www.hbpatel.in

JAVA

Programming

Subject: OOP using Java

www.hbpatel.in

Subject Code: CT405-N
Subject Title: Object Oriented Programming Using Java
Detail Contents:
https://www.ldrp.ac.in/images/syllabus/BE-Computer-CBCS/CT405-N-Object%20Oriented%20Programming%20using%20Java.pdf

Kadi Sarva Vishwavidyalaya

Faculty of Engineering & Technology
Second Year Bachelor of Engineering (CE/IT) – Semester IV
(With effect from: Academic Year 2018-19)

Subject Code: CT405-N	**Subject Title:** Object Oriented Programming Using Java
Pre-requisite	-

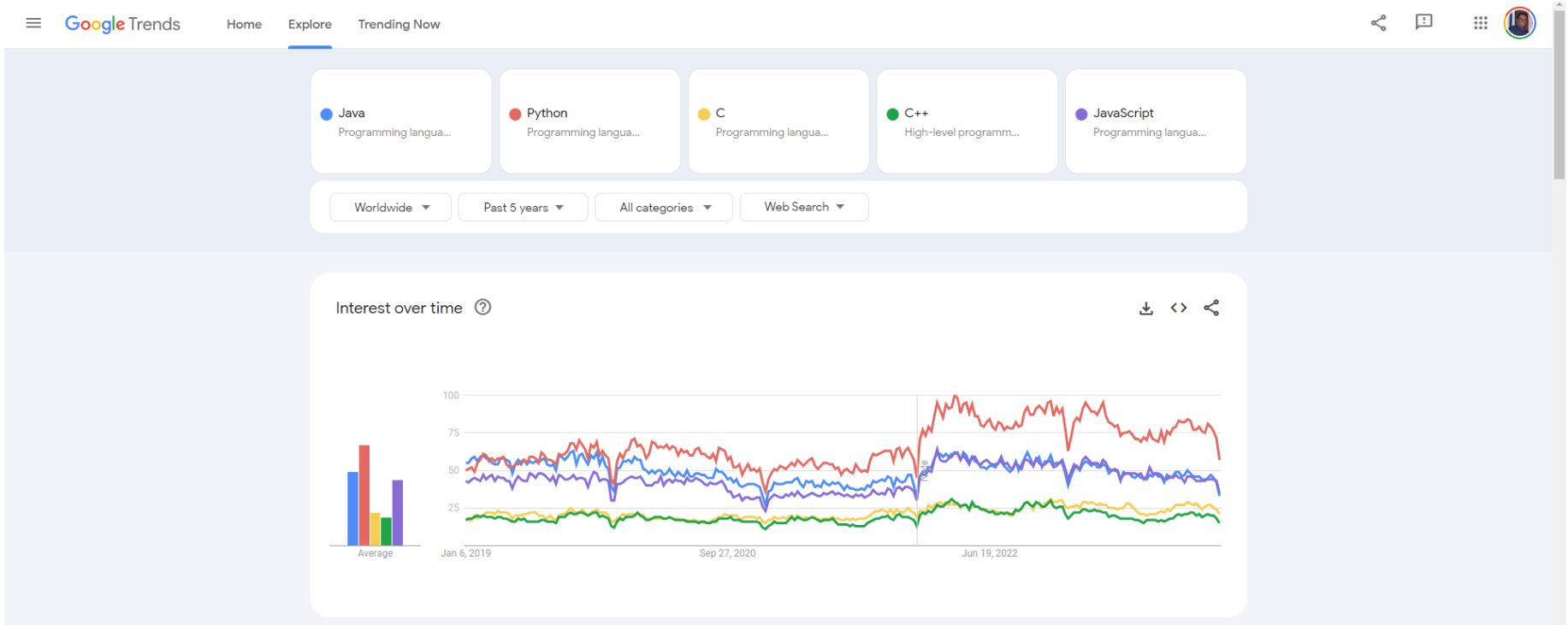
Teaching Scheme (Credits and Hours)

Teaching Scheme				Total Credit	Evaluation Scheme					
L	T	P	Total		Theory		Mid Sem Exam	CIA	Practical	Total
Hours	Hours	Hours	Hours		Hours	Marks	Marks	Marks	Marks	Marks
03	00	04	07	05	03	70	30	20	30	150

Why Java?

www.hbpatel.in

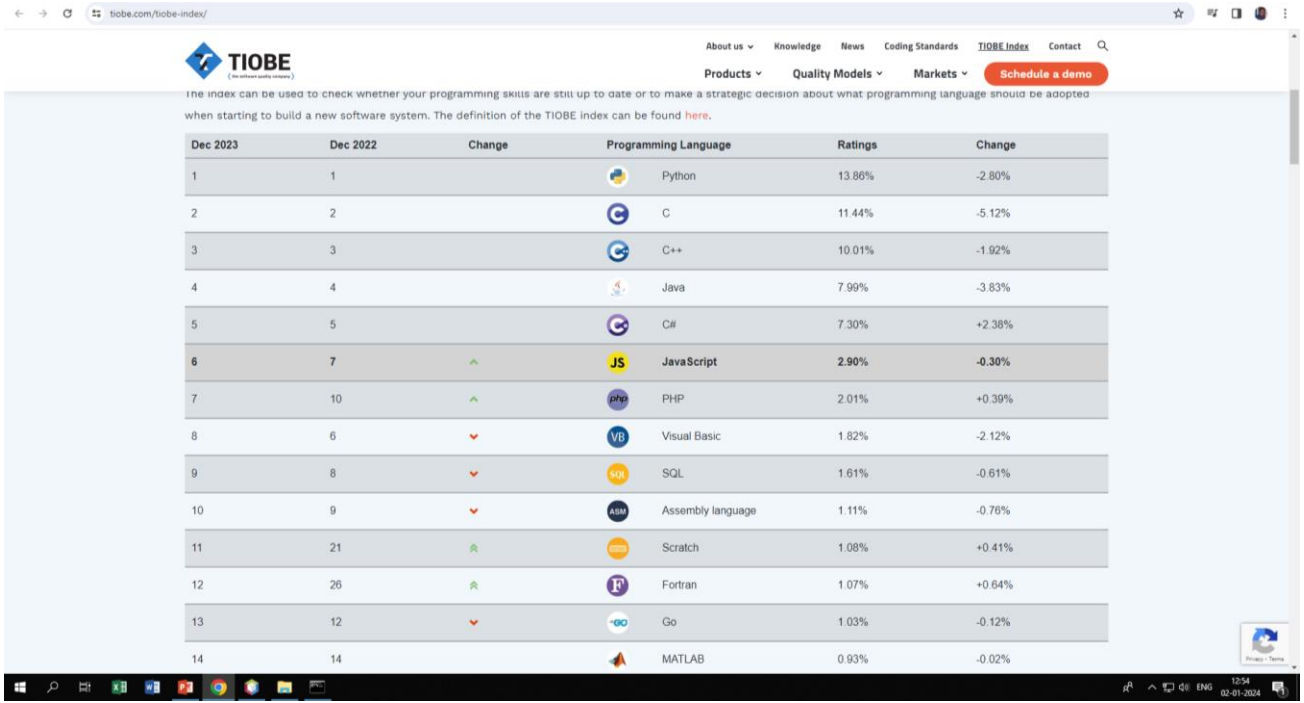
Java is the (second) most in-demand programming language



Why Java?

www.hbpatel.in

TIOBE Index for December 2023 (Source: https://www.tiobe.com/tiobe-index/)



tiobe.com/tiobe-index/

TIOBE

About us | Knowledge | News | Coding Standards | TIOBE Index | Contact

Products | Quality Models | Markets | Schedule a demo

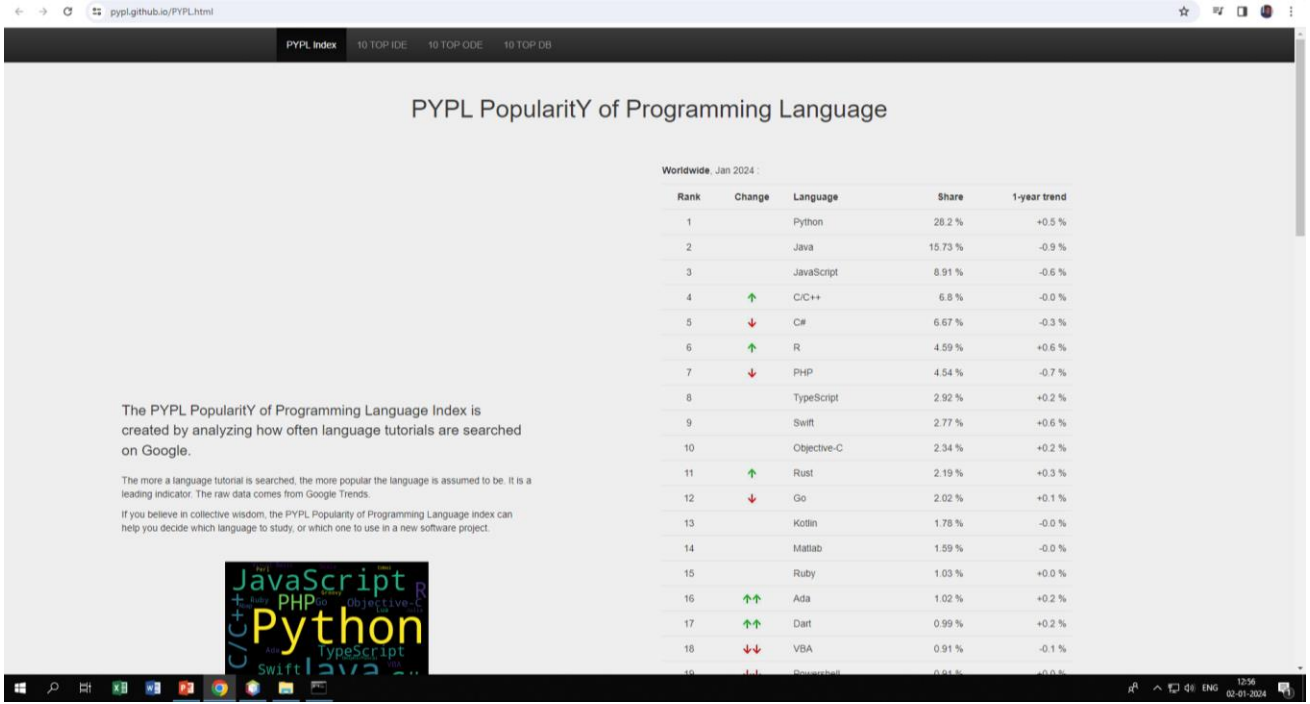
The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Dec 2023	Dec 2022	Change	Programming Language	Ratings	Change
1	1		Python	13.86%	-2.80%
2	2		C	11.44%	-5.12%
3	3		C++	10.01%	-1.92%
4	4		Java	7.99%	-3.83%
5	5		C#	7.30%	+2.38%
6	**7**	↑	**JavaScript**	**2.90%**	**-0.30%**
7	10	↑	PHP	2.01%	+0.39%
8	6	↓	Visual Basic	1.82%	-2.12%
9	8	↓	SQL	1.61%	-0.61%
10	9	↓	Assembly language	1.11%	-0.76%
11	21	⇈	Scratch	1.08%	+0.41%
12	26	⇈	Fortran	1.07%	+0.64%
13	12	↓	Go	1.03%	-0.12%
14	14		MATLAB	0.93%	-0.02%

Why Java?

www.hbpatel.in

PYPL PopularitY of Programming Language **(Source: https://pypl.github.io/PYPL.html)**



Java Programming: Modules

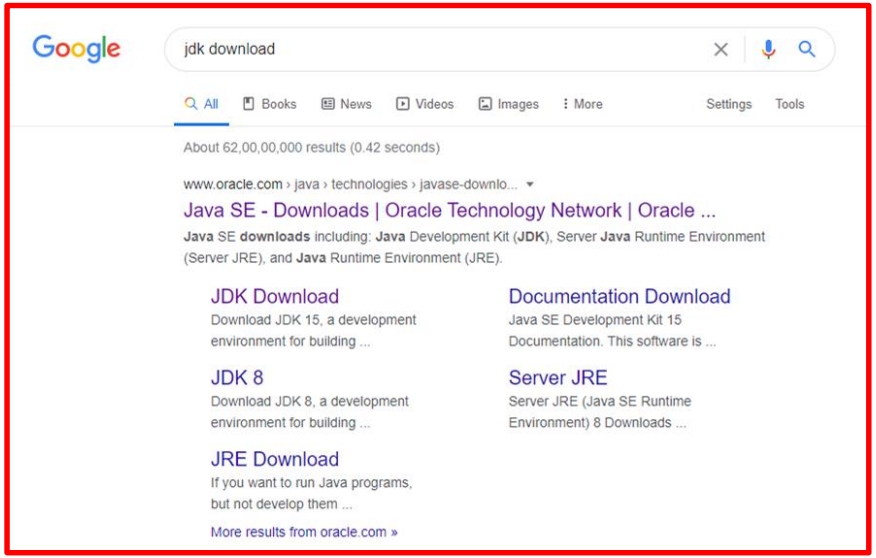
www.hbpatel.in

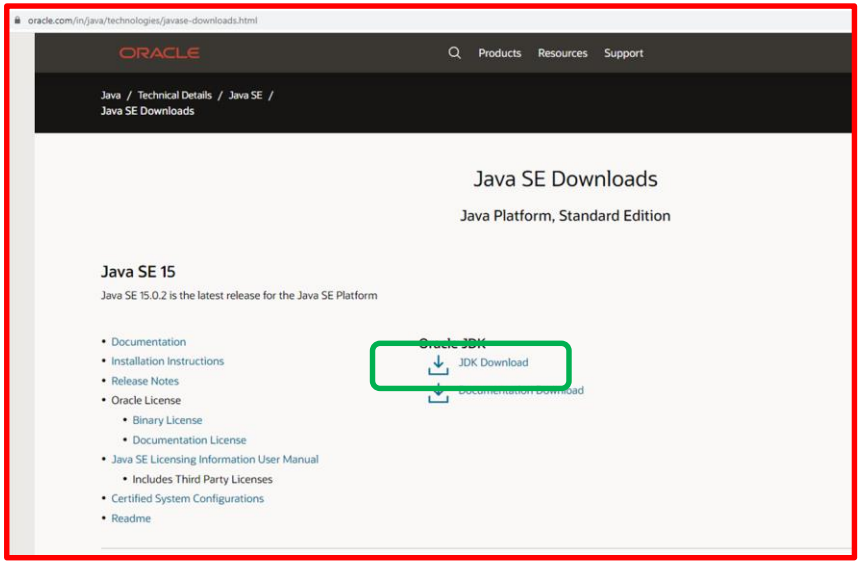
- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

First, let us install Java

Installing JDK

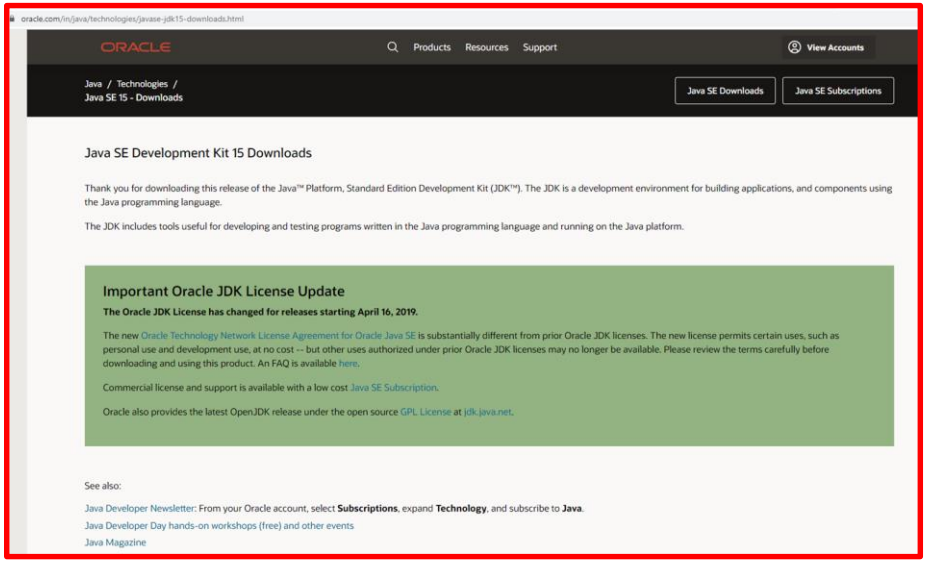
www.hbpatel.in

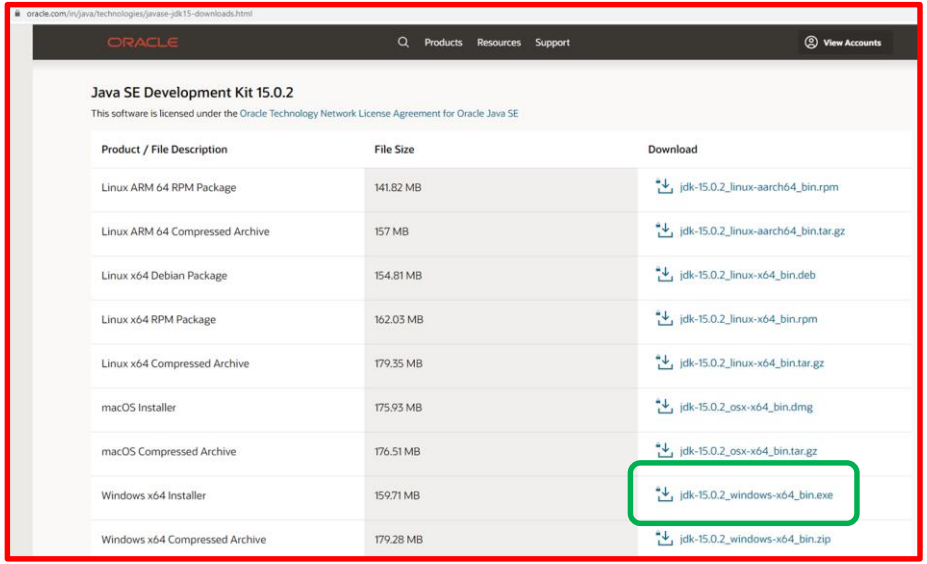




Installing JDK

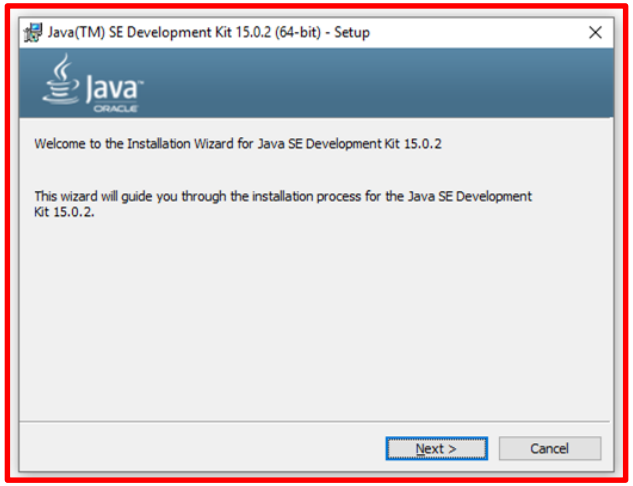
www.hbpatel.in

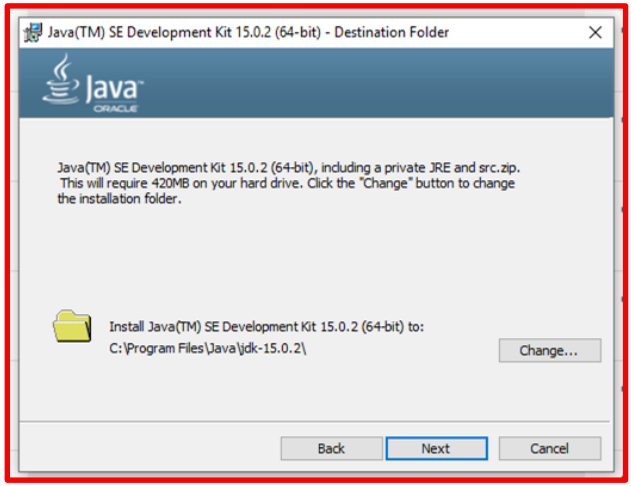


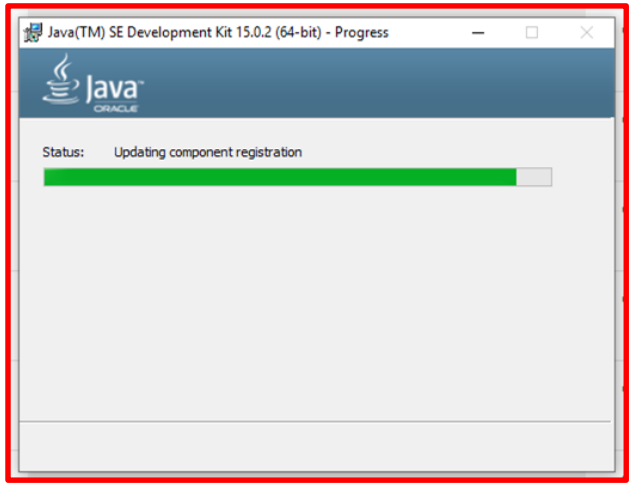


Installing JDK

www.hbpatel.in







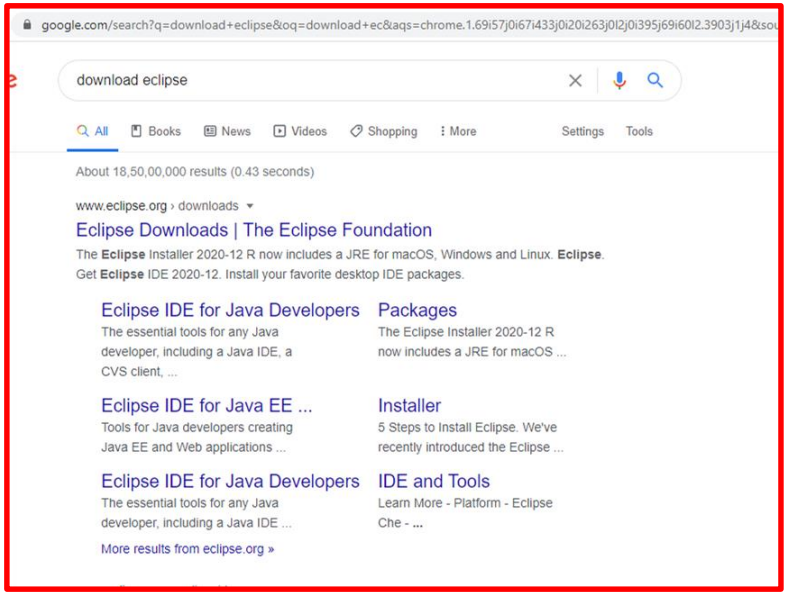
Installing Eclipse

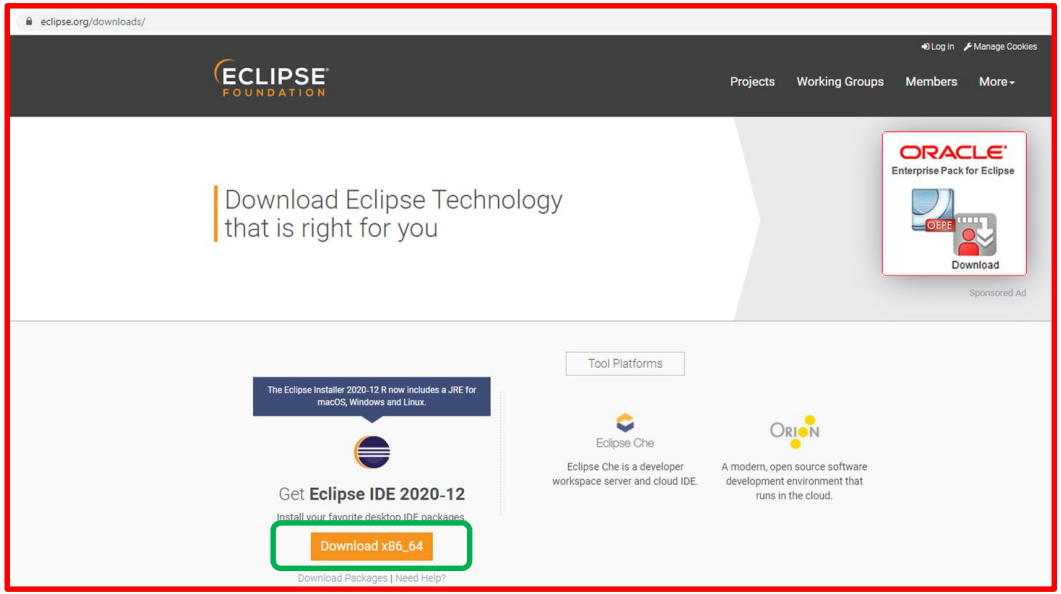
Second, let us install Java editor (Eclipse)

Or alternatively, you may go for other editors/IDEs such as NetBeans and others.

