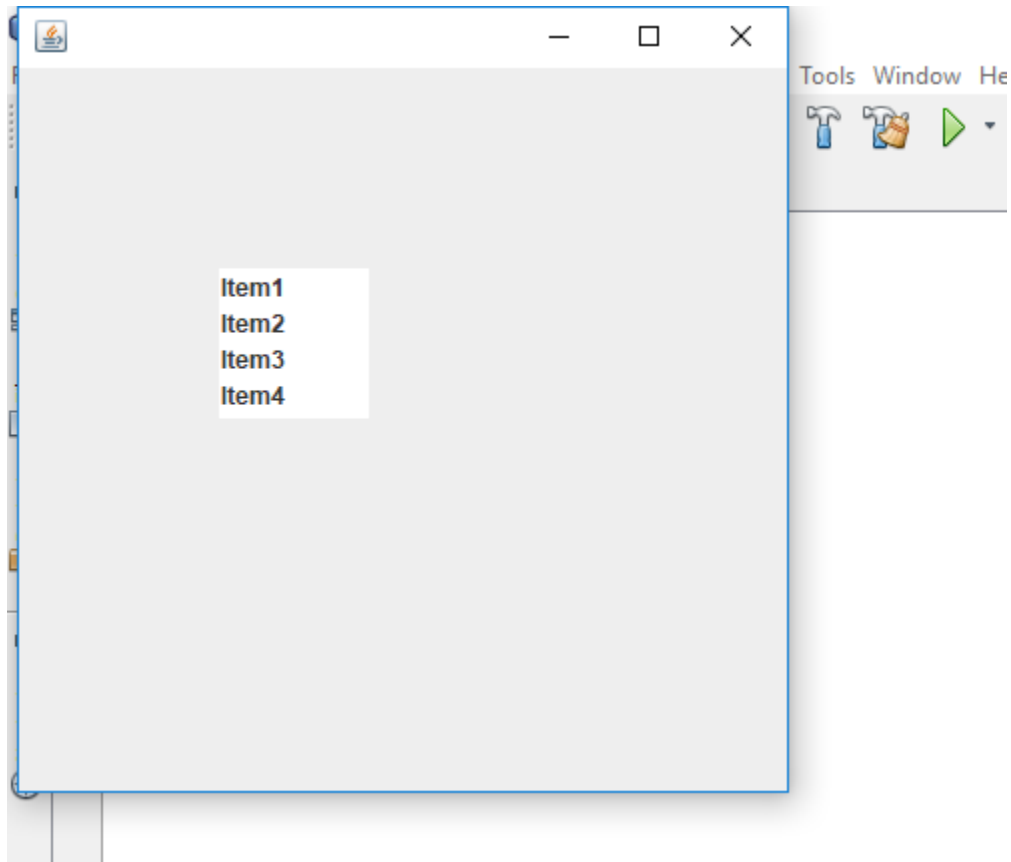
STEP6: END

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

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

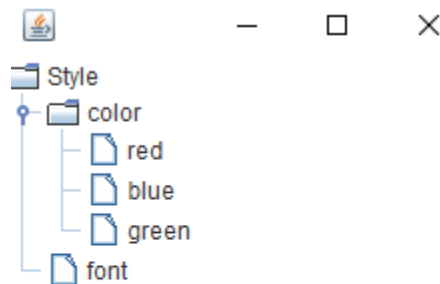
```
import javax.swing.*;

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**

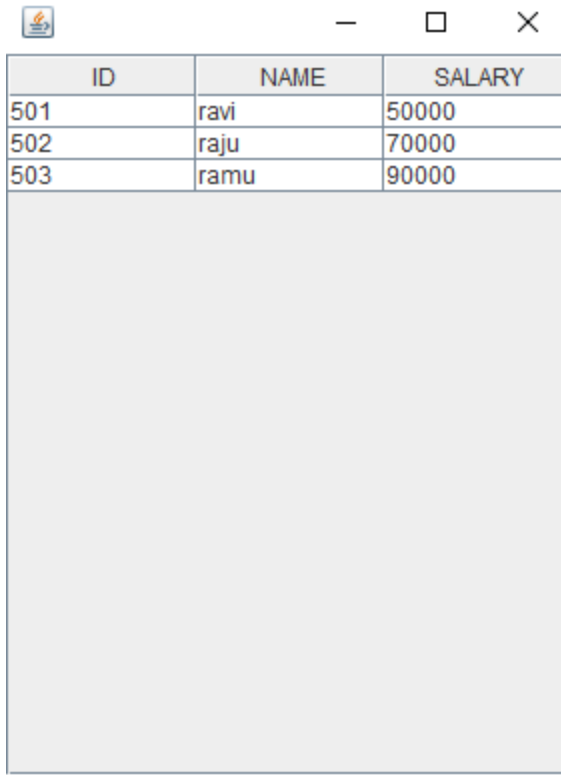
STEP1: START
STEP2: Create a Panel and set panel Layout.
STEP3: Add JTable to the Panel.
STEP4: Set the JTable rows and columns .
STEP5: Load the data into the JTable
STEP6: END

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



The image shows a Java Swing window with a title bar containing a small icon, a minus sign, a maximize button, and a close button. The window contains a table with three columns: ID, NAME, and SALARY. The table has three rows of data. Below the table is a large, empty rectangular area.

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 JTree class?

The JTree class is used to display the tree structured data or hierarchical data. JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree.

4. What is the purpose of JTable class?

The JTable class is a part of Java Swing Package and is generally used to display or edit two-dimensional data that is having both rows and columns. It is similar to a spreadsheet. This arranges data in a tabular form.

5. Differentiate between applet and swing.

Applet uses AWT Layouts like flowlayout. Swing have some Thread rules. Applet doesn't have any rule. To execute Swing no need any browser By which we can create stand alone application But Here we have to add container and maintain all action control with in frame container.

WEEK 13

AIM: To develop Swing application which uses JTabbedPane and JScrollPane

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

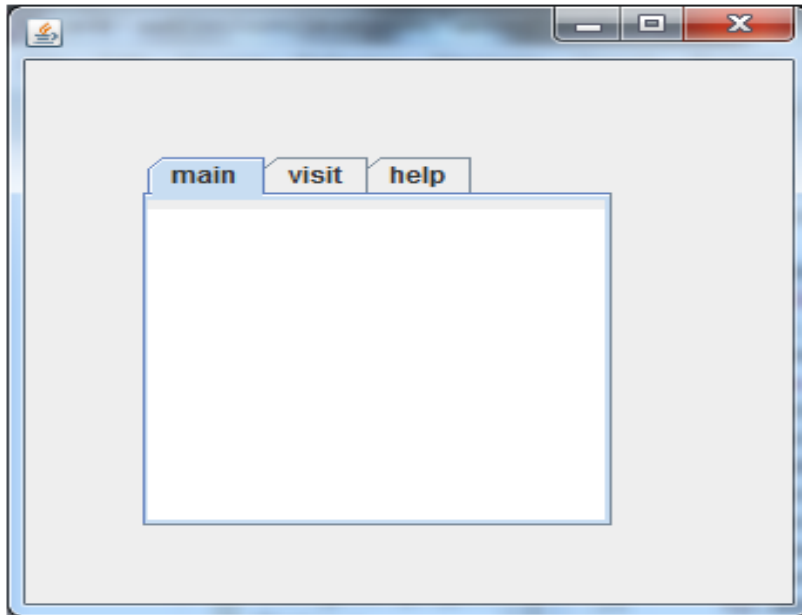
SOURCE CODE: JTabbedPane

```