Installing Eclipse

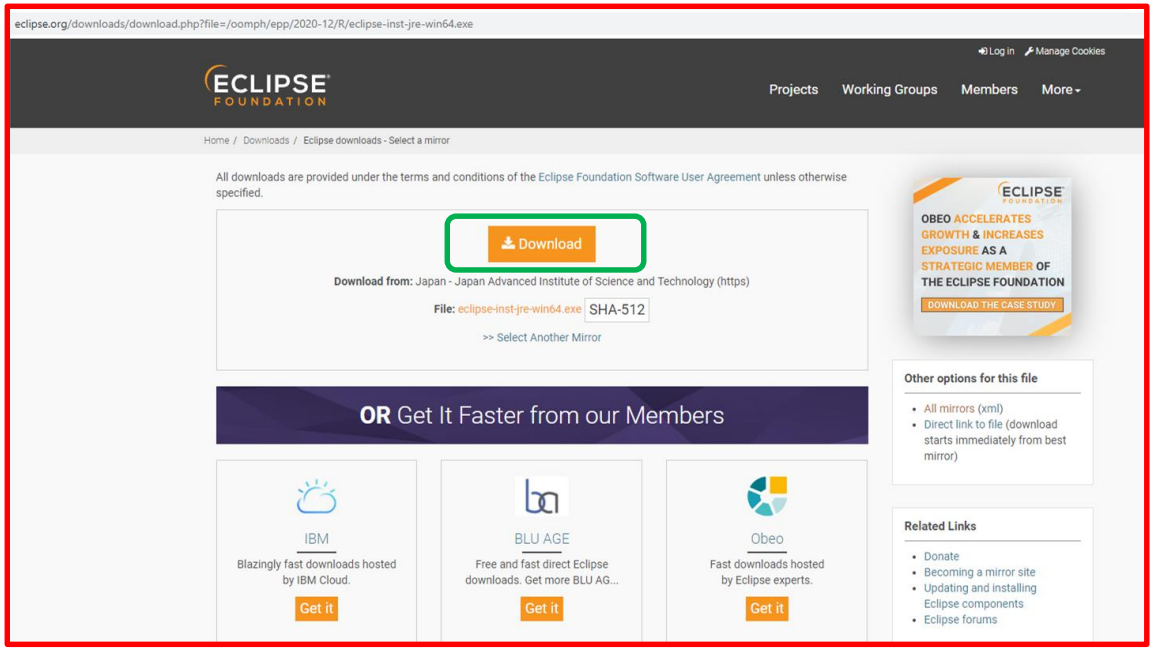
www.hbpatel.in

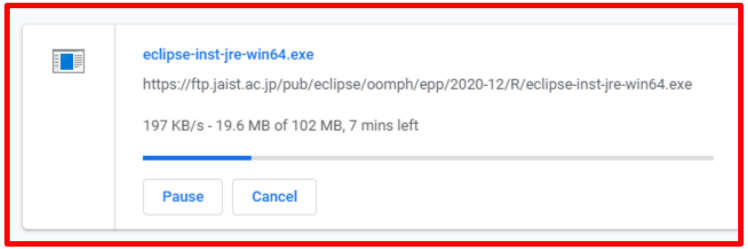




Installing Eclipse

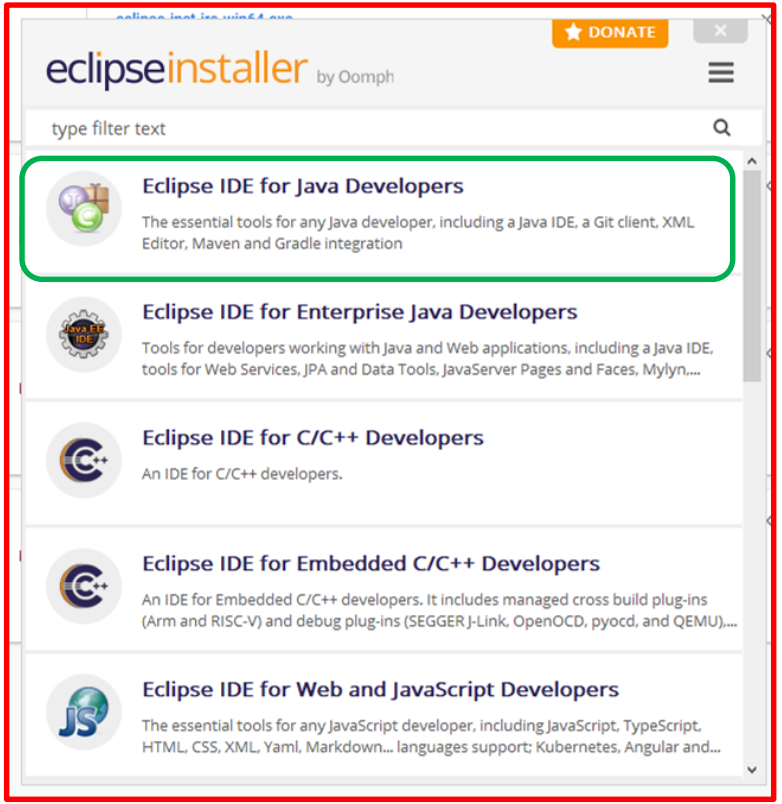
www.hbpatel.in

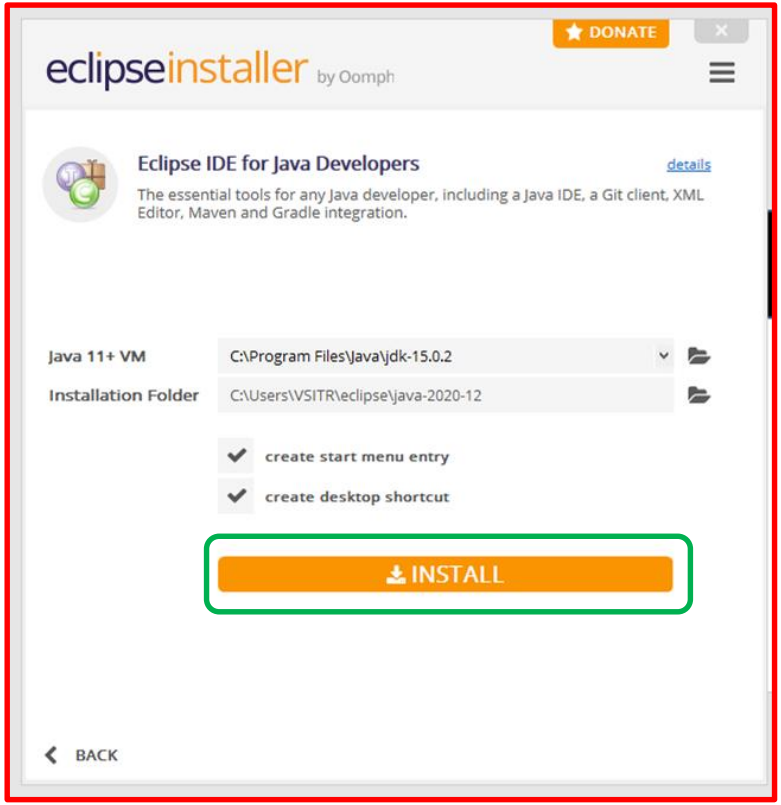




Installing Eclipse

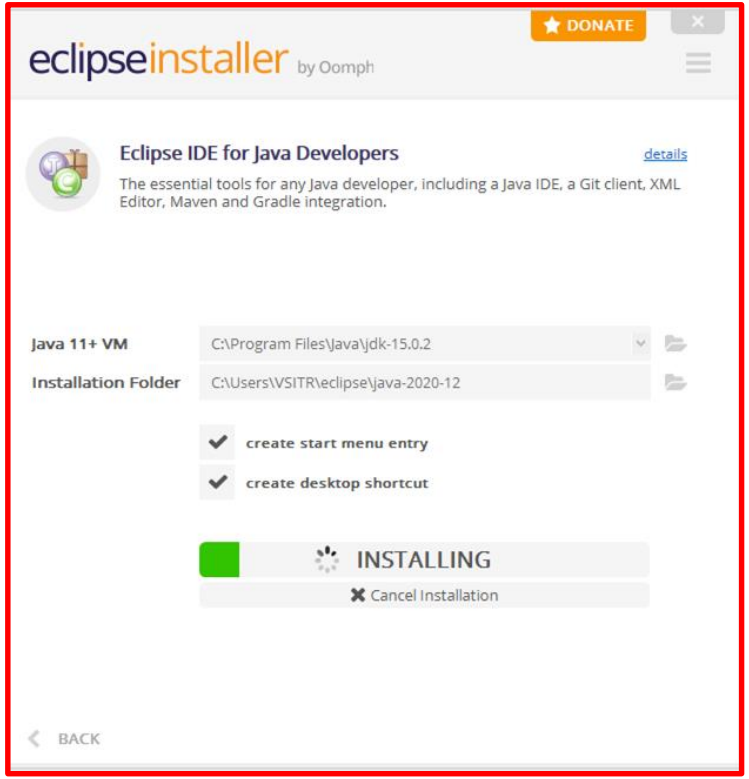
www.hbpatel.in





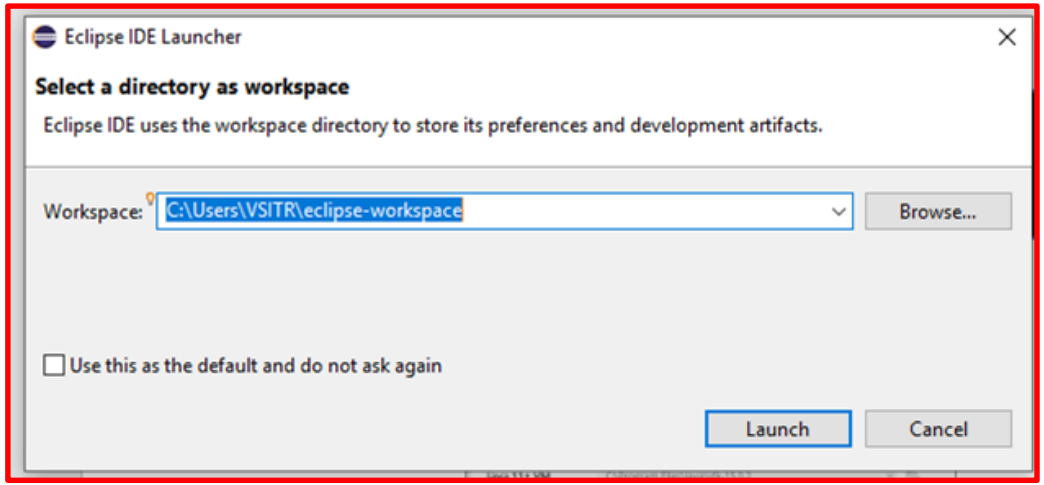
Installing Eclipse

www.hbpatel.in



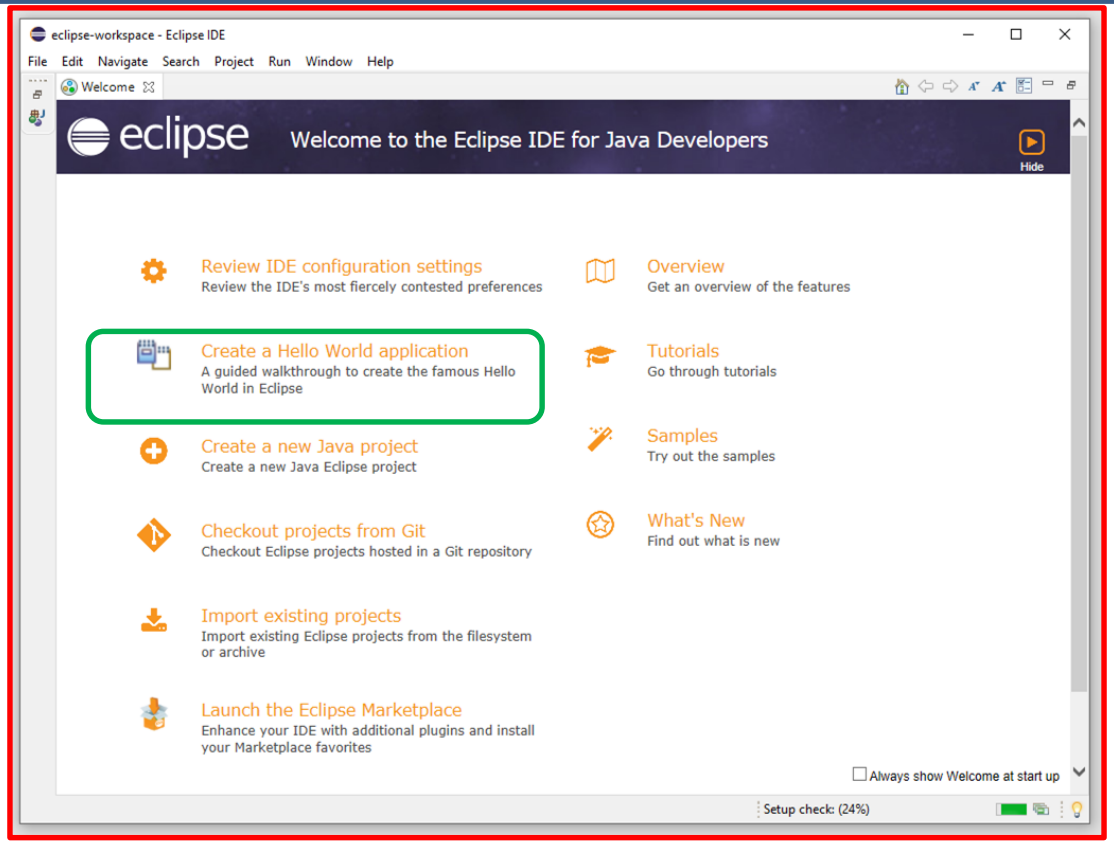
Installing Eclipse

www.hbpatel.in



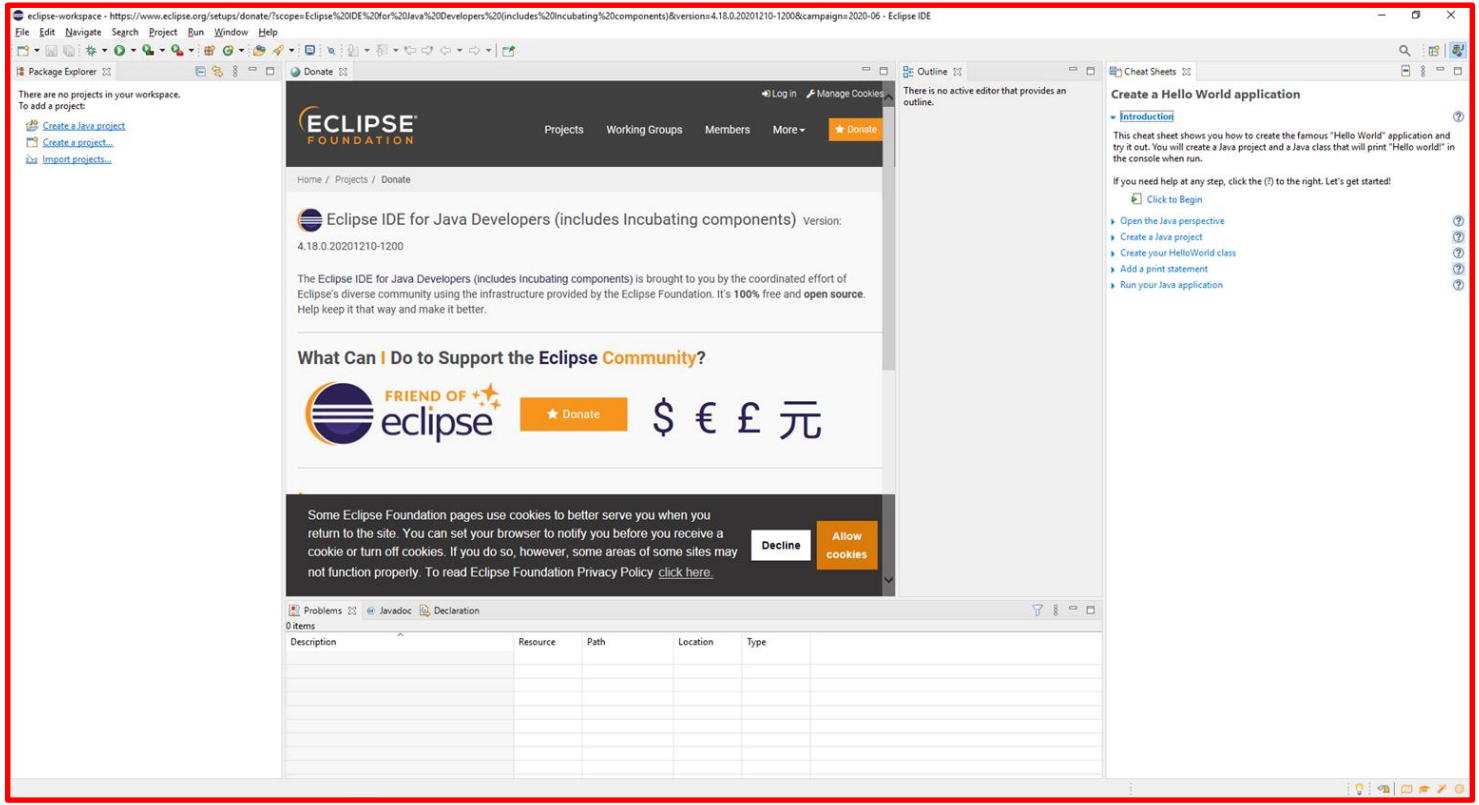
Writing an application in Eclipse

www.hbpatel.in



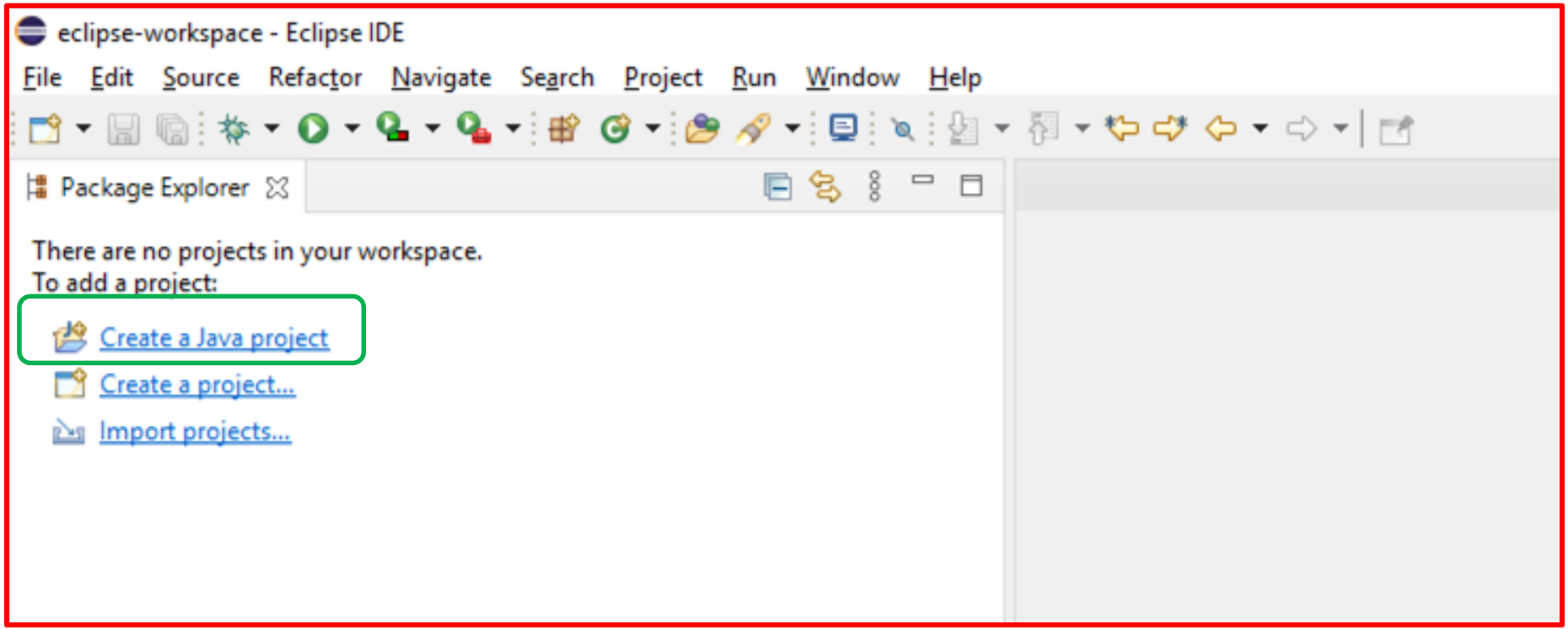
Writing an application in Eclipse

www.hbpatel.in



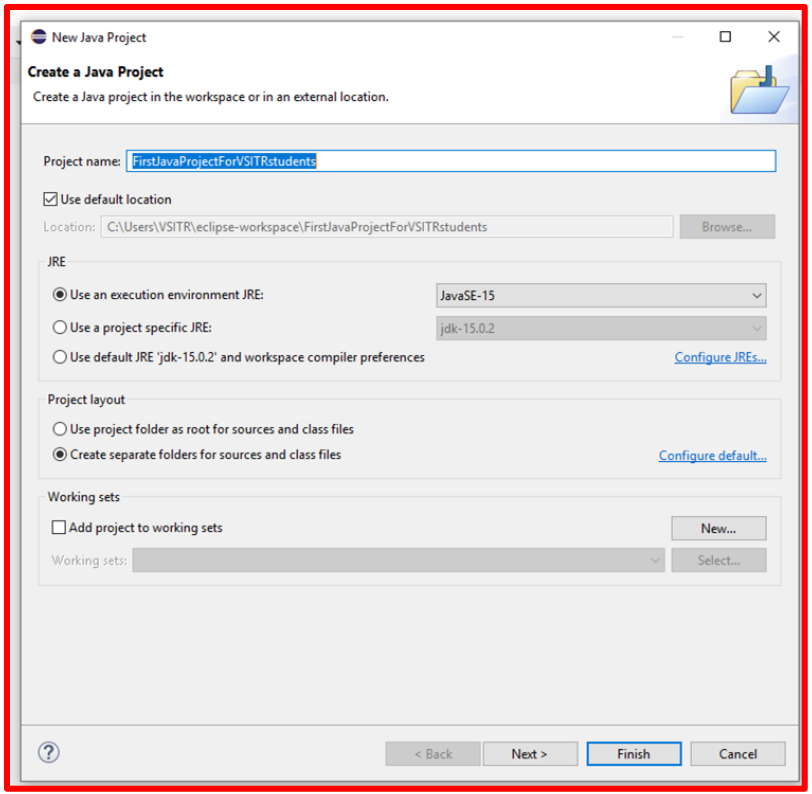
Writing an application in Eclipse

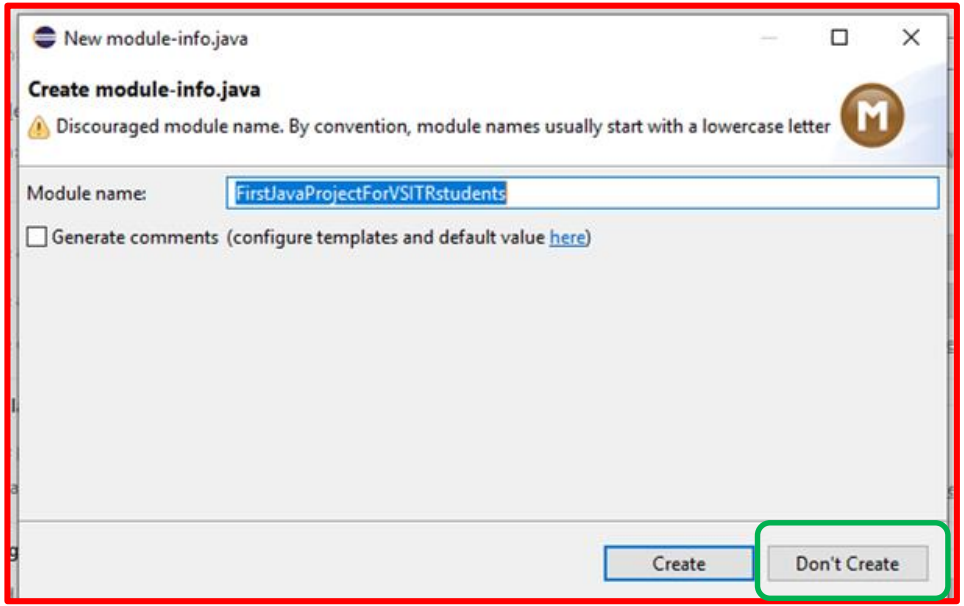
www.hbpatel.in



Writing an application in Eclipse

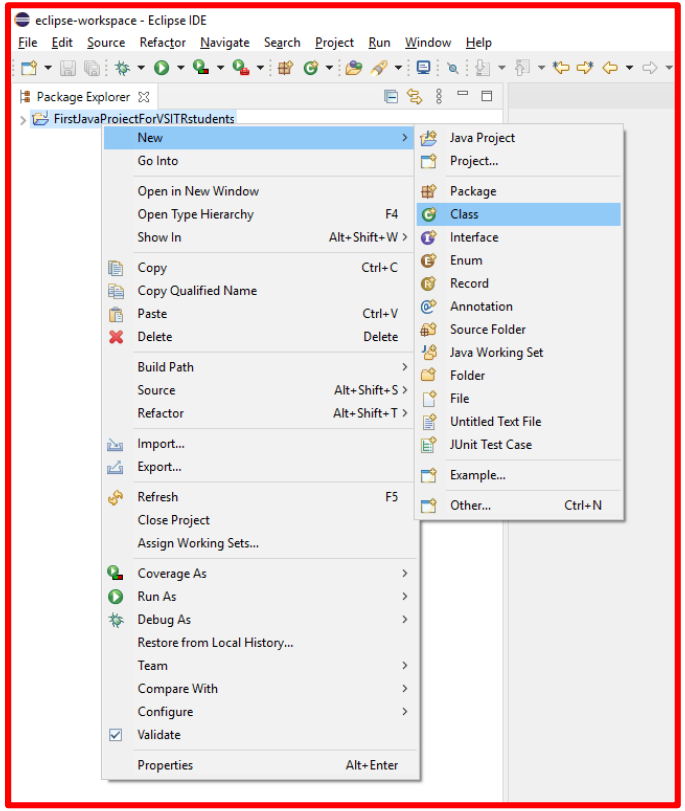
www.hbpatel.in

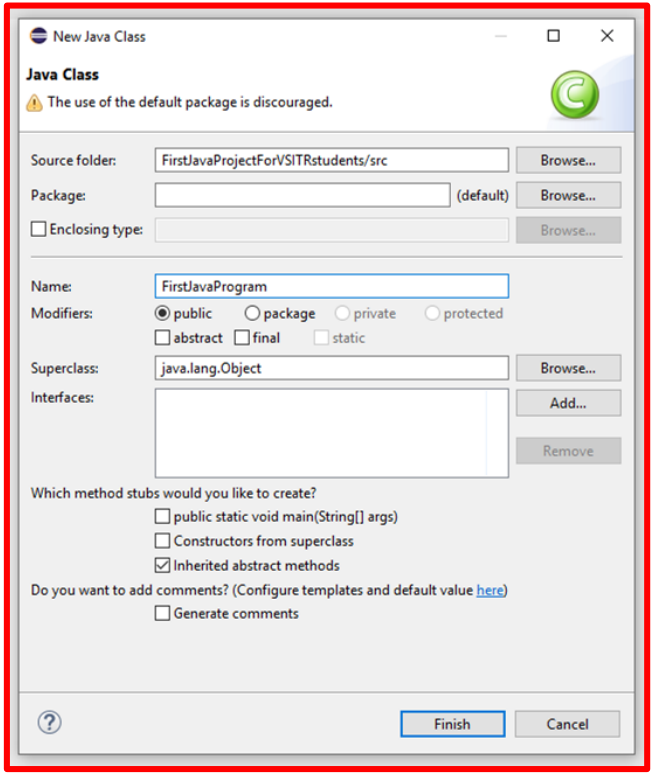




Writing an application in Eclipse

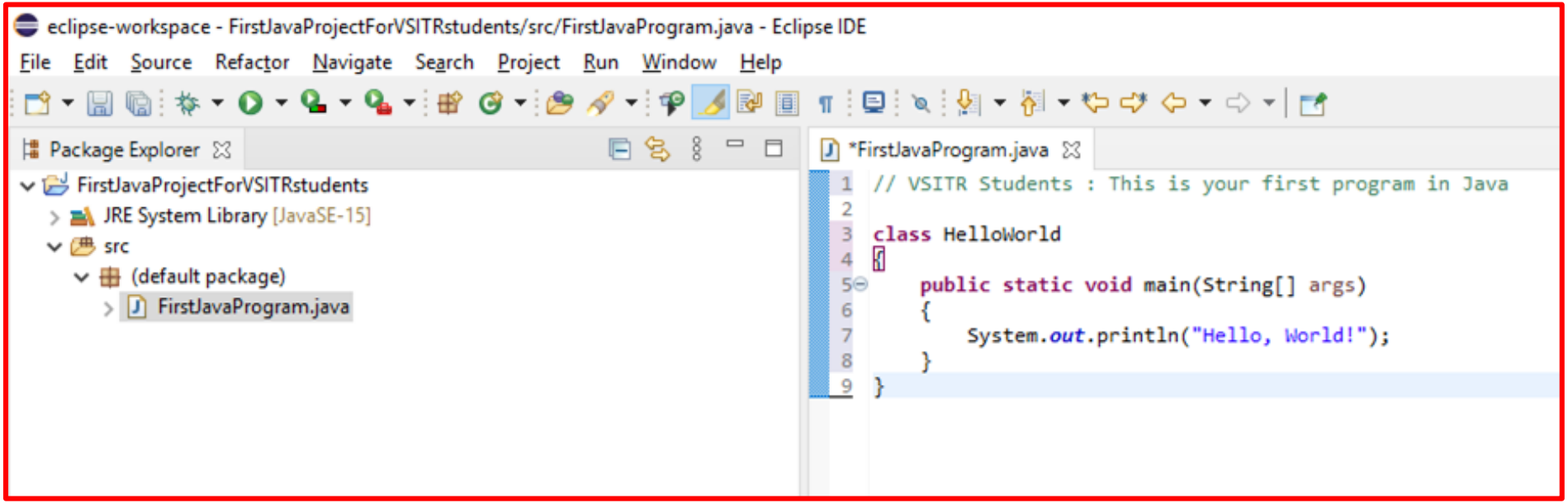
www.hbpatel.in





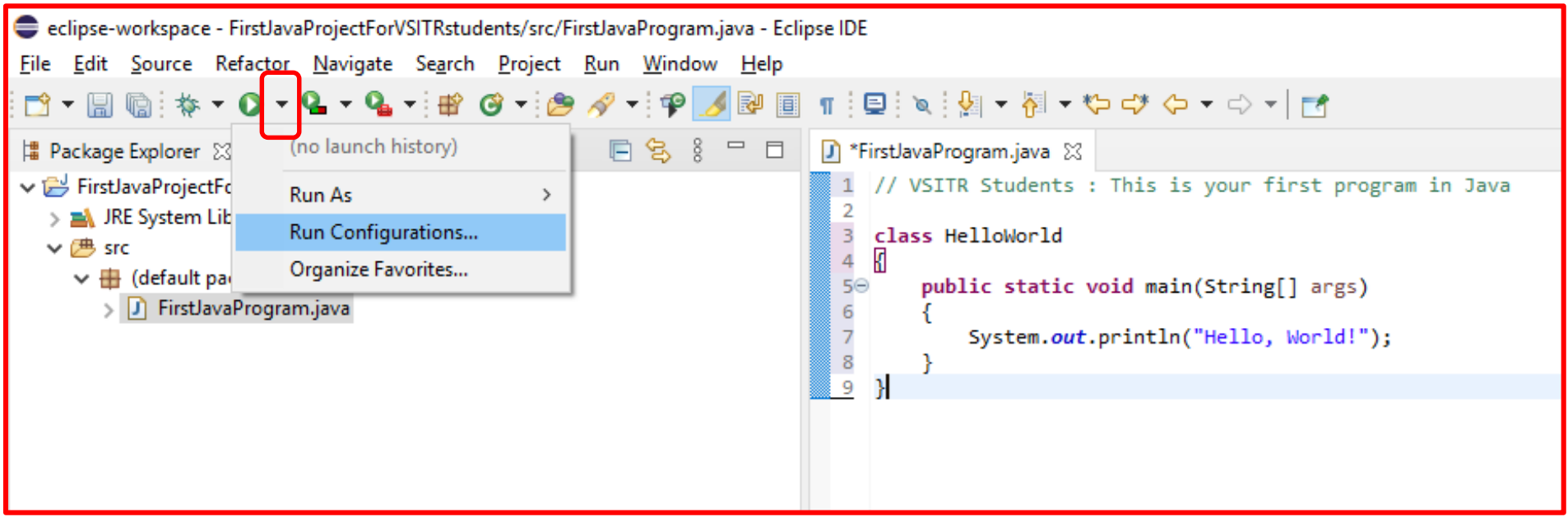
Writing an application in Eclipse

www.hbpatel.in



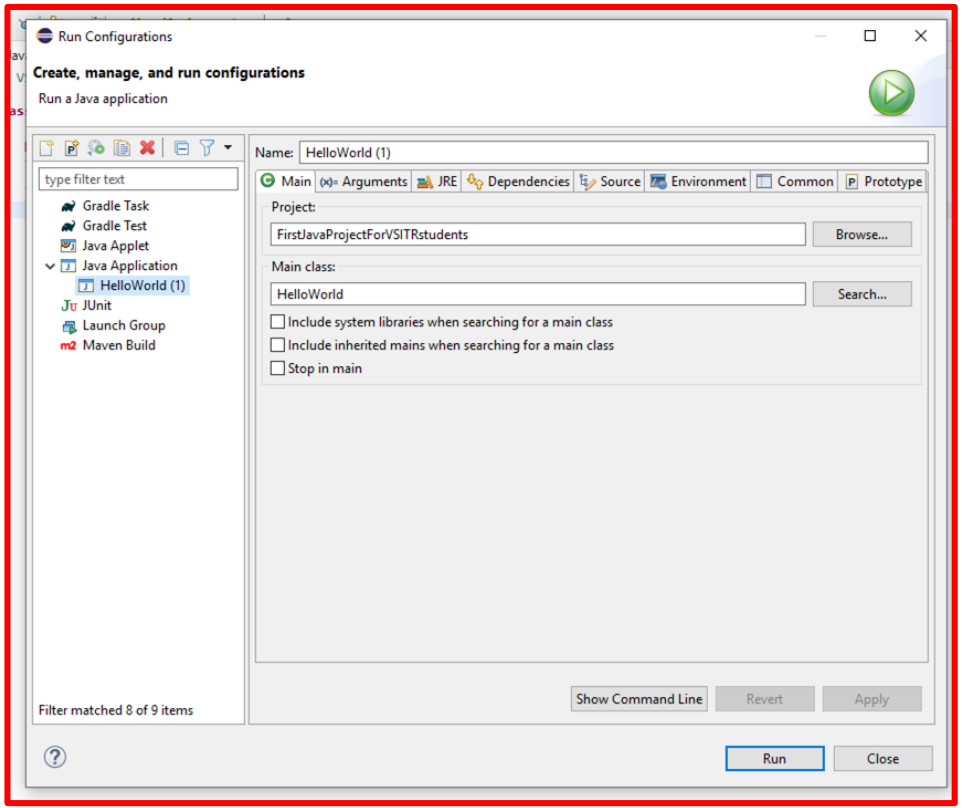
Writing an application in Eclipse

www.hbpatel.in



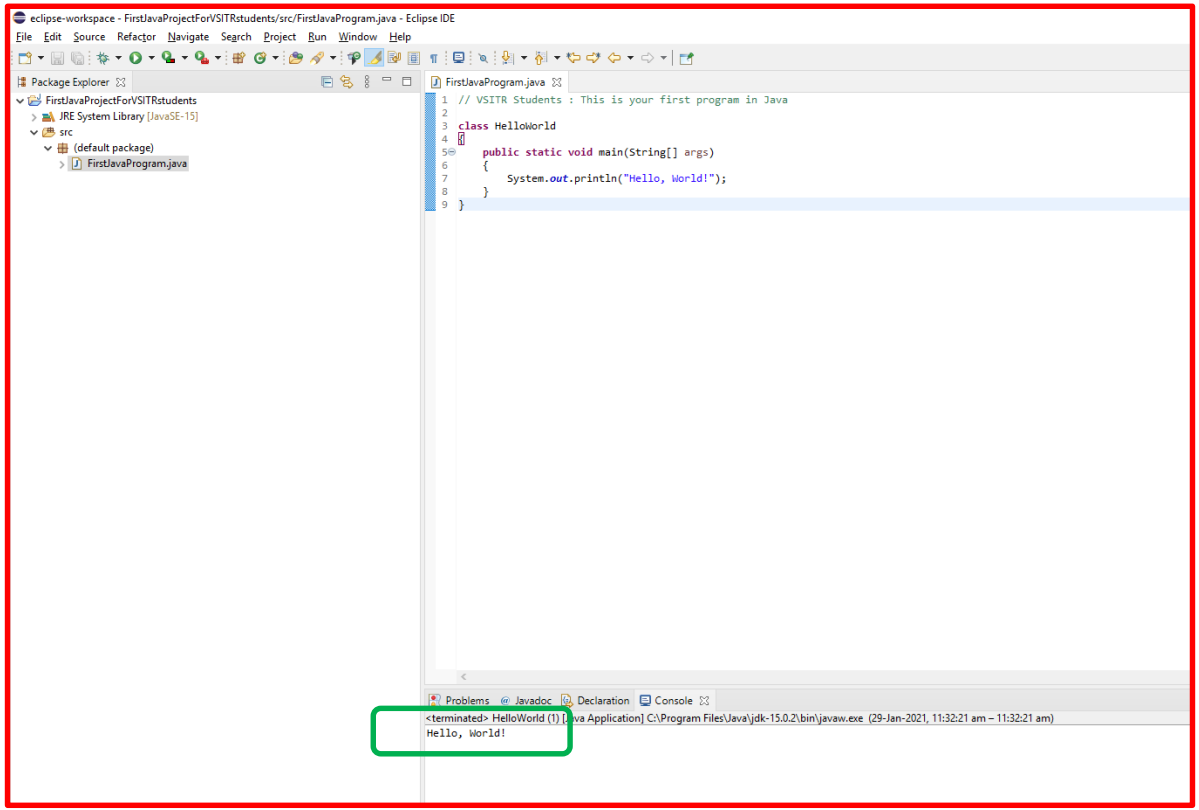
Writing an application in Eclipse

www.hbpatel.in



Writing an application in Eclipse

www.hbpatel.in



Running Java Program from Command Prompt

www.hbpatel.in

Command Prompt

```
C:\Users\VSITR>dir Hello*.*
 Volume in drive C has no label.
 Volume Serial Number is 88F7-3B84

 Directory of C:\Users\VSITR

27-12-2023  09:39               109 HelloWorld.java
               1 File(s)            109 bytes
               0 Dir(s)  224,507,420,672 bytes free

C:\Users\VSITR>type HelloWorld.java
class HelloWorld
{
        public static void main(String args[])
        {
                System.out.println("Hello World");
        }
}
```

```
}
C:\Users\VSITR>javac HelloWorld.java

C:\Users\VSITR>dir Hello*.*
 Volume in drive C has no label.
 Volume Serial Number is 88F7-3B84

 Directory of C:\Users\VSITR

27-12-2023  09:42               425 HelloWorld.class
27-12-2023  09:39               109 HelloWorld.java
               2 File(s)            534 bytes
               0 Dir(s)  224,507,379,712 bytes free

C:\Users\VSITR>java HelloWorld
Hello World

C:\Users\VSITR>
```

Java Programming: Modules

www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

Introduction to Java

www.hbpatel.in

Java is a set of computer software and specifications developed by James Gosling at **Sun Microsystems** (1995), which was later acquired by the **Oracle Corporation** (2010), that provides a system for developing application software and deploying it in a cross-platform computing environment. ***Oak*** *is a discontinued programming language created by James Gosling in 1991, initially for Sun Microsystems' set-top box project. The language later evolved to become Java.*

Source: https://en.wikipedia.org/wiki/Java_(software_platform)

Introduction to Java

Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java virtual machine (**JVM**)

The Java platform is a suite of programs that facilitate developing and running programs written in the Java programming language. A Java platform includes an execution engine (called a virtual machine), a compiler and a set of libraries; there may also be additional servers and alternative libraries that depend on the requirements.

Source: https://en.wikipedia.org/wiki/Java_(software_platform)

OOP Vs. Procedural Programming

www.hbpatel.in

Procedural Oriented Programming	Object-Oriented Programming
In procedural programming, the program is divided into small parts called ***functions***.	In object-oriented programming, the program is divided into small parts called ***objects***.
Procedural programming follows a ***top-down approach***.	Object-oriented programming follows a ***bottom-up approach***.
There is no access specifier in procedural programming.	Object-oriented programming has access specifiers like private, public, protected, etc.
Adding new data and functions is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way of hiding data so it is ***less secure***.	Object-oriented programming provides data hiding so it is ***more secure***.
In procedural programming, overloading is not possible.	Overloading is possible in object-oriented programming.
In procedural programming, there is no concept of data hiding and inheritance.	In object-oriented programming, the concept of data hiding and inheritance is used.
In procedural programming, the function is more important than the data.	In object-oriented programming, data is more important than function.
Procedural programming is based on the ***unreal world***.	Object-oriented programming is based on the ***real world***.
Procedural programming uses the concept of procedure abstraction.	Object-oriented programming uses the concept of data abstraction.
Examples: C, FORTRAN, Pascal, Basic, etc.	**Examples:** C++, Java, Python, C#, etc.

- Compare OOP with Procedural programming. [5, Jun-2022]
- List OOP characteristics and describe Inheritance [5, Oct-2023]

Source: www.geeksforgeeks.org/

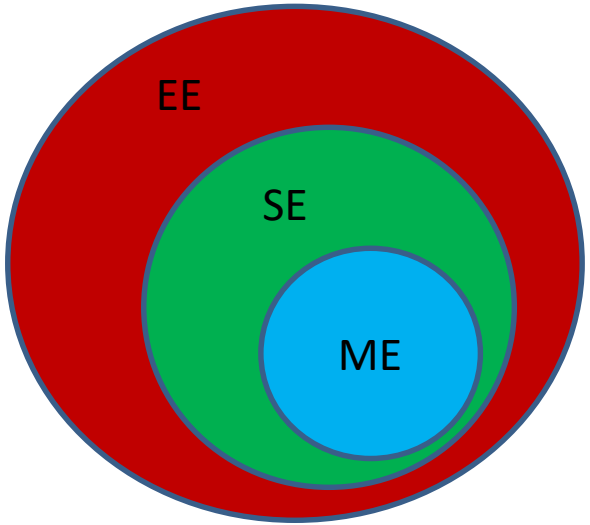
Java Editions

Java Standard Edition (SE)

Java Enterprise Edition (EE)

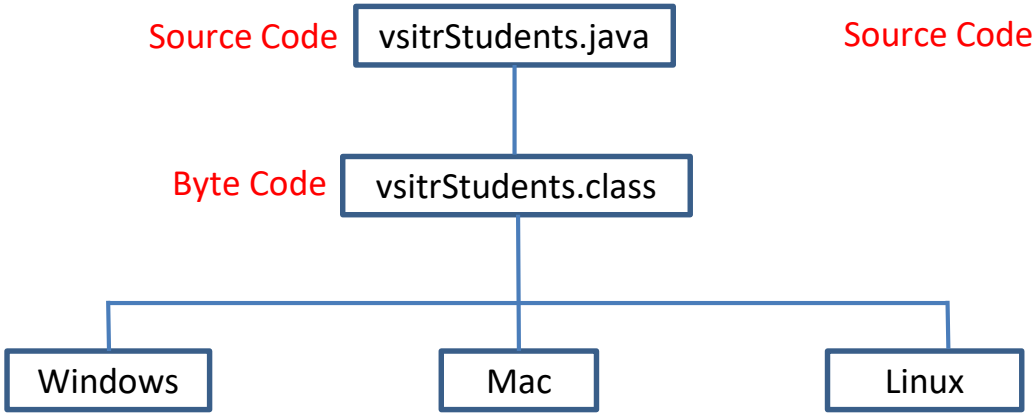
Java Micro Edition (ME)

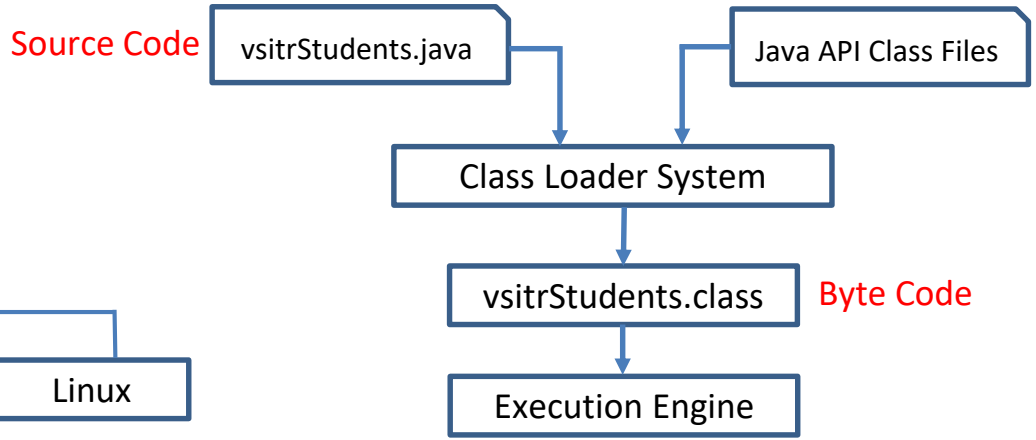
Java Card



Introduction to Java

www.hbpatel.in





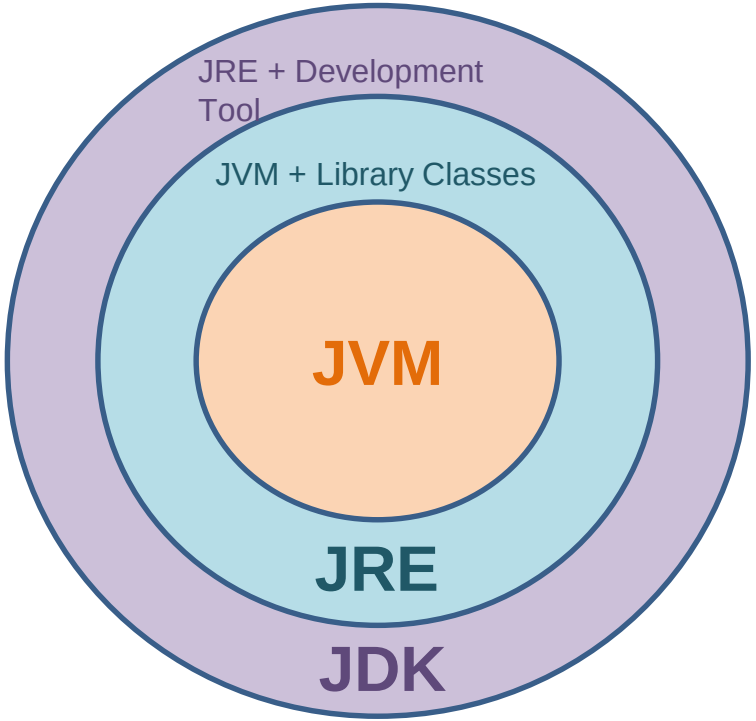
Process of compilation

Bytecode to Machine Code

- What is bytecode? Justify why Java is platform independent. [5, Jun-2022]
- Discuss the significance of bytecode. [5, Oct-2023]

Introduction to Java

www.hbpatel.in



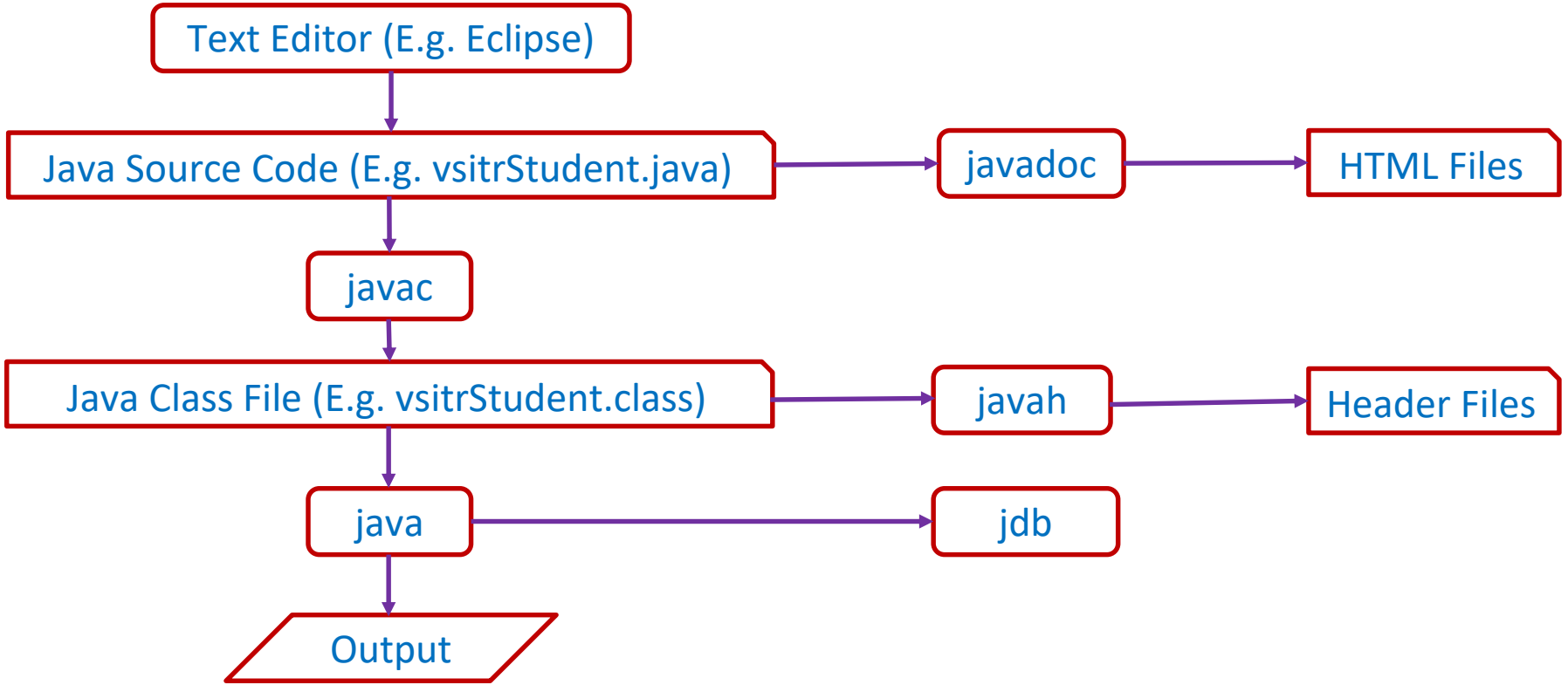
The **Java Development Kit (JDK)** is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

Java Runtime Environment (JRE) provides the minimum requirements for executing a Java application; it consists of the Java Virtual Machine (JVM), core classes, and supporting files.