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();  
} }
```

OUTPUT: JTabbedPane

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 java.awt.*;

import javax.swing.*;

public class ScrollEx
{
    JFrame f;

    ScrollEx()
    {
        f=new JFrame();

        JPanel jp=new JPanel() ;

        jp.setLayout(new GridLayout(20,20)) ;

        for(int i=0;i<20;i++)
        for(int j=0;j<20;j++)
        {
            jp.add(new JButton("Button "+j));
        }

        //int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
        //int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;

        //JScrollPane js=new JScrollPane(jp,v,h);

        JScrollPane js=new JScrollPane(jp);

        f.add(js,BorderLayout.CENTER);

        f.setSize(300,400);

        f.setVisible(true);
    }
}
```



```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
public static void main(String[] args)  
{  
    new ScrollEx();  
}
```

OUTPUT: JScrollPane



VIVA VOCE:

1. What is JFC?

JFC is short for Java Foundation Classes, which encompass a group of features for building graphical user interfaces (GUIs) and adding rich graphics functionality and interactivity to Java applications. It is defined as containing the features shown in the table below. Feature. Description. Swing GUI Components.

2. What is Light weight component?

A lightweight component has no native screen resource of its own, so it is "lighter." A lightweight component relies on the screen resource from an ancestor in the containment hierarchy, possibly the underlying Frame object. Components from the javax.

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.

5. What are the differences between a JScrollBar and a JScrollPane in Java?

A JScrollBar is a component and it doesn't handle its own events whereas a JScrollPane is a Container and it handles its own events and performs its own scrolling.