Java Virtual Machine (JVM) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

Source: https://www.geeksforgeeks.org/differences-jdk-jre-jvm/

Process of building and running Java Programs



Java Features

- Compiled and interpreted
- Platform independent and portable
- Object-oriented
- Robust and secure
- Distributed
- Familiar, simple and small
- Multithreaded and interactive
- High performance
- Dynamic and extensible

List and explain the features of Java.
[5, May-2023], [5, Nov-2022], [5, Jun-2022] [5, Oct-2023]

Writing first simple Java Programs

www.hbpatel.in

```
/* A first Java Program */
class vsitrStudent  // Class name
          {
          public static void main(String args[])  // declaration of main
                    {
                    System.out.println("Welcome to VSITR");  // printing command
                    }
          }
```

Save this program as `vsitrStudent.java`. Remember, the name of program file is generally the same as name of class. (Case sensitive). Also note the case of ".java". If your program contains more than one classes, then give the name of file same as class in which "main" resides.

Compile the program with command: `javac vsitrStudent.java` (javac stands for Java Compiler)
This program will create `vsitrStudent.class` file, which is our bytecode.
Now to convert this bytecode into machine code, use command `java vsitrStudent`.
This command will convert your bytecode into machine code and will give you output.
The output will be: `Welcome to VSITR`

`static` keyword in Java

www.hbpatel.in

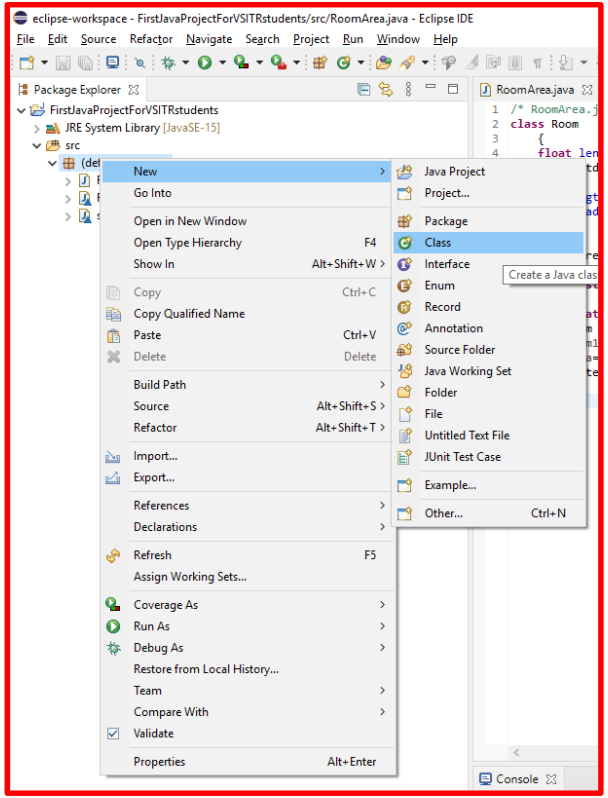
Static keyword in java in Java indicates that a particular member is not an instance, but rather part of a type. The static member will be shared among all instances of the class, so we will only create one instance of it.

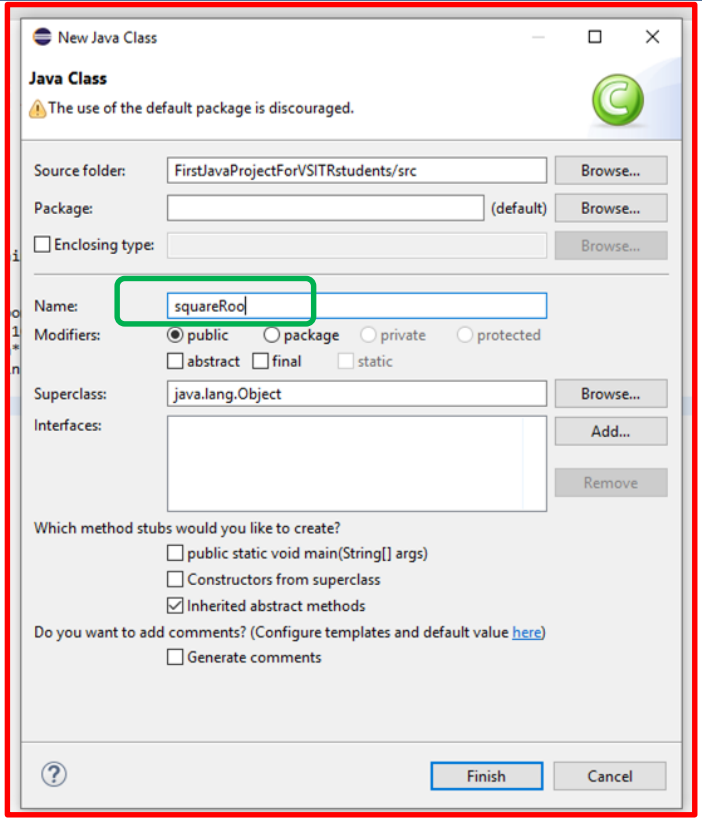
If any member in a class is declared as static, it means that even before the class is initiated, all the static members can be accessed and become active. In contrast to this, non-static members of the same class will cease to exist when there is no object or the object goes out of scope.

Explain "`static`" keyword with proper example. (b) [2.5, Nov-2022]

Writing few simple Java Programs

www.hbpatel.in



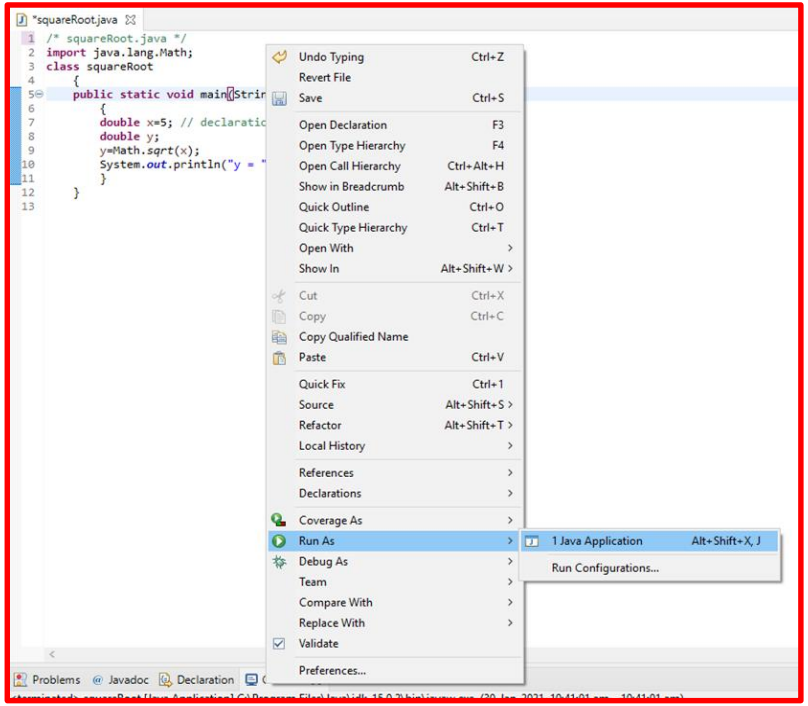


Simple Java Programs: Square Root

www.hbpatel.in

```java
/* squareRoot.java */
import java.lang.Math;
class squareRoot
    {
    public static void main(String args[])
        {
        double x=5; // declaration and initialization
        double y;
        y=Math.sqrt(x);
        System.out.println("y = "+y);
        }
    }
```

```
Problems  Javadoc  Declaration  Console
<terminated> squareRoot [Java Application] C:\Program Files\Java\jdk
y = 2.23606797749979
```



Code: https://github.com/hbpatel1976/Java/blob/main/squareRoot.java

Simple Java Programs: Area of a Room

www.hbpatel.in

RoomArea.java

```
1  /* RoomArea.java */
2  class Room
3      {
4      float length,breadth;
5      void getdata(float a, float b)
6          {
7          length=a;
8          breadth=b;
9          }
10     }
11 class RoomArea
12     {
13     public static void main(String args[])
14         {
15         float area;
16         Room room1=new Room();  // Creates an object room1
17         room1.getdata(14,10);
18         area=room1.length*room1.breadth;
19         System.out.println("Area = " + area);
20         }
21     }
```

Console

<terminated> RoomArea [Ja

```
Area = 140.0
```

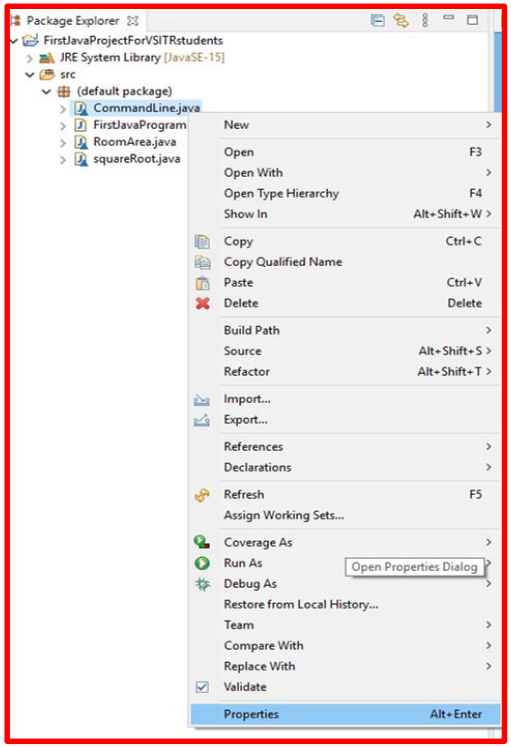
Code: https://github.com/hbpatel1976/Java/blob/main/RoomArea.java

Simple Java Programs: Command Line Argument

www.hbpatel.in

*CommandLine.java

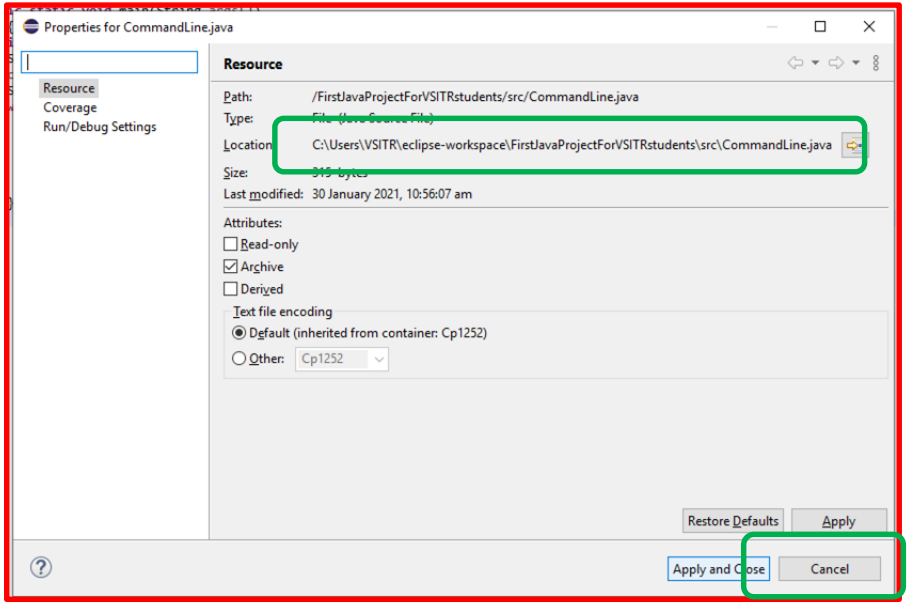
```
1  /* CommandLine.java */
2  class CommandLine
3  {
4  public static void main(String args[])
5      {
6      int count,i=0;
7      String string;
8      count=args.length;
9      System.out.println("Number of arguments = "+count);
10     while ( i < count)
11         {
12         string=args[i];
13         i=i+1;
14         System.out.println(i+":"+"Hello "+string+"!");
15         }
16     }
17 }
```

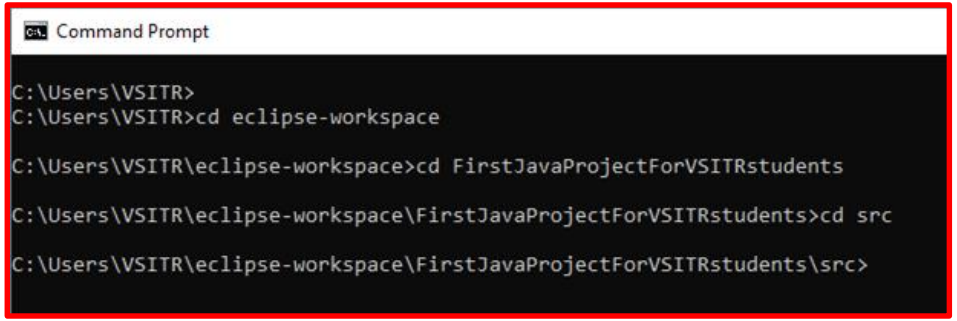


Code: https://github.com/hbpatel1976/Java/blob/main/CommandLine.java

Simple Java Programs: Command Line Argument

www.hbpatel.in





Simple Java Programs: Command Line Argument

www.hbpatel.in

*CommandLine.java

```
1  /* CommandLine.java */
2  class CommandLine
3  {
4  public static void main(String args[])
5      {
6      int count,i=0;
7      String string;
8      count=args.length;
9      System.out.println("Number of arguments = "+count);
10     while ( i < count)
11         {
12         string=args[i];
13         i=i+1;
14         System.out.println(i+":"+"Hello "+string+"!");
15         }
16     }
17 }
```

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\src>javac CommandLine.java

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\src>java CommandLine VSITR Kadi KSV Gandhinagar
Number of arguments = 4
1:Hello VSITR!
2:Hello Kadi!
3:Hello KSV!
4:Hello Gandhinagar!

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\src>
```

Simple Java Programs: Reading Data from Keyboard

www.hbpatel.in

```
import java.io.DataInputStream;
class Reading
{
public static void main(String args[])
          {
          DataInputStream in=new DataInputStream(System.in);
          int intNumber=0;
          float floatNumber=0.0f;
          try
                    {
                    System.out.println("Enter an integer: ");
                    intNumber=Integer.parseInt(in.readLine());
                    System.out.println("Enter a float number: ");
                    floatNumber=Float.valueOf(in.readLine()).floatValue();
                    }
          catch(Exception e) {System.out.println("Exception Generated"); }
          System.out.println("intNumber = " +intNumber);
          System.out.println("floatNumber = " +floatNumber);
          }
}
```

OUTPUT

```
Enter an integer:
10
Enter a float number:
20.40
intNumber = 10
floatNumber = 20.4
```

OUTPUT

```
Enter an integer:
Abc
Exception generated
intNumber=0
floatNumber=0
```

Code: https://github.com/hbpatel1976/Java/blob/main/Reading.java

Java Data Types

www.hbpatel.in

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Source: https://www.w3schools.com/

Operators in Java

www.hbpatel.in

Arithmetic

* Multiplication
/ Division
% Modulo
+ Addition
–Subtraction

Unary

– Unary minus
+ Unary plus
++ Increment
– – Decrement
! Logical not

Assignment

= Assignment
+= -= *= /= %=

Relational

== Equal to
!= Not Equal to
< less than
<= less than or equal to
> Greater than
>= Greater than or equal to

Logical

&& Logical AND
|| Logical OR
! Logical NOT

Ternary operator

? :

Bitwise

& Bitwise AND
| Bitwise OR
^ Bitwise XOR
~ Bitwise Complement

Shift

<< Left shift operator
>> Signed Right shift
>>> Unsigned Right shift

List the various operators in Java and explain any two with proper example. [5, May-2023]

Simple Java Programs: Creation and casting of variables

www.hbpatel.in

```
class TypeWrap
{
public static void main(String args[])
{
System.out.println("Variables Created");
char c='x';
byte b=50;
short s=1996;
int i=123456789;
long l=1234567654321L;
float f1=3.142f;
float f2=1.2e-5f;
double d2=0.000000987;
System.out.println("c = " + c);
System.out.println("b = " + b);
System.out.println("s = " + s);
System.out.println("i = " + i);
System.out.println("l = " + l);
System.out.println("f1 = " + f1);
System.out.println("f1 = " + f2);
System.out.println("d2 = " + d2);
System.out.println(" ");
System.out.println("Types Converted");
short s1=(short)b;
short s2=(short)i;
float n1=(float)i;
int m1=(int)f1;
System.out.println("(short)b = " + s1);
System.out.println("(short)i = " + s2);
System.out.println("(float)l = " + n1);
System.out.println("(int)f1 = " + m1);
}
}
```

OUTPUT

```
Variables Created
c = x
b = 50
s = 1996
i = 123456789
l = 1234567654321
f1 = 3.142
f1 = 1.2E-5
d2 = 9.87E-7

Types Converted
(short)b = 50
(short)i = -13035
(float)l = 1.23456792E8
(int)f1 = 3
```

Code: https://github.com/hbpatel1976/Java/blob/main/TypeWrap.java

Simple Java Programs: Relational Operators

www.hbpatel.in

```
class Relation
        {
        public static void main(String args[])
                {
                float a=15.0F,b=20.75F,c=15.05F;
                System.out.println("a = "+a);
                System.out.println("b = "+b);
                System.out.println("c = "+c);
                System.out.println("a < b = "+(a<b));
                System.out.println("a > b = "+(a>b));
                System.out.println("a == c = "+(a==b));
                System.out.println("a <= c = "+(a<=c));
                System.out.println("a >= b = "+(a>=b));
                System.out.println("b != c = "+(b!=c));
                System.out.println("b == a+c = "+(b==a+c));
                }
        }
```

OUTPUT

```
a = 15.0
b = 20.75
c = 15.05
a < b = true
a > b = false
a == c = false
a <= c = true
a >= b = false
b != c = true
b == a+c = false
```

Code: https://github.com/hbpatel1976/Java/blob/main/Relation.java

Java Control Statements

www.hbpatel.in

- Decision Making statements
  - if statements
    - Simple if statement
    - if-else statement
    - if-else-if ladder
    - Nested if-statement
  - switch statement
- Loop statements
  - do while loop
  - while loop
  - for loop
- Jump statements
  - break statement
  - continue statement

What is control structures and selection in Java? Explain it with example. [5, May-2023]

Simple Java Programs: if statement

www.hbpatel.in

```
class IfTest
{
public static void main(String args[])
            {
            int i,count,count1,count2;
            float[] weight = {45.0F,55.0F,47.0F,51.0F,54.0F};
            float[] height = {176.5F,174.2F,168.0F,170.7F,169.0F};
            count=0;
            count1=0;
            count2=0;
            for(i=0; i<=4; ++i)
                        {
                        if(weight[i]<50.0 && height[i]>170.0)count1++;
                        count++;
                        }
            count2=count-count1;
            System.out.println("Number of persons with ... ");
            System.out.println("Weight<50 and height>170 = "+count1);
            System.out.println("Others = "+count2);
            }
}
```

OUTPUT

```
Number of persons with ...
Weight<50 and height>170 = 1
Others = 4
```

Code: https://github.com/hbpatel1976/Java/blob/main/IfTest.java

Simple Java Programs: if-else statement

www.hbpatel.in

```
class IfElseTest
{
public static void main(String args[])
        {
        int number[] = {50,65,56,71,81};
        int even=0,odd=0;
        for(int i=0; i<5; ++i)
                {
                if(number[i]%2==0){even++;}
                else{odd++;}
                }
        System.out.println("Even Numbers : "+even+"    Odd Numbers = "+odd);
        }
}
```

OUTPUT

```
Even Numbers : 2    Odd Numbers = 3
```

Code: https://github.com/hbpatel1976/Java/blob/main/IfElseTest.java

Simple Java Programs: if-else-if ladder statement

www.hbpatel.in

```
class ElseIfLadder
{
public static void main(String args[])
        {
        int rollno[] = { 111,222,333,444 };
        int marks[] = {81,75,43, 58};
        for(int i=0; i<rollno.length; ++i)
                {
                if(marks[i] > 79)System.out.println(rollno[i]+" - Honours");
                else if(marks[i] > 59)System.out.println(rollno[i]+" - I division");
                else if(marks[i] > 49)System.out.println(rollno[i]+" - II division");
                else System.out.println(rollno[i]+" - III division");
                }
        }
}
```

OUTPUT

```
111 - Honours
222 - I division
333 - III division
444 - II division
```

Code: https://github.com/hbpatel1976/Java/blob/main/ElseIfLadder.java

Simple Java Programs: switch statement

www.hbpatel.in

```
class CityGuide
{
public static void main(String args[])
              {
              char choice;
              System.out.println("Select your choice");
              System.out.println("     M  :   Madras");
              System.out.println("     B  :   Bombay");
              System.out.println("     C  :   Calcutta");
              System.out.println("Choice : ");
              System.out.flush();
              try
              {
              switch(choice=(char)System.in.read())
                            {
                            case 'M':
                            case 'm':     System.out.println("Madras : Booklet 5");break;
                            case 'B':
                            case 'b':     System.out.println("Bombay : Booklet 9");break;
                            case 'C':
                            case 'c':     System.out.println("Calcutta : Booklet  15");break;
                            default:      System.out.println("Invalid choice ");
                            }
              }
              catch(Exception e) {System.out.println("I/O Error");}
              }
}
```

OUTPUT

```
Select your choice
     M  :   Madras
     B  :   Bombay
     C  :   Calcutta
Choice :
M
Madras : Booklet 5
```

OUTPUT

```
Select your choice
     M  :   Madras
     B  :   Bombay
     C  :   Calcutta
Choice :
x
Invalid choice
```

Code: https://github.com/hbpatel1976/Java/blob/main/CityGuide.java

Arrays: Initialize 1-D Array

www.hbpatel.in

```
// Initialize 1-D Array
public class arr01
    {
    public static void main(String args[])
        {
            int arr[] = new int[] {89,68,36,45,93};
            int i;
            for(i=0; i<arr.length; ++i)System.out.println(arr[i]);
        }
    }
```

OUTPUT
89
68
36
45
93

Code: https://github.com/hbpatel1976/Java/blob/main/arr01.java

Arrays: Assigning values to 1-D array

www.hbpatel.in

```
public class arr02
    {
    public static void main(String args[])
        {
            int arr[], i;
            arr = new int[5];
            arr[0]=10;
            arr[1]=20;
            arr[2]=30;
            arr[3]=40;
            arr[4]=50;
            for(i=0; i<arr.length; ++i)System.out.println(arr[i]);
        }
    }
```

OUTPUT
10
20
30
40
50

Code: https://github.com/hbpatel1976/Java/blob/main/arr02.java

Simple Java Programs: Pattern

www.hbpatel.in

```
class displayPattern
{
public static void main(String args[])
        {
        System.out.println("Screen Display");
        for(int i=1; i<9; ++i)
                {
                for(int j=1; j<=i; ++j)
                        {
                        System.out.print(" ");
                        System.out.print(i);
                        }
                System.out.print("\n");
                }
        System.out.println("Screen Display Done");
        }
}
```

OUTPUT

```
Screen Display
 1
 2 2
 3 3 3
 4 4 4 4
 5 5 5 5 5
 6 6 6 6 6 6
 7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8
Screen Display Done
```

Code: https://github.com/hbpatel1976/Java/blob/main/displayPattern.java

Matrix Multiplication

www.hbpatel.in

```
class matrixMultiplication
{
public static void main(String args[])
        {
                int row = 3, column = 3, i, j, k;
                int [][] matrixA = {{1,2,3},{2,3,4},{3,4,5}};
                int [][] matrixB = {{9,8,7},{6,5,4},{3,2,1}};
                int [][] matrixC = new int[row][column];
                for(i=0; i<row; ++i)
                        {
                        for(j=0; j<column; ++j)
                                {matrixC[i][j]=0;
                                for(k=0; k<column; ++k)
                                        {matrixC[i][j]+= matrixA[i][k]*matrixB[k][j];}
                                System.out.print(matrixC[i][j] + "   ");
                                }
                         System.out.println();
                        }
        }
}
```

OUTPUT

```
30   24   18
48   39   30
66   54   42
```

Write a Java program for matrix multiplication. [5, May-2023] [5, Oct-2023]

Code: https://github.com/hbpatel1976/Java/blob/main/matrixMultiplication.java

Fibonacci Series

www.hbpatel.in

```
public class Fibonacci
{
    public static void main(String args[])
    {
        int term1=0, term2=1, term3, i, totalTerm=10;
        System.out.print(term1+" , "+term2);
        for(i=2;i<totalTerm;++i)
        {
            term3 = term1 + term2;
            System.out.print(" , " + term3);
            term1 = term2;
            term2 = term3;
        }
    }
}
```

OUTPUT

```
0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34
```

Write a Java program to implement Fibonacci series using for loop control structure. [5, May-2023]

Code: https://github.com/hbpatel1976/Java/blob/main/Fibonacci.java

Prime Number

www.hbpatel.in

```
public class primeNumbers
{
    public static void main(String[] args)
        {
            for(int i=2; i<100; ++i)isPrime(i);
         }
    public static void isPrime(int n)
        {
            boolean prime=true;
            for (int i = 2; i <= n / 2; i++)if (n % i == 0)prime = false;
            if(prime==true)System.out.print(n+"\t");
         }
}
```

OUTPUT

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Code: https://github.com/hbpatel1976/Java/blob/main/primeNumbers.java

Arrays: Initializing values: Array of objects

www.hbpatel.in

```
class Student
{
    public int rollno;
    public String name;
    Student (int r, String n) {this.rollno=r; this.name=n;}
}
public class arr03
    {
    public static void main(String args[])
        {
            Student s[]= new Student[]{new Student(10,"Pradip"), new Student(20,"Sanjay"), new Student(30,"Vijay")};
            for (int i=0; i<s.length; ++i)
            {
                System.out.println("Rollno " + i + " = " + s[i].rollno + "\tName " + i + " = " + s[i].name);
            }
        }
    }
```

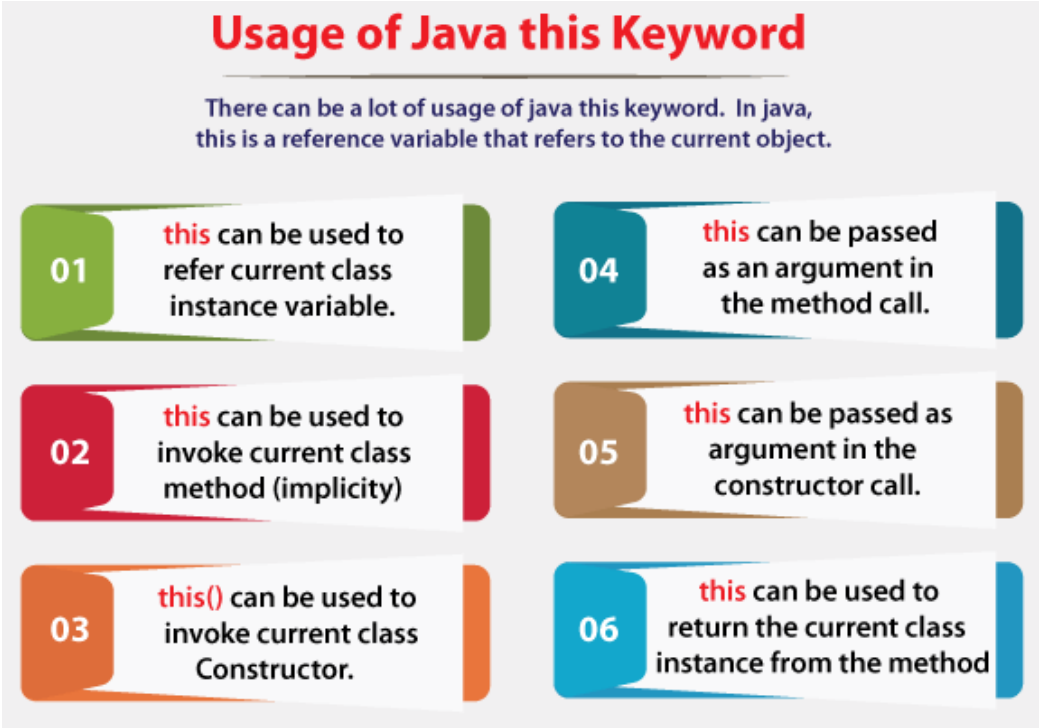
OUTPUT

```
Rollno 0 = 10    Name 0 = Pradip
Rollno 1 = 20    Name 1 = Sanjay
Rollno 2 = 30    Name 2 = Vijay
```

Code: https://github.com/hbpatel1976/Java/blob/main/arr03.java

“this” key word

www.hbpatel.in



- Explain “`this`” keyword with proper example. (a) this [2.5, May-2023], [2.5, Nov-2022]
- Explain following keywords with example. (a) `this` (b) `final` (c) `super`. [5, Jun-2022]
- Explain the words `final` and `this` with the help of examples. [5, Oct-2023]

Image source: www.javatpoint.com

Arrays: Assigning values: Array of objects

www.hbpatel.in

```
class Student
{
    public int rollno;
    public String name;
    Student (int r, String n) {this.rollno=r; this.name=n;}
}
public class arr04
    {
    public static void main(String args[])
        {
            Student s[]= new Student[3];
            s[0] = new Student(10,"Pradip");
            s[1] = new Student(20,"Sanjay");
            s[2] = new Student(30,"Vijay");

            for (int i=0; i<s.length; ++i)
            {
                System.out.println("Rollno " + i + " = " + s[i].rollno + "\tName " + i + " = " + s[i].name);
            }
        }
    }
```

OUTPUT

```
Rollno 0 = 10    Name 0 = Pradip
Rollno 1 = 20    Name 1 = Sanjay
Rollno 2 = 30    Name 2 = Vijay
```

Code: https://github.com/hbpatel1976/Java/blob/main/arr04.java

Arrays: 2-D Initialization

www.hbpatel.in

```
public class arr05
    {
    public static void main(String args[])
        {
            int matrix[][] = {{10,20,30,40},{50,60,70,80},{90,100,110,120}};
            for(int i=0; i<matrix.length; ++i)
            {
                for(int j=0; j<matrix[0].length; ++j)
                {
                    System.out.print(matrix[i][j]+"\t");
                }
                System.out.println();
            }
        }
    }
```

OUTPUT

```
10      20      30      40
50      60      70      80
90      100     110     120
```

Code: https://github.com/hbpatel1976/Java/blob/main/arr05.java

Arrays: Passing array to a method

www.hbpatel.in

```
public class arr06
    {
    public static void main(String args[])
        {
            int matrix[] = {10,20,30,40};
            int ans = sum(matrix);
            System.out.println(ans);
        }
    public static int sum(int[] arr)
        {
            int sum=0;
            for(int i=0; i<arr.length; ++i)sum+=arr[i];
            return sum;
        }
    }
```

OUTPUT

```
100
```

Code: https://github.com/hbpatel1976/Java/blob/main/arr06.java

Arrays: Returning array from a method

www.hbpatel.in

```
public class arr07
    {
    public static void main(String args[])
        {
            int matrix1[] = {10,20,30,40};
            int matrix2[] = {50,60,70,80};
            int matrix3[];
            matrix3 = sum(matrix1, matrix2);
            for(int i=0; i<matrix1.length; ++i)System.out.println(matrix3[i]);
        }
    public static int [] sum(int[] arr1, int[] arr2)
        {
            int add[] = new int[arr1.length];
            for(int i=0; i<arr1.length; ++i)add[i]=arr1[i]+arr2[i];
            return add;
        }
    }
```

OUTPUT

```
60
80
100
120
```

Code: https://github.com/hbpatel1976/Java/blob/main/arr07.java

Arrays: Cloning an array

www.hbpatel.in

```
public class arr08
    {
    public static void main(String args[])
        {
            int Matrix[] = {10,20,30,40};
            int copiedMatrix [] = Matrix.clone();
            for(int i=0; i<Matrix.length; ++i)System.out.println(copiedMatrix[i]);
        }
    }
```

OUTPUT

```
10
20
30
40
```

Code: https://github.com/hbpatel1976/Java/blob/main/arr08.java

Arrays

www.hbpatel.in

```
class NumberSorting
{
public static void main(String args[])
          {
          int number[] = { 55,40,80,65,71};
          int n=number.length,i,j;
          System.out.println("Given List : ");
          for (i=0; i<n; ++i) System.out.print(" "+number[i]);
          System.out.println("\n");
          for(i=0; i<n; ++i)
                    {for(j=i+1; j<n; ++j)
                              {if(number[i]<number[j])
                                        {int temp=number[i];
                                        number[i]=number[j];
                                        number[j]=temp;
                                        }
                              }
                    }
          System.out.println("Sorted List:");
          for(i=0; i<n; ++i) System.out.print("  "+number[i]);
          }
}
```

OUTPUT

Given List :
55 40 80 65 71

Sorted List:
80 71 65 55 40

Code: https://github.com/hbpatel1976/Java/blob/main/NumberSorting.java

2-D Arrays

www.hbpatel.in

```
class MulTable
{
final static int Rows=20;
final static int  Columns=20;
public static void main(String args[])
          {
          int product[][]= new int[Rows][Columns];
          System.out.println("MULTIPLICATION TABLE");
          System.out.println(" ");
          int i,j;
          for(i=10; i<Rows; ++i)
                    {
                    for(j=10; j<Columns; ++j)
                              {
                              product[i][j]=i*j;
                              System.out.print("   "+product[i][j]);
                              }
                    System.out.println(" ");
                    }
          }
}
```

OUTPUT

MULTIPLICATION TABLE

100	110	120	130	140	150	160	170	180	190
110	121	132	143	154	165	176	187	198	209
120	132	144	156	168	180	192	204	216	228
130	143	156	169	182	195	208	221	234	247
140	154	168	182	196	210	224	238	252	266
150	165	180	195	210	225	240	255	270	285
160	176	192	208	224	240	256	272	288	304
170	187	204	221	238	255	272	289	306	323
180	198	216	234	252	270	288	306	324	342
190	209	228	247	266	285	304	323	342	361

Code: https://github.com/hbpatel1976/Java/blob/main/MulTable.java

Vectors

Java does not support the concept of variable arguments to a function. This feature can be achieved in Java through the use of **Vector** class contained in **java.util** package. This class can be used to create a generic dynamic array known as *vector* that can hold *objects of any type* and *any number*. The object do not have to be homogenous. Arrays can be easily implemented as vectors. Vectors are created like arrays as follows.

```
Vector intVect = new Vector();    //declaring without size
Vector list = new Vector(3);      // declaring with size
```

Vectors

Vectors possess number of advantages over arrays.
It is convenient to use vector to store objects.
A vector can be used to store a list of objects that may vary in size.
We can add or delete objects from list as and when required.

A major constraint in using vectors is that we can not directly store simple datatype in a vector; we can only store objects. Therefore, we need to convert simple types of objects. This can be done using the *wrapper classes*.

Vectors

Method call	Task performed
List.addElement(item)	Adds the item specified to the list at the end.
List.elementAt(10)	Gives the name of the 10th object
List.size()	Gives the number of objects present
List.removeElementAt(n)	Removes the item stored in nth position
List.removeAllElement()	Removes all the elements in the list
List.insertElementAt(item,n)	Inserts the item at nth position

Vectors

www.hbpatel.in

```
import java.util.*;
class LanguageVector
        {
        public static void main(String args[])
                {Vector list = new Vector();
                int length = args.length;
                System.out.println(length);
                for(int i=0; i<length; ++i)
                        {list.addElement(args[i]);}
                list.insertElementAt("COBOL",2);
                int size = list.size();
                String listArray[] = new String[size];
                list.copyInto(listArray);
                System.out.println("List of Languages");
                for(int i=0; i<size; ++i)
                        {System.out.println(listArray[i]);}
                }
        }
```

```
C:\Users\VSITR>java LanguageVector Ada BASIC CPP ForTran Java
5
List of Languages
Ada
BASIC
COBOL
CPP
ForTran
Java

C:\Users\VSITR>
```

Code: https://github.com/hbpatel1976/Java/blob/main/LanguageVector.java

Java Programming: Modules

www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

Object & Classes

www.hbpatel.in

```
class Rectangle
        {
        int l,w;
        void getdata(int x,int y)
          {
          l=x;
          w=y;
          }
        int area()
          {
          int a=l*w;
          return (a);
          }
        }
```

```
class RectArea
        {
        public static void main(String args[])
                {
                int a1,a2;
                Rectangle r1=new Rectangle();
                Rectangle r2=new Rectangle();
                r1.l=15;
                r1.w=10;
                a1=r1.l*r1.w;
                r2.getdata(20,12);
                a2=r2.area();
                System.out.println("Area 1 = "+a1);
                System.out.println("Area 2 = "+a2);
                }
        }
```

OUTPUT

Area 1 = 150

Area 2 = 240

Code: https://github.com/hbpatel1976/Java/blob/main/RectArea.java

Types of Constructors

A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes

There are three types of constructor in java.

- Default Constructor
- No-Argument constructor
- parametrized constructor

Explain the type of constructor with the help of example. [5, Oct-2023]

Default Constructors


```
public class defaultConstructor
{
    public static void main(String args[])
    {
        defaultConstructor obj = new defaultConstructor();
    }
}
```

No Argument Constructor


```
public class noargConstructor
{
    public noargConstructor()
    {
        System.out.println("This is no-argument constructor");
    }
    public static void main(String args[])
    {
        noargConstructor obj = new noargConstructor();
    }
}
```

Parametrized Constructor


```
public class parametrizedConstructor
{
    int data;
    public parametrizedConstructor()
    {
        data = -1;
        System.out.println("This is no-argument constructor");
    }
    public parametrizedConstructor(int parameter)
    {
        data = parameter;
        System.out.println("This is parametrized constructor");
    }
    void show()
    {
        System.out.println(data);
    }
    public static void main(String args[])
    {
        parametrizedConstructor obj1 = new parametrizedConstructor();
        parametrizedConstructor obj2 = new parametrizedConstructor(5);
        obj1.show();
        obj2.show();
    }
}
```

OUTPUT

```
This is no-argument constructor
This is parametrized constructor
-1
5
```

Constructor

www.hbpatel.in

A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes

```
class Rectangle
{
int l,w;
Rectangle(int x, int y)
          {
          l=x;
          w=y;
          }
int rectArea()
          {
          return l*w;
          }
}
```

```
class RectangleArea
          {
          public static void main(String args[])
                    {
                    Rectangle rect1 = new Rectangle(15,10);
                    int area1 = react1.rectArea();
                    System.out.println("Area1 = " + area1);
                    }
          }
```

OUTPUT

Area 1 = 150

- Explain constructor in Java along with its rules and types. Also write down the program to demonstrate the concept of constructor overloading. [5, May-2023] [5, Nov-2023]
- Explain constructor in Java and its rule with program. [5, Jun-2022]
- Explain the type of constructor with the help of example. [5, 2023]

Code: https://github.com/hbpatel1976/Java/blob/main/RectangleArea.java

Constructor Overloading

www.hbpatel.in

```
public class learnConstructor
{    int data1, data2;
     public learnConstructor()
     {    System.out.println("This is a constructor without any argument");
          data1=-1; data2=-1;
     }
     public learnConstructor(int a)
     {    System.out.println("This is a constructor with one any argument");
          data1=a; data2=-1;
     }
     public learnConstructor(int a, int b)
     {    System.out.println("This is a constructor with two arguments");
          data1=a; data2=b;
     }
     void show()
     {    System.out.println("Data 1 = " + data1 + " Data 2 = " + data2);
     }
     public static void main(String args[])
     {    learnConstructor object1 = new learnConstructor();
          learnConstructor object2 = new learnConstructor(5);
          learnConstructor object3 = new learnConstructor(5,10);
          object1.show();
          object2.show();
          object3.show();
     }
}
```

OUTPUT

```
This is a constructor without any argument
This is a constructor with one any argument
This is a constructor with two arguments
Data 1 = -1 Data 2 = -1
Data 1 = 5 Data 2 = -1
Data 1 = 5 Data 2 = 10
```

Code: https://github.com/hbpatel1976/Java/blob/main/learnConstructor.java

Built-in Class: String

www.hbpatel.in

```
public class testingStringClass
{
public static void main(String args[])
{
1. String college = "Vidush Somany Institute of Technology and Research";
2. System.out.println ("College Name : " + college);
3. System.out.println ("College Name Length : " + college.length());
4. System.out.println ("Character at 7th position : " + college.charAt(7));
5. System.out.println ("Substring from 7th position : " + college.substring(7));
6. System.out.println ("Substring between 7th & 13th position : " + college.substring(7, 13));
7. String university = " Kadi Sarva Vishwavidyalaya";
8. System.out.println ("Combined / Concatenated : " + college.concat(university));
9. System.out.println ("College Name : " + college);
10.System.out.println ("University Name : " + university);
11.System.out.println("First occurrence of the word Institute : " + college.indexOf("Institute"));
12.System.out.println("First occurrence of the character a after 15: " + college.indexOf('a',15));
13.System.out.println("String Equality : " + college.equals(university));
14.System.out.println("String Equality : " + college.equals(college));
15.System.out.println("College name in lower Case : " + college.toLowerCase());
16.System.out.println("College name in UPPER Case : " + college.toUpperCase());
17.System.out.println ("College Name : " + college);
18.System.out.println("Replace a with x : " + college.replace('a','x'));
}
}
```

- Explain any two methods of String class and String buffer class of Java with suitable example. [5, Nov-2022]
- What is String class? Explain String class method with example. [5, Jun-2022]
- Compare String with StringBuffer. Also, write a program to count occurrence of a character in a string. [5, Oct-2023]

Code: https://github.com/hbpatel1976/Java/blob/main/testingStringClass.java

Built-in Class: String

www.hbpatel.in

OUTPUT

```
1. -
2. College Name : Vidush Somany Institute of Technology and Research
3. College Name Length : 50
4. Character at 7th position : S
5. Substring from 7th position : Somany Institute of Technology and Research
6. Substring between 7th & 13th position : Somany
7. -
8. Combined / Concatenated : Vidush Somany Institute of Technology and Research Kadi Sarva Vishwavidyalaya
9. College Name : Vidush Somany Institute of Technology and Research
10.University Name :  Kadi Sarva Vishwavidyalaya
11.First occurrence of the word Institute : 14
12.First occurrence of the character a after 15: 38
13.String Equality : false
14.String Equality : true
15.College name in lower Case : vidush somany institute of technology and research
16.College name in UPPER Case : VIDUSH SOMANY INSTITUTE OF TECHNOLOGY AND RESEARCH
17.College Name : Vidush Somany Institute of Technology and Research
18.Replace a with x : Vidush Somxny Institute of Technology xnd Resexrch
```

Code: https://github.com/hbpatel1976/Java/blob/main/testingStringClass.java

Guess the output !

www.hbpatel.in

```
public class Code
{
    public static void main(String args[])
    {
      StringBuffer str1 = new StringBuffer("ldrp");
      StringBuffer str2 = str1;
      str1.append("itr");
      System.out.println(str1 + " " + str2 + " " + (str1==str2));
    }
}
```

OUTPUT

```
ldrpitr ldrpitr true
```

Give the output of the given code.. [5, Nov-2022]

Code: https://github.com/hbpatel1976/Java/blob/main/testingStringClass.java

Built-in Class: StringBuffer

www.hbpatel.in

```
public class learnStringBuffer
{
    public static void main(String[] args)
    {
     1. StringBuffer obj = new StringBuffer();
     2. obj.append("Vidush");
     3. obj.append(" ");
     4. obj.append("Somany");
     5. System.out.println(obj);
     6. String message = obj.toString();
     7. System.out.println(message);
     8. obj.insert(7, "Kadi ");
     9. System.out.println(obj);
     10.obj.replace(7,11,"Gngr");
     11.System.out.println(obj);
     12.obj.delete(7,12);
     13.System.out.println(obj);
     14.obj.reverse();
     15.System.out.println(obj);
    }
}
```

OUTPUT

```
 5. Vidush Somany
 7. Vidush Somany
 9. Vidush Kadi Somany
11. Vidush Gngr Somany
13. Vidush Somany
15. ynamoS hsudiV
```

Code: https://github.com/hbpatel1976/Java/blob/main/learnStringBuffer.java

Final Variables and Methods

All methods can be overridden by default in subclasses. If we wish to **prevent** the subclasses from overriding the members of the superclass, we can declare them as final using the keyword final as a modifier. For example,

```
final int SIZE=100;
final void Display() { . . . . . }
```

Making a method final ensures that the functionality defined in this method will never be altered any way. Similarly, the value of final variable can never be changed. Final variables, behave like class variables and they do not take any space on individual objects of the class.

- Explain the use of `final` keyword in Java with proper example. [2.5, May-2023] [2.5, Nov-2023]
- Explain following keywords with example. (a) `this` (b) `final` (c) `super`. [5, Jun-2022]
- Explain the words `final` and `this` with the help of examples. [5, Oct-2023]

Final Variables and Methods

Sometimes we may like to prevent a class being further subclassed for security reasons. A class that can not be subclassed is called a final class. This is achieved syntactically as follow,

```
final class Aclass {. . . . . }
final class Bclass extends Cclass {. . . . . }
```

Final Variables

www.hbpatel.in

```
class Vehicle
{
        final int speedLimit = 90;
        void run()
        {
                speedLimit = 120;
        }
}
public class FinalVariableTesting
{
        public static void main(String args[])
        {
                Vehicle desire = new Vehicle();
                desire.run();
        }
}
```

ERROR

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The final field Vehicle.speedLimit **cannot** be assigned

at Vehicle.run(FinalVariableTesting.java:6)
at FinalVariableTesting.main(FinalVariableTesting.java:14)

Code: https://github.com/hbpatel1976/Java/blob/main/FinalVariableTesting.java

Final Method

www.hbpatel.in

```
class Car
{
        final void run()
        {
                System.out.println("Car is running");
        }
}
class Honda extends Car
{
        void run()
        {
                System.out.println("Honda Car is running");
        }
}
public class FinalMethodTesting
{
        public static void main(String args[])
        {
                Honda city = new Honda();
                city.run();
        }
}
```

ERROR

Exception in thread "main" java.lang.VerifyError: class Honda overrides final method Car.run()

Code: https://github.com/hbpatel1976/Java/blob/main/FinalMethodTesting.java

Final Class

www.hbpatel.in

```
final class car
{
        void run()
        {
                System.out.println("Car is running");
        }
}
class honda extends car
{
        void run()
        {
                System.out.println("Honda Car is running");
        }
}
public class FinalClassTesting
{
        public static void main(String args[])
        {
                honda city = new honda();
                city.run();
        }
}
```

ERROR

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The type honda cannot subclass the final class car

Code: https://github.com/hbpatel1976/Java/blob/main/FinalClassTesting.java

Finalizer Methods

Java supports a concept called finalization, which is similar to destructors in C++. Java runtime is an automatic garbage collecting system. It automatically frees up the memory resources used by the objects. But objects may hold other non-object resources such as file descriptor or window system fonts. The garbage collector can not free these resources. In order to free these resources, we must use a finalizer method.

The finalizer method is simply **`finalize()`** and can be added to any class. Java calls this method whenever it is about to bring back the space for the object. The **finalize** method should explicitly define the tasks to be performed.

Finalizer Methods

www.hbpatel.in

```
public class FinalizeMethodTesting
{
public static void main (String args[])
            {
            FinalizeMethodTesting d =  new FinalizeMethodTesting();
            try
                        {
                        System.out.println("Testing 1 2 3 ...");
                        System.out.println("Testing A B C ...");
                        System.out.println("Deleting......");
                        d.finalize();
                        System.out.println("Deleted......");
                        }
            catch(Exception e)
                        {
                        System.out.println("Exception Printed: "+e);
                        }
            }
protected void finalize()
            {
            System.out.println("Collecting and trashing all the garbage....");
            }
}
```

OUTPUT

```
Testing 1 2 3 ...
Testing A B C ...
Deleting......
Collecting and trashing all the garbage....
Deleted......
```

Code: https://github.com/hbpatel1976/Java/blob/main/FinalizeMethodTesting.java

Access Modifiers

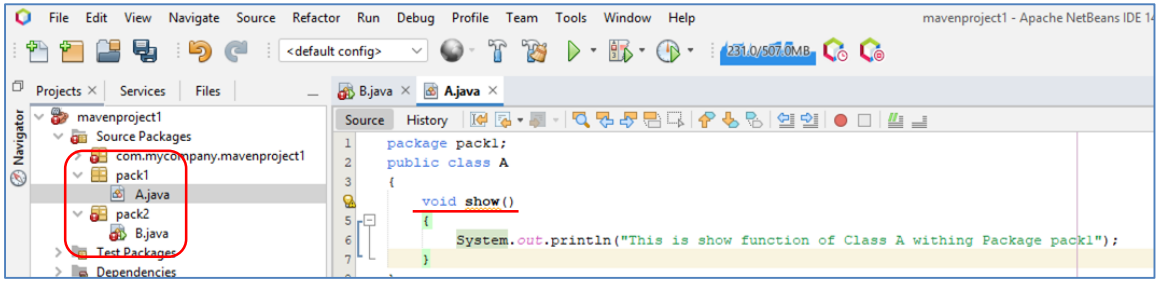
www.hbpatel.in

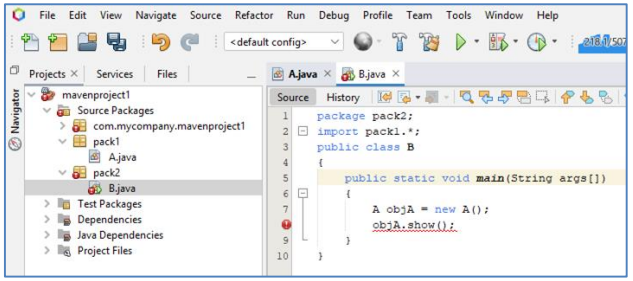
Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

- Discuss various access modifiers available in Java? How access modifier affects the visibility of a member in different access locations? Explain with suitable example. [5, May-2023]
- Explain use of protected and default access specifiers of Java with suitable example. [5, Nov-2022]
- Explain visibility modifiers with their scope in packages. [5, Jun-2022]
- Explain all access modifiers and their visibility as class members. [5, Oct-2023]

Access Modifiers

www.hbpatel.in

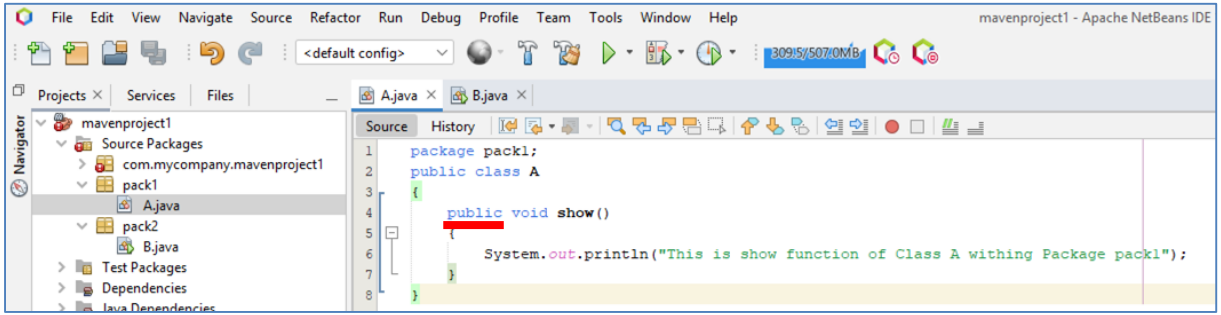


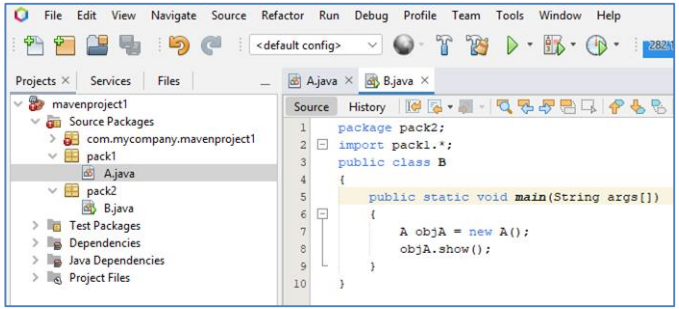



```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
Exception in thread "main" java.lang.RuntimeException: Uncompilable code - show() is not public in pack1.A; cannot be accessed from outside package
        at pack2.B.main(B.java:1)
Command execution failed.
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)
    at org.apache.commons.exec.DefaultExecutor.executeInternal (DefaultExecutor.java:404)
```

Access Modifiers

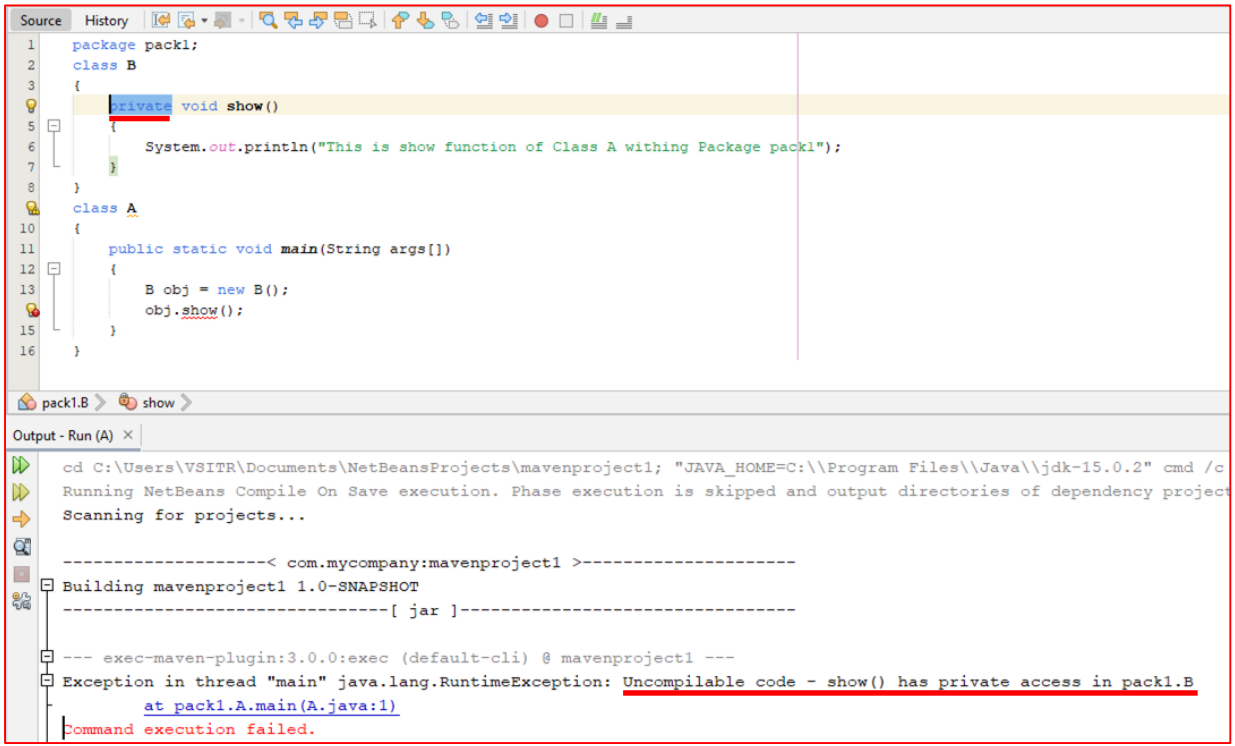
www.hbpatel.in





Access Modifiers

www.hbpatel.in



Access Modifiers

www.hbpatel.in

```
1  package pack1;
2  class B
3  {
4      public void show()
5      {
6          System.out.println("This is show function of Class A withing Package pack1");
7      }
8  }
9  class A
10 {
11     public static void main(String args[])
12     {
13         B obj = new B();
14         obj.show();
15     }
16 }
```

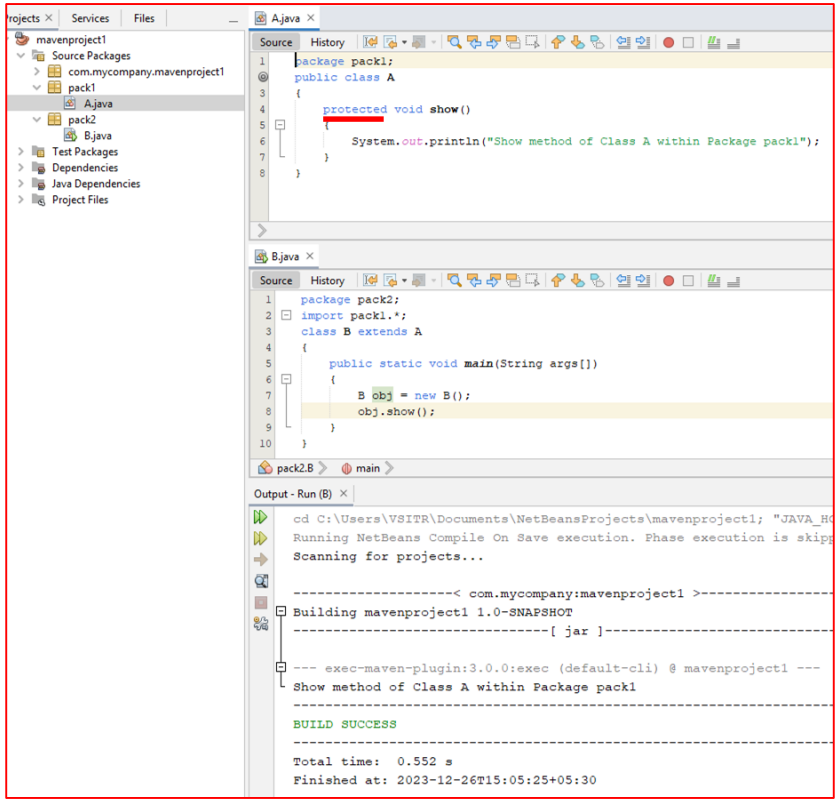
pack1.B > show

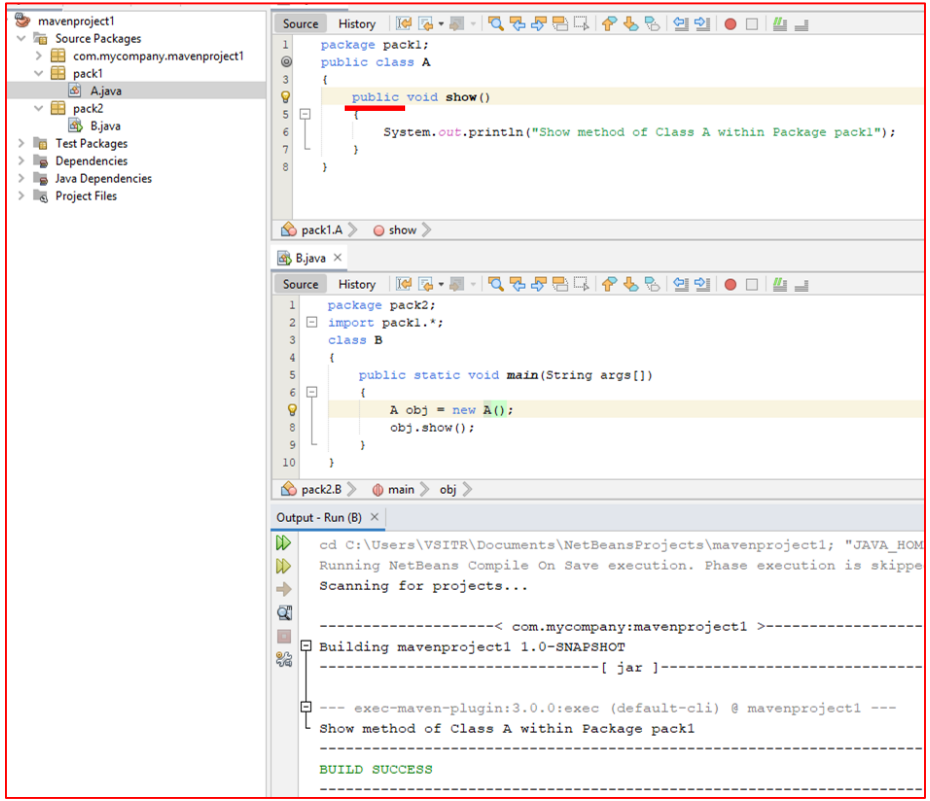
output - Run (A)

```
cd C:\Users\VSITR\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C:\\Prog
Running NetBeans Compile On Save execution. Phase execution is skipped and outp
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >--------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
This is show function of Class A withing Package pack1
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Access Modifiers

www.hbpatel.in





Java Programming: Modules

www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

Inheritance

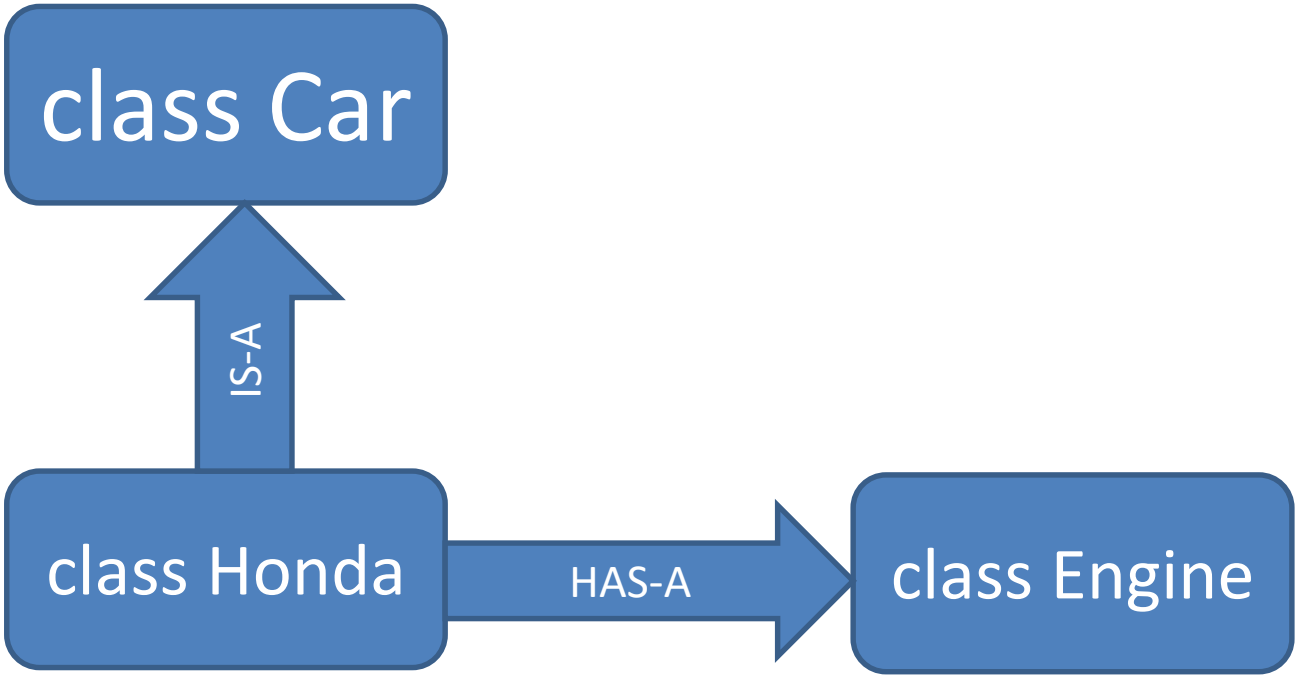
Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOP.

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can **reuse** methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the IS-A relationship (shown in next slide) which is also known as a parent-child relationship.

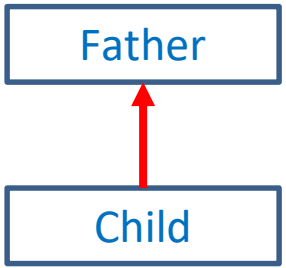
- List OOP characteristics and describe Inheritance [5, Oct-2023]

Inheritance

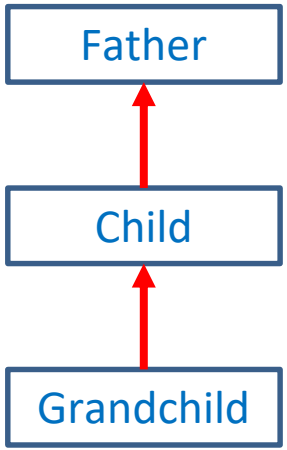


Inheritance

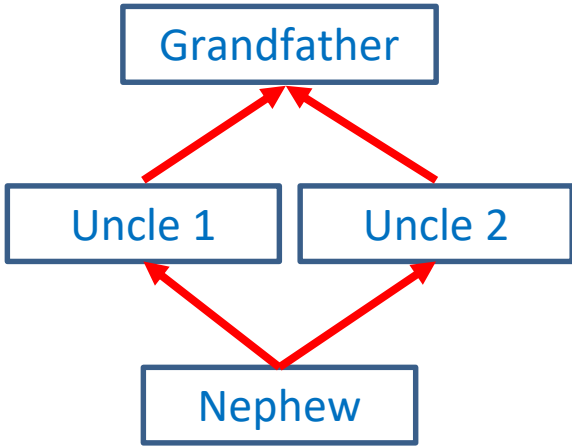
www.hbpatel.in



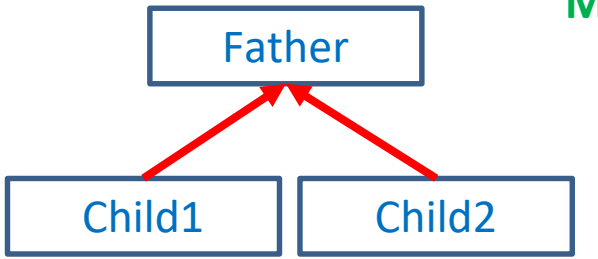
Single



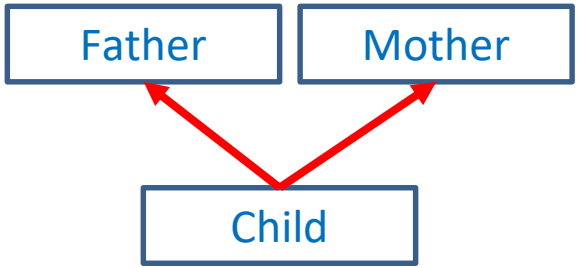
Multilevel



Hybrid



Hierarchical

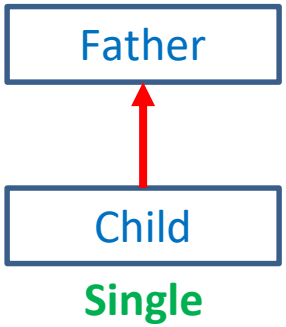


Multiple

What is inheritance. List out its types and explain any one with proper example. [5, May-2023]

Inheritance

www.hbpatel.in



Single

singleInheritance.java

Source | History

```
class Father
{
    void showF() {System.out.println("This is show function in Father class");}
}
class Child extends Father
{
    void showC() {System.out.println("This is show function in Child class");}
}
public class singleInheritance
{
    public static void main(String args[])
    {
        Child s = new Child();
        s.showC();
        s.showF();
    }
}
```

com.mycompany.mavenproject1.Child > showC

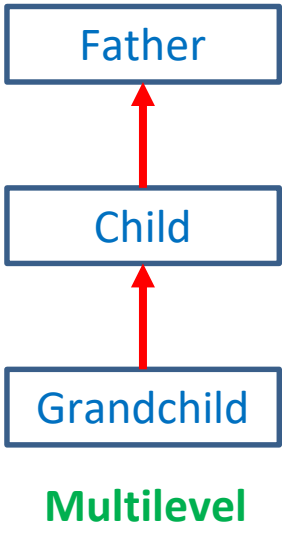
Output - Run (singleInheritance)

```
cd C:\Users\VSITR\Documents\NetBeansProjects\mavenproject1; "JAVA_HOME=C
Running NetBeans Compile On Save execution. Phase execution is skipped a
Scanning for projects...

--------------------< com.mycompany:mavenproject1 >--------------------
Building mavenproject1 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
This is show function in Child class
This is show function in Father class
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Code: https://github.com/hbpatel1976/Java/blob/main/singleInheritance.java

Inheritance

www.hbpatel.in



Multilevel

```
class Father
{
    void showF() {System.out.println("This is show function in Father class");}
}
class Child extends Father
{
    void showC() {System.out.println("This is show function in Child class");}
}
class grandChild extends Child
{
    void showG() {System.out.println("This is show function in Grand Child class");}
}
public class multilevelInheritance
{
    public static void main(String args[])
    {
        grandChild g = new grandChild();
        g.showG();
        g.showC();
        g.showF();
    }
}
```

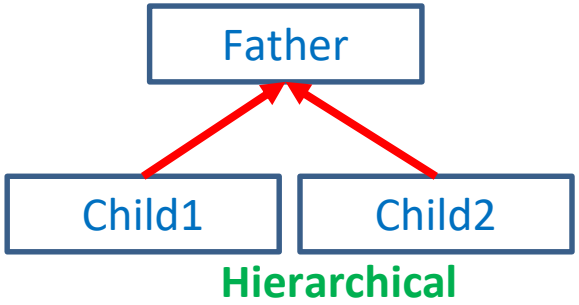
```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ mavenproject1 ---
This is show function in Grand Child class
This is show function in Child class
This is show function in Father class
------------------------------------------------------------------------
```

- Write a Java program of multilevel inheritance with suitable example. [5, Nov-2022]
- What is inheritance? Explain multilevel inheritance with program. [5, Jun-2022]

Code: https://github.com/hbpatel1976/Java/blob/main/multilevelInheritance.java

Inheritance

www.hbpatel.in



Hierarchical

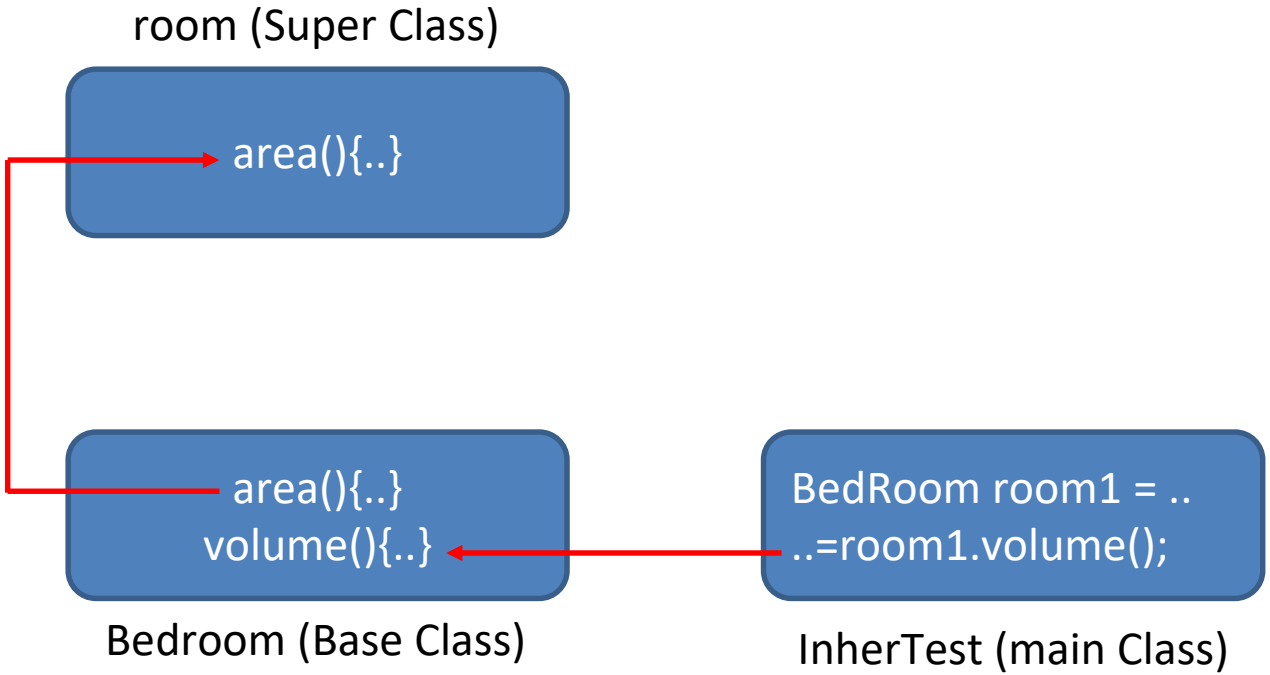
```
Source  History
4   class Father
5   {
6       void showF() {System.out.println("This is show function in Father class");}
7   }
8   class Child1 extends Father
9   {
10      void showC1() {System.out.println("This is show function in Child 1 class");}
11  }
12  class Child2 extends Father
13  {
14      void showC2() {System.out.println("This is show function in Child 2 class");}
15  }
16  public class heirarchicalInheritance
17  {
18   public static void main(String args[])
19   {
20       Child1 c1 = new Child1();
21       c1.showC1();
22       c1.showF();
23       Child2 c2 = new Child2();
24       c2.showC2();
25       c2.showF();
26   }
27  }
```

```
This is show function in Child 1 class
This is show function in Father class
This is show function in Child 2 class
This is show function in Father class
```

Code: https://github.com/hbpatel1976/Java/blob/main/heirarchicalInheritance.java

Inheritance

www.hbpatel.in



The "**`super`**" keyword refers to superclass (parent) objects. It is used to call superclass methods, and to access the superclass constructor. The most common use of the super keyword is to eliminate the confusion between super classes and subclasses that have methods with the same name.

- Explain the keyword with proper example. (b) `super` [2.5, May-2023, 2.5, Nov-2022]
- Explain following keywords with example. (a) `this` (b) `final` (c) `super`. [5, Jun-2022]

Inheritance

www.hbpatel.in

```
class room
{
int l,w;
room(int x, int y) {l=x; w=y; }
int area() { return l*w; }
}

class BedRoom extends room
{
int h;
BedRoom(int x, int y, int z)
          {
          super(x,y);
          h=z;
          }
int volume()
          {
          return l*w*h;
          }
}
```

```
class InherTest
{
public static void main(String args[])
          {
          BedRoom room1 = new BedRoom(14,12,10);
          int area1 = room1.area();
          int volume1=room1.volume();
          System.out.println("Area 1 = "+area1);
          System.out.println("Volume 1 = "+volume1);
          }
}
```

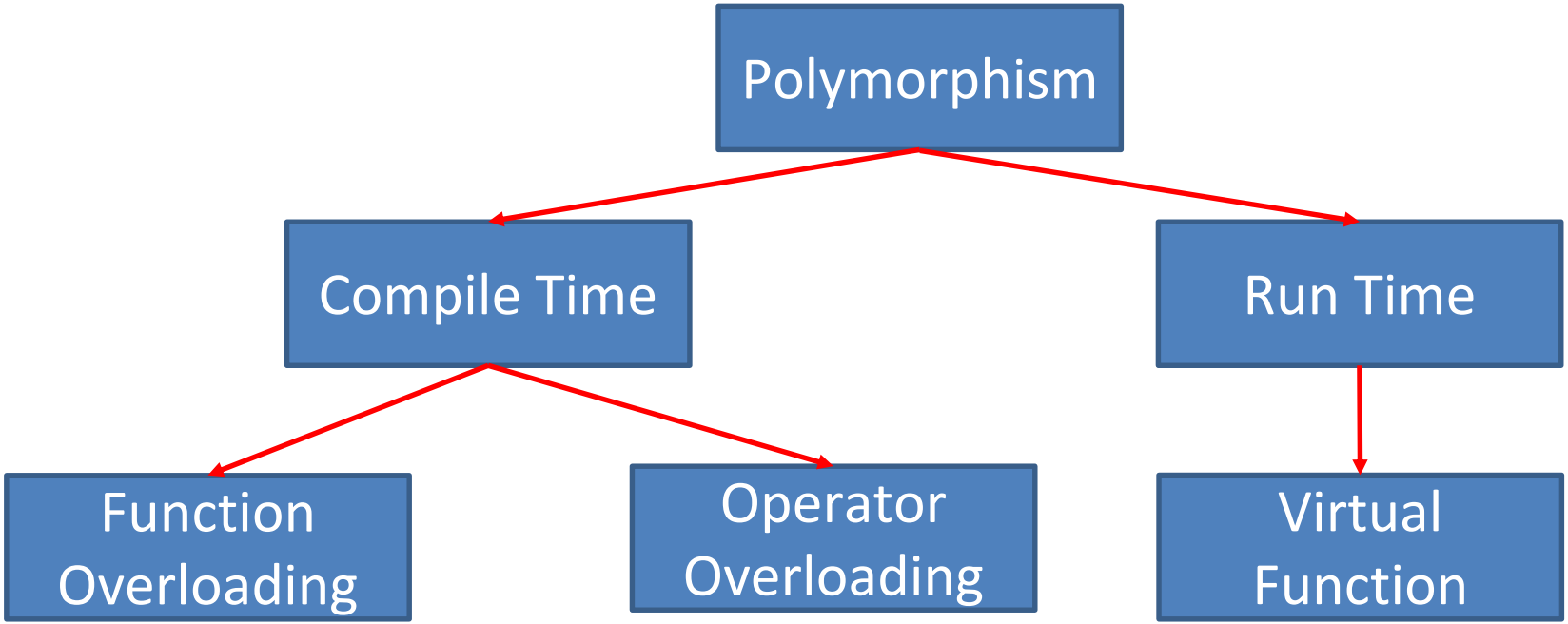
OUTPUT

Area 1 = 168

Volume 1 = 1680

Code: https://github.com/hbpatel1976/Java/blob/main/InherTest.java

Polymorphism



Method Overloading

Method Overloading is a Compile time polymorphism. In method overloading, more than one method shares the same method name with a different signature in the class. In method overloading, the return type can or can not be the same, but we have to change the parameter because, in Java, we can not achieve method overloading by changing only the return type of the method.

Source: https://www.geeksforgeeks.org/

- Explain method overloading with program [5, Jun-2022]

Method Overloading

www.hbpatel.in

```
public class methodOverloading
{
    static void sum() {System.out.println("No data for summation");}
    static int sum(int p) {return p;}
    static int sum(int p, int q) {return p+q;}
    static int sum(int p, int q, int r) {return p+q+r;}
    public static void main(String args[])
    {
        sum();
        System.out.println(sum(10));
        System.out.println(sum(10,20));
        System.out.println(sum(10,20,30));
    }
}
```

OUTPUT
No data for summation
10
30
60

Explain: Arguments and Parameters of a function. [2.5, Oct-2023]

https://github.com/hbpatel1976/Java/blob/main/methodOverloading.java

Method Overriding

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

It is used to provide the **specific implementation** of a method which is already provided by its superclass. Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

The method must have the same name as in the parent class

The method must have the same parameter as in the parent class.

There must be an IS-A relationship (inheritance).

- What is polymorphism? Explain method overriding with program. [5, Jun-2022]

Method Overriding

www.hbpatel.in

```
class Super
{
int x;
Super (int x ) { this.x = x; }
void display()
          {
          System.out.println("Super x = " +x);}
          }

class Sub extends Super
{
int y;
Sub(int x, int y){super(x);this.y=y;}
void display()
          {
          System.out.println("Super x = "+x);
          System.out.println("Sub y = "+y);
          }
}
```

```
class OverRideTest
{
public static void main(String args[])
          {
          Sub s1 = new Sub(100,200);
          s1.display();
          }
}
```

OUTPUT

Super x = 100
Sub y = 200

Code: https://github.com/hbpatel1976/Java/blob/main/OverRideTest.java

Method Overriding

www.hbpatel.in

```
class Car
{
    void show() {System.out.println("This is a show function of Car class");}
}
class Maruti extends Car
{
    void show() {System.out.println("This is a show function of Maruti class");}
}
class Dezire extends Car
{
    void show() {System.out.println("This is a show function of Dezire class");}
}
public class methodOverriding
{
    public static void main(String args[])
    {
        Car c = new Car();
        Maruti m = new Maruti();
        Dezire d = new Dezire();

        c.show();
        m.show();
        d.show();
    }
}
```

OUTPUT

```
This is a show function of Car class
This is a show function of Maruti class
This is a show function of Dezire class
```

Code: https://github.com/hbpatel1976/Java/blob/main/methodOverriding.java

Overloading Vs. Overriding

www.hbpatel.in

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.
Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.
In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

Differentiate method overloading and method overriding with the help of example.
[5, May-2023] [5, Nov-2022]

Source: https://www.javatpoint.com/

Abstract methods and classes

This is just opposite to **final**. We can indicate that a method must always be redefined in a subclass, thus making overriding compulsory. This is done using keyword **abstract** as below.

```
abstract class Shape
                {
                . . . . . .
                abstract void draw();
                . . . . . .
                }
```

While using abstract classes, we must satisfy the following conditions.

1. We can not use abstract classes to instantiate object directly.

For example, `Shape s = new Shape()` // illegal because Shape is an abstract class.

2. The abstract method in abstract class must be defined in its subclass.
3. We can not declare abstract constructor or abstract static methods.

- Compare abstract & interface in Java. [5, May-2023] [5, Jun-2022] [5, Oct-2023]
- Write a Java program to demonstrate abstract class with suitable example. [5, Nov-2022]

Abstract methods / class

Example # 1

www.hbpatel.in

```
abstract class bike
{
        abstract void run();
}
class splendor extends bike
{
        void run()
        {
                System.out.println("Splendor motorcycle is running");
        }
}
public class AbstractMethodTesting
{
        public static void main(String args[])
        {
                splendor s = new splendor ();
                s.run();
        }
}
```

OUTPUT

Splendor motorcycle is running

Trying to add a line "`bike b = new bike();`" here would result into an error "*Cannot instantiate the type bike*"

Code: https://github.com/hbpatel1976/Java/blob/main/AbstractMethodTesting.java

Abstract methods / class

Describe abstract class called Shape which has three subclasses say Triangle, Rectangle and Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate their specific object i.e. area() of Triangle subclass should calculate the area of triangle etc. Same for Rectangle and Circle. [5, May-2023]

Abstract methods / class

Example # 2: Assignment

www.hbpatel.in

```
abstract class shape
{
int radius, length, width;
abstract void showArea();
}

class rectangle extends shape
{
rectangle(int a, int b)
          {length=a; width=b;}
void showArea()
          {System.out.println("Area of Rectangle = " + length*width);}
}

class circle extends shape
{
circle(int a){radius=a;}
void showArea()
          {System.out.println("Area of Rectangle = " + 3.14 * radius * radius);}
}
```

```
public class AbstractClassShape
{
public static void main (String args[])
          {
          rectangle r = new rectangle(10, 20);
          circle c = new circle (10);
          r.showArea();
          c.showArea();
          }
}
```

OUTPUT

```
Area of Rectangle = 200
Area of Rectangle = 314.0
```

Code: https://github.com/hbpatel1976/Java/blob/main/AbstractClassShape.java

Wrapper Class

As pointed earlier, vectors can not handle primitive data types like int, float, long, char and double. Primitive data types may be converted into object types by using wrapper classes contained in the **java.lang** package.

Wrapper classes for converting simple types

Simple Type	Wrapper Class
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long

Wrapper Class

Converting primitive numbers to object numbers using constructor method

Constructor calling	Conversion action
Integer IntVal = new Integer(i);	Primitive integer to Integer object
Float FloatVal = new Float(f);	Primitive float to Float object
Double DoubleVal = new Double(d);	Primitive double to Double object
Long LongVal = new Long(l);	Primitive long to Long object

(**Note**:i,f,d and l are primitive data values(constants or variables) denoting int, float, double and long datatypes.

Wrapper Class

Converting object numbers to primitive numbers using typeValue() method

Method calling	Conversion Action
int i = IntVal.intValue();	Object to primitive integer
float f= FloatVal.floatValue();	Object to primitive float
long l = LongVal.longValue().	Object to primitive long
double d=DoubleVal.doubleValue();	Object to primitive double

Wrapper Class

Converting numbers to Strings using String() methods

Method calling	Conversion Action
str = Integer.toString(i);	Primitive integer to string
str = Float.toString(f);	Primitive float to string
str = Double.toString(d);	Primitive double to string
str = Long.toString(l);	Primitive long to string

Wrapper Class

Converting String objects to Numeric Objects Using the Static Method Valueof()

Method calling	Conversion Action
DoubleVal=Double.ValueOf(str);	Converts string to Double object
FloatVal=Float.ValueOf(str);	Converts string to Float object
IntVal=Integer.ValueOf(str);	Converts string to Integer object
LongVal=Long.ValueOf(str);	Converts string to Long object

Wrapper Class

Converting numeric strings to primitive numbers using Parsing Mthods

Method calling	Conversion Action
int i = Integer.parseInt(str);	converts string to primitive integer
long l = Long.parseLong(str);	converts string to primitive long

Wrapper Class

www.hbpatel.in

```
import java.io.*;
class Invest
        {public static void main(String args[])
                {Float principalAmount = new Float(0);
                Float interestRate = new Float(0);
                int numYears=0;
                try             {DataInputStream in = new DataInputStream(System.in);
                                System.out.print("Enter Principal Amount : ");
                                principalAmount = Float.valueOf(in.readLine());
                                System.out.print("Enter Interest Rate : ");
                                String interestString= in.readLine();
                                interestRate=Float.valueOf(interestString);
                                System.out.print("Enter Number of years: ");
                                String yearString= in.readLine();
                                numYears=Integer.parseInt(yearString);
                                }
                catch(IOException e)
                                {System.out.println("IO Error");
                                System.exit(1);
                                }
```

Code: https://github.com/hbpatel1976/Java/blob/main/Invest.java

Wrapper Class


```
                float value=calculateInterest(principalAmount.floatValue(),interestRate.floatValue(),numYears);
                System.out.println("Final Value = " + value);
                }
static float calculateInterest(float p, float r, int n)
                {
                return (p*r*n)/100;
                }
}
```

OUTPUT

```
Enter Principal Amount : 1000
Enter Interest Rate : 12
Enter Number of years: 3
Final Value = 360.0
```

OUTPUT

```
Enter Principal Amount : abcd
IO Error
```

OUTPUT

```
Enter Principal Amount : 1000
Enter Interest Rate : abc
IO Error
```

OUTPUT

```
Enter Principal Amount : 1000
Enter Interest Rate : 12
Enter Number of years: abc
IO Error
```

Interface: Multiple Inheritance

An interface is basically a class. The difference between interface and class is, interface defines only abstract methods and final fields. This means that interfaces do not specify any code to implement these methods and data fields contain only constants. Therefore, it is the responsibility of the class that **implements** an interface to define the code for implementation of these methods.

Syntax: **`interface`** *`InterfaceName`*

```
{
variables declaration;
methods declaration;
}
```

Note that all variables are declared as constants. Methods declaration will contain only a list of methods *without* any body statement.

Interface: Multiple Inheritance

Extending Interfaces:

Like classes, interfaces can also be extended.

Syntax: **`interface`** `name2` **`extends`** `name1`

```
{
body of name2
}
```

Implementing Interfaces:

Interfaces are used as a “superclasses” whose properties are inherited by classes. It is therefore necessay to create a class that inherits the given interface.

Syntax: **`class`** *`classname`* **`implements`** *`interfacename`*

```
{
body of classname
}
```

Interface: Multiple Inheritance

Extending Interfaces:

Like classes, interfaces can also be extended.

Syntax: **`interface`** `name2` **`extends`** `name1`

```
{
body of name2
}
```

Implementing Interfaces:

Interfaces are used as a “superclasses” whose properties are inherited by classes. It is therefore necessay to create a class that inherits the given interface.

Syntax: **`class`** *`classname`* **`implements`** *`interfacename`*

```
{
body of classname
}
```

Abstract Vs. Interface

www.hbpatel.in

Abstract Class	Interface
An abstract class can contain both abstract and non-abstract methods.	Interface contains only abstract methods.
An abstract class can have all four; static, non-static and final, non-final variables.	Only final and static variables are used.
To declare abstract class abstract keywords are used.	The interface can be declared with the interface keyword.
It supports multiple inheritance.	It does not support multiple inheritance.
The keyword ‘extend’ is used to extend an abstract class	The keyword implement is used to implement the interface.
It has class members like private and protected, etc.	It has class members public by default.

Compare abstract & interface in Java. [5, May-2023]

Source: www.byjus.com

Interface (Multiple Inheritance)

Example #1 (implement + implement)

www.hbpatel.in

```
interface interfaceA
{
public void methodA();
}

interface interfaceB
{
public void methodB();
}
```

```
public class MultipleInheritance2
{
        public static void main(String args[])
        {
                classC objc = new classC();
                objc.methodA();
                objc.methodB();
        }
}
```

OUTPUT

```
This is Method A from class C
This is Method B from class C
```

```
class classC implements interfaceA, interfaceB
{
public void methodA() {System.out.println("This is Method A from class C");}
public void methodB() {System.out.println("This is Method B from class C");}
}
```

Code: https://github.com/hbpatel1976/Java/blob/main/MultipleInheritance2.java

Interface (Multiple Inheritance)

Example #2 (extend + implement)

www.hbpatel.in

```
interface Wheeler2
{
        public void show();
}
abstract class Wheeler4
{
        abstract public void show();
}

class myVehicle extends Wheeler4 implements Wheeler2
{
        public void show() {System.out.println("This is my vehicle");}
}
```

```
public class MultipleInheritance4
{
        public static void main(String args[])
        {
                myVehicle m = new myVehicle();
                m.show();
        }
}
```

OUTPUT

This is my vehicle

Code: https://github.com/hbpatel1976/Java/blob/main/MultipleInheritance4.java

Interface (Multiple Inheritance)

Example #3

www.hbpatel.in

```
interface Area
        {
        final static float pi=3.14f;
        float compute(float x, float y);
        }
class R implements Area
        {
        public float compute(float x, float y)
                { return (x*y); }
        }
class C implements Area
        {
        public float compute(float x, float y)
                { return (x*x*pi); }
        }
```

Code: https://github.com/hbpatel1976/Java/blob/main/InterfaceTest.java

Interface (Multiple Inheritance)

Example #3

www.hbpatel.in

```
class InterfaceTest
        {
        public static void main(String args[])
                {
                R rect=new R();
                C cir=new C();
                Area area;
                area=rect;
                System.out.println("Area of Rectangle = "+rect.compute(10,20));
                area=cir;
                System.out.println("Area of Circle = "+cir.compute(10,0));
                }
        }
```

OUTPUT

```
Area of Rectangle = 200.0
Area of Circle = 314.0
```

Code: https://github.com/hbpatel1976/Java/blob/main/InterfaceTest.java

Interface (Multiple Inheritance)

Example #4

www.hbpatel.in

```
interface motorcycle
{
        int speed=50;
        public void distanceCovered();
}
interface cycle
{
        int time=60;
        public void runningTime();
}

class TwoWheeler implements motorcycle, cycle
{
        int avgSpeed=speed;
        int avgTime=time;
        public void distanceCovered()
                {System.out.println("Distance Travelled = "+avgSpeed*avgTime);}
        public void runningTime()
                {System.out.println("Average Running Time = "+avgTime);}
}
```

```
public class MultipleInheritance1
{
        public static void main(String args[])
        {
                TwoWheeler t = new TwoWheeler();
                t.distanceCovered();
                t.runningTime();
        }
}
```

OUTPUT

```
Distance Travelled = 3000
Average Running Time = 60
```

Code: https://github.com/hbpatel1976/Java/blob/main/MultipleInheritance1.java

Interface (Multiple Inheritance)

Example #5 (Assignment)


```
class Student
        {
        int rollNumber;
        void getNumber(int n)
                {rollNumber=n;}
        void putNumber()
                {System.out.println("Roll Number = "+rollNumber);}
        }
class Test extends Student
        {
        float part1,part2;
        void getMarks(float m1, float m2)
                {part1=m1;part2=m2;}
        void putMarks()
                {
                System.out.println("Marks Obtained");
                System.out.println("part1 = "+part1);
                System.out.println("part2 = "+part2);
                }
        }
```

Code: https://github.com/hbpatel1976/Java/blob/main/Hybrid.java

Interface (Multiple Inheritance)

Example #5


```
interface Sports
        {
        float sportWt=6.0F;
        void putWt();
        }
class Results extends Test implements Sports
        {
        float total;
        public void putWt()
                {
                System.out.println("Sport Wt = "+sportWt);
                }
        void display()
                {
                total=part1+part2+sportWt;
                putNumber();
                putMarks();
                putWt();
                System.out.println("Total Score = " +total);
                }
        }
```

Code: https://github.com/hbpatel1976/Java/blob/main/Hybrid.java

Interface (Multiple Inheritance)

Example #5

www.hbpatel.in

```
class Hybrid
        {
        public static void main(String args[])
                {
                Results student1=new Results();
                student1.getNumber(1234);
                student1.getMarks(27.5f,33.0f);
                student1.display();
                }
        }
```

OUTPUT

```
Roll Number = 1234
Marks Obtained
part1 = 27.5
part2 = 33.0
Sport Wt = 6.0
Total Score = 66.5
```

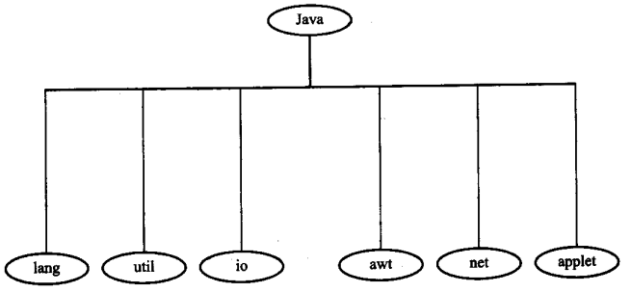
Code: https://github.com/hbpatel1976/Java/blob/main/Hybrid.java

Packages

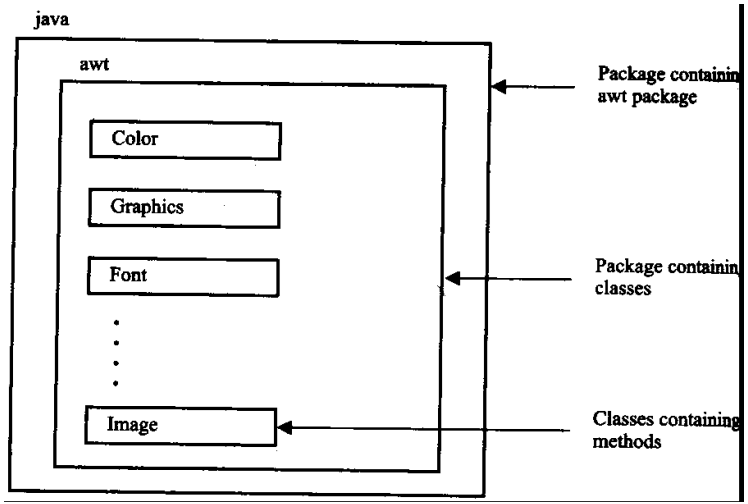
www.hbpatel.in

Package is a concept similar to “class library” in other languages. Packages are Java’s way of grouping a variety of classes and/or interfaces together. Packages act as “container” for classes.

Java’s API Packages



Hierarchical Representation of java.awt package



Naming Conventions

```
double y= java.lang.Math.sqrt(x);
```

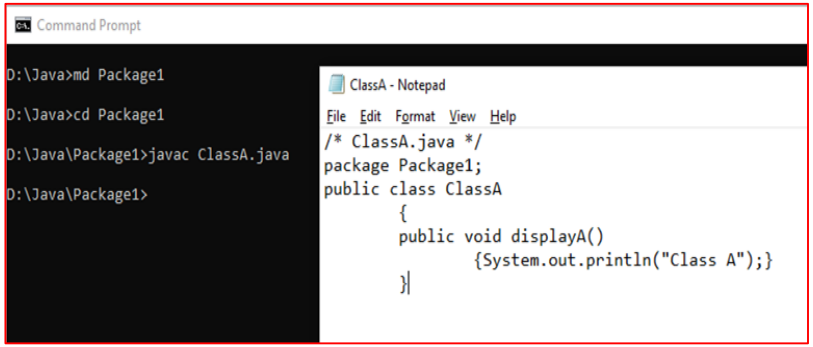
Package Name Class Name Function Name

Creating Packages

www.hbpatel.in

Suppose you are in a directory **D:\Java**. Now, create one sub directory **Package1** under this directory. Now make a file **ClassA.java** as following.

```
/* ClassA.java */
package Package1;
public class ClassA
{
public void displayA()
          {System.out.println("Class A");}
}
```



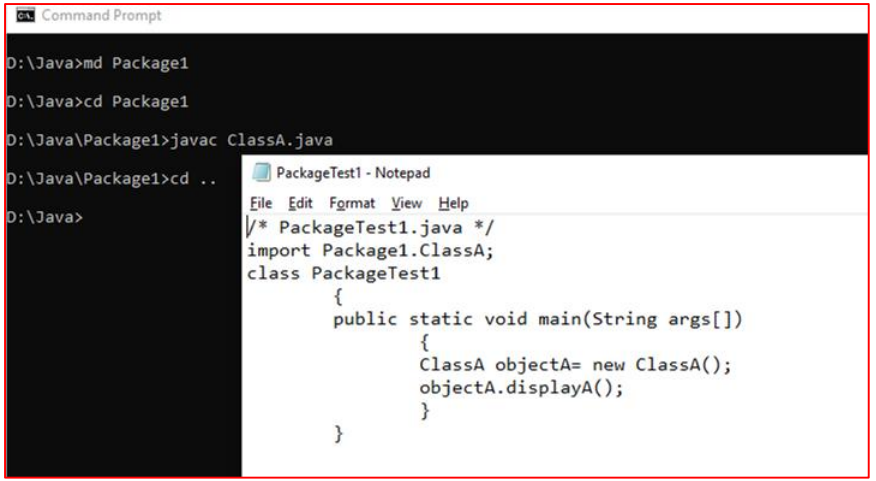
Now, using javac command generate **ClassA.class** in this sub directory (i.e. in package1). Now go to main directory (i.e here D:\Java) and make following file **PackageTest1.java**.

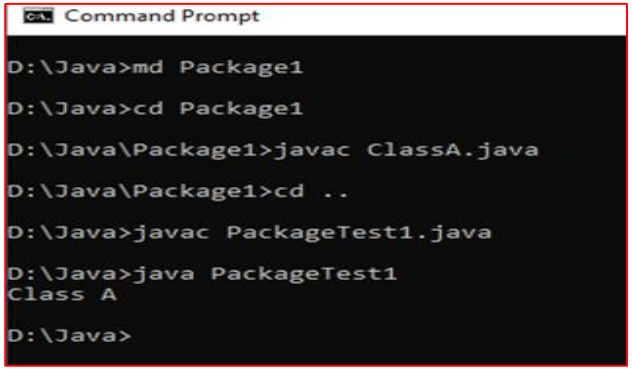
- Write a Java program to implement user defined package. [5, Nov-2022]
- What is Package in Java? Explain step to create package with example. [5, Jun-2022]
- How can you create package in Java? Explain with example. [5, Oct-2023]

Creating Packages

www.hbpatel.in

```
/* PackageTest1.java */
import Package1.ClassA;
class PackageTest1
        {
        public static void main(String args[])
                {
                ClassA objectA= new ClassA();
                objectA.displayA();
                }
        }
```





Java Programming: Modules

www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

Collection Interface in Java

- The Collection interface is a member of the Java Collections Framework.
- It is a part of `java.util` package.
- It is one of the root interfaces of the Collection Hierarchy.
- The Collection interface is not directly implemented by any class. However, it is implemented indirectly via its subtypes or subinterfaces like List, Queue, Set and Tree.

What is collection in Java? Differentiate between Vector and ArrayList. [5, May-2023] [5, Jun-2022] [5, Oct-2023]

Source: https://www.geeksforgeeks.org/

List Interface in Java

www.hbpatel.in

```
import java.util.*;
public class listInterface
{
    public static void main(String args[])
    {
        List<Integer> list = new ArrayList<Integer>();
        list.add(0, 100);
        list.add(1, 120);
        System.out.println(list);
        list.add(300);
        list.add(450);
        list.add(629);
        System.out.println(list);
        list.remove(2);
        System.out.println(list);
        System.out.println(list.get(2));
        list.set(0,99);
        System.out.println(list);
    }
}
```

OUTPUT

```
[100, 120]
[100, 120, 300, 450, 629]
[100, 120, 450, 629]
450
[99, 120, 450, 629]
```

Write a Java program to implement ArrayList & TreeSet class. [5, Nov-2022]

Code: https://github.com/hbpatel1976/Java/blob/main/listInterface.java

Queue Interface in Java

www.hbpatel.in

```
import java.util.*;
public class queueInterface
{
     public static void main(String args[])
    {
        Queue<String> queue = new LinkedList<>();
        queue.add("VSITR");
        queue.add("LDRP");
        queue.add("KSV");
        System.out.println(queue);
        System.out.println(queue.remove());
        System.out.println(queue);
        queue.add("GANDHINAGAR");
        System.out.println(queue);
        System.out.println(queue.peek());
    }
}
```

OUTPUT

```
[VSITR, LDRP, KSV]
VSITR
[LDRP, KSV]
[LDRP, KSV, GANDHINAGAR]
LDRP
```

Code: https://github.com/hbpatel1976/Java/blob/main/queueInterface.java

Set Interface in Java

www.hbpatel.in

```
import java.util.*;
public class setInterface
{
     public static void main(String args[])
    {
        Set<Integer> set1 = new HashSet<Integer>();
        Set<Integer> set2 = new HashSet<Integer>();

        set1.addAll(Arrays.asList(new Integer[] {10,20,30,40,50}));
        set2.addAll(Arrays.asList(new Integer[] {5,20,35,40,70}));
        System.out.println("Set 1 : " + set1);
        System.out.println("Set 2 : " + set2);

        Set<Integer> union = new HashSet<Integer>(set1);
        union.addAll(set2);
        System.out.println("Union : " + union);
```

OUTPUT

```
Set 1 : [50, 20, 40, 10, 30]
Set 2 : [35, 20, 5, 70, 40]
Union : [50, 35, 20, 5, 70, 40, 10, 30]
Intersection : [20, 40]
Difference : [50, 10, 30]
Set 1 contains 10 is true
Set 2 contains 99 is false
```

Code: https://github.com/hbpatel1976/Java/blob/main/setInterface.java

Set Interface in Java

www.hbpatel.in

```
        Set<Integer> intersection = new HashSet<Integer>(set1);
        intersection.retainAll(set2);
        System.out.println("Intersection : " + intersection);

        Set<Integer> difference = new HashSet<Integer>(set1);
        difference.removeAll(set2);
        System.out.println("Difference : " + difference);

        System.out.println("Set 1 contains " + 10 + " is " + set1.contains(10));
        System.out.println("Set 2 contains " + 99 + " is " + set2.contains(99));
    }
}
```

OUTPUT

```
Set 1 : [50, 20, 40, 10, 30]
Set 2 : [35, 20, 5, 70, 40]
Union : [50, 35, 20, 5, 70, 40, 10, 30]
Intersection : [20, 40]
Difference : [50, 10, 30]
Set 1 contains 10 is true
Set 2 contains 99 is false
```

Code: https://github.com/hbpatel1976/Java/blob/main/setInterface.java

Tree Interface in Java

www.hbpatel.in

```
import java.util.*;
public class treeInterface
{
     public static void main(String args[])
    {
        Set<Integer> tree = new TreeSet<>();
        tree.add(100);
        tree.add(50);
        tree.add(150);
        System.out.println(tree);
        tree.add(25);
        tree.add(75);
        tree.add(125);
        tree.add(175);
        System.out.println(tree);
        tree.add(150);
        System.out.println(tree);
    }
}
```

OUTPUT

```
[50, 100, 150]
[25, 50, 75, 100, 125, 150, 175]
[25, 50, 75, 100, 125, 150, 175]
```

- Write a Java program to implement ArrayList & TreeSet class. [5, Nov-2022]
- Write a program to explain the concept of TreeSet. [5, Oct-2023]

Code: https://github.com/hbpatel1976/Java/blob/main/treeInterface.java

Java Programming: Modules

www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

Exception Handling

- Exception is an abnormal/unexpected condition/situation.
- The Exception Handling mechanism handles the runtime errors so that the normal flow of the application can be maintained.
- Major reasons for exceptions:
  - Invalid user input
  - Device failure
  - Loss of network connection
  - Physical limitations (out-of-disk memory)
  - Code errors
  - Opening an unavailable file

What is an exception? Explain try, catch and finally with example. [5, May-2023] [5, Nov-2022] [5, Oct-2023]

Exception Handling

www.hbpatel.in

```
public class withoutExceptionHandling
{
    public static void main(String args[])
    {
        int marks[] = {89, 58, 91};
        System.out.println(marks[10]);
    }

}
```

Error

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 3
at com.mycompany.mavenproject1.withoutExceptionHandling.main(withoutExceptionHandling.java:7)
```

Write a Java program to handle ArrayIndexOutOfBounds exception. [Nov, 2022] (See the program on next slide).

Code: https://github.com/hbpatel1976/Java/blob/main/withoutExceptionHandling.java

Exception Handling

www.hbpatel.in

```
public class withExceptionHandling
{
    public static void main(String args[])
    {
        try
        {
            int marks[] = {89, 58, 91};
            System.out.println(marks[10]);
        }
        catch(Exception e)
        {
            System.out.println("Something is wrong");
        }
        finally
        {
            System.out.println("Have a great day");
        }
    }
}
```

Output

```
Something is wrong
Have a great day
```

Code: https://github.com/hbpatel1976/Java/blob/main/withExceptionHandling.java

Exception Handling

www.hbpatel.in

```
public class moreExceptionHandling
{
    public static void main(String args[])
    {
        int number1=10, number2=0;
        try
        {
            System.out.println(number1/number2);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Something is wrong....following is the exact problem");
            System.out.println(e);
        }
        finally
        {
            System.out.println("Have a great day");
        }
    }
}
```

Error

```
Something is wrong....following is the exact problem
java.lang.ArithmeticException: / by zero
Have a great day
```

- Write a program to raise and handle divide by zero exception. [5, May-2023]
- Write a Java program to handle Arithmetic exception. [5, Nov-2022]
- Write a Java program to implement custom exception (user defined exception). [5, Nov-2022]

Code: https://github.com/hbpatel1976/Java/blob/main/moreExceptionHandling.java

Runtime Exception


```
public class runtimeExceptionHandling
{
    public void sampleMethod()
    {
        throw new sampleMethod();
    }
    public static void main(String[] args)
    {
        try {new runtimeExceptionHandling().sampleMethod();}
        catch (Exception x) {System.out.println("example of runtime exception"); }
    }
}
class sampleMethod extends RuntimeException
{
    public sampleMethod()
    {
        super();
    }
}
```

Output

```
example of runtime
exception
```

Java Programming: Modules

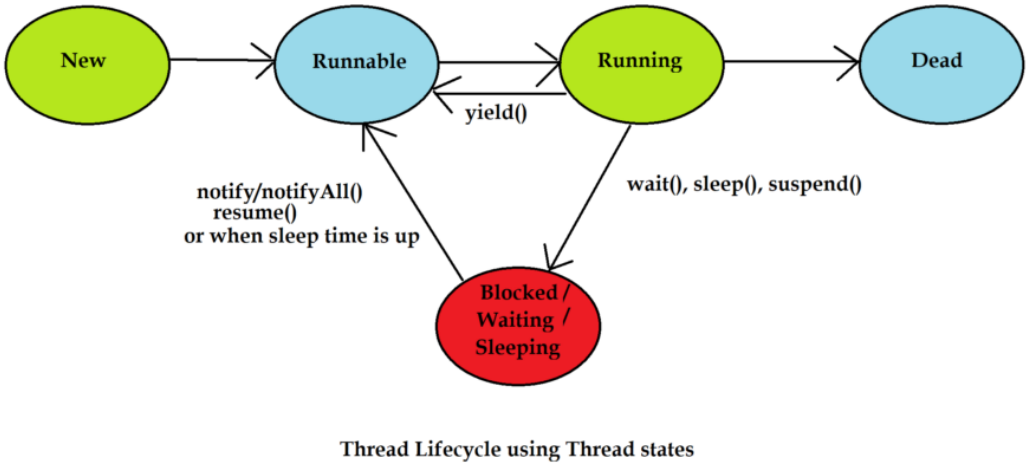
www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

What is Multithreading?

www.hbpatel.in

The ability of OS to execute several programs simultaneously is known as ***multitasking***. In system terminology, it is called ***multithreading***. In multithreading, the program (process) is divided into two or more subprograms (processes), which can be implemented at the same time in parallel. In fact, the processor is doing only one thing at a time, but it switches between the processes so fast that it appears to human beings that all of them are being done simultaneously.



Thread Lifecycle using Thread states

Image Source: javatrainingschool.com

What is multithreading? Explain the thread life cycle in Java. [5, May-2023] [5, Nov-2022] [5, Jun-2022]

What is Multithreading?

A thread is similar to a program that has a single flow of control. The programs so far we have seen are called single threaded programs, i.e. it has a beginning, a body and an end.

Java enables us to use multiple flows of control in developing programs. Each flow of control may be thought of as a separate tiny program (or module) known as *thread* that runs in parallel to others. A program that contains multiple flows of control is known as multithreaded program.

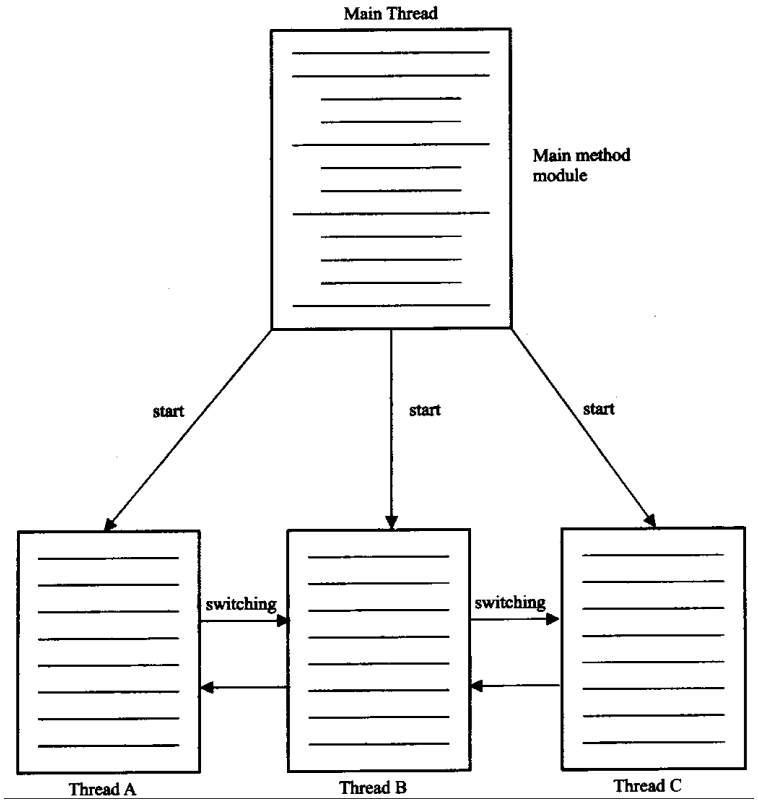
The ability of language to support multithread is referred to as concurrency. Since threads in Java are subprograms of main application and share the same memory space, they are also known as *lightweight threads* or *lightweight processes*.

Multithreading: Advantages

1. It enables the programmers to do multiple things at a time.
2. They can divide a long program (containing operations that are conceptually concurrent) into threads and execute them in parallel.

Creating Threads

www.hbpatel.in



Threads are implemented in form of objects that contain a method called **run()**. The run() method is the heart and soul of any thread. It makes up the entire body of a thread and is the only method in which the thread's behaviour can be implemented.

Creating Threads

www.hbpatel.in

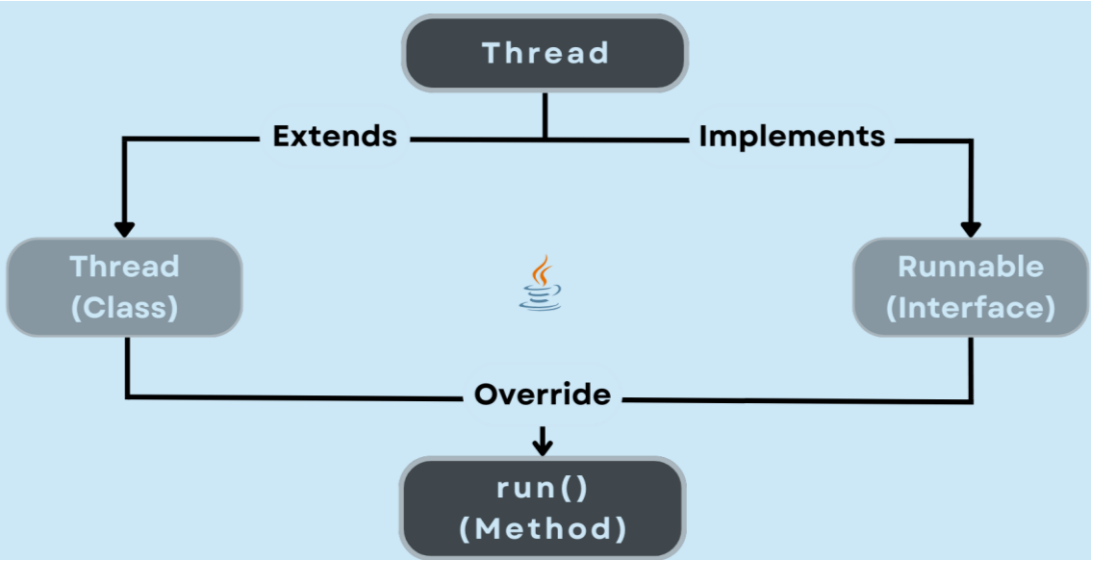


Image source: www.boardinfinity.com

Explain the process of thread creation with example. [5, May-2023]

Thread Lifecycle

www.hbpatel.in

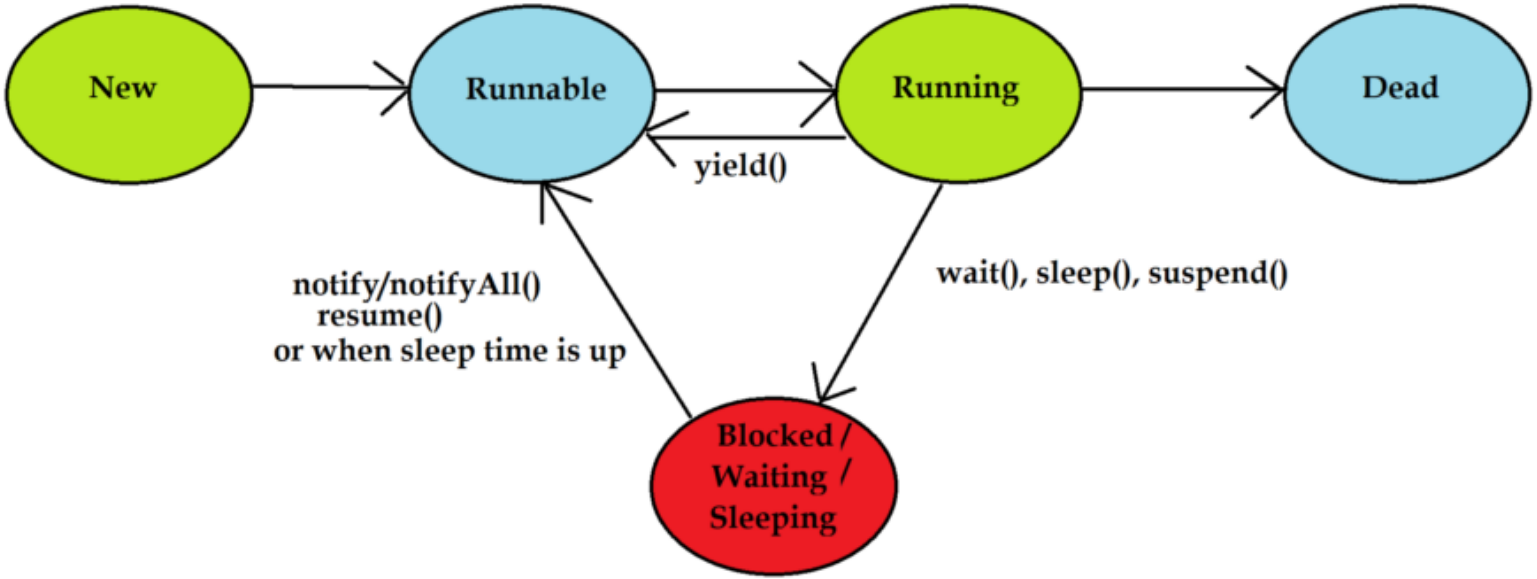


Image source: javatrainingschool.com

What is thread. Describe the complete lifecycle of a thread. [5, Oct-2023]

Creating Threads

A thread can be created in two ways.

1. **By creating a thread class**: Define a class that extends **Thread** class and override tits run() method with the code required by the thread.

```
Class MyThread extends Thread
{
- - -
public void run()
        {
        ---
        ---
        }
        - - - -
}
```

Creating Threads

By converting a class to a thread: Define a class that implements **Runnable** interface. The **Runnable** interface has only one method, **run()**, that is to be defined in the method with the code to be executed by the thread.

```
Class X implements Runnable
        {
        -----
        public void run()
                {
                ----
                ----
                }
        -----
        }
```

Threads: An example

www.hbpatel.in

Write a Java program to define two threads. One will print 1 to 10 and other will print 11 to 20 numbers. [5, Nov-2022] {Modify this program a bit}

```
class A extends Thread
          {public void run()
                    {for(int i=1; i<=5; ++i)
                              {System.out.println("\tFrom Thread A : i = " + i);
                              }
                    System.out.println("Exit From A");
                    }
          }


class B extends Thread
          {public void run()
                    {for(int j=1; j<=5; ++j)
                              {System.out.println("\tFrom Thread B : j = " + j);
                              }
                    System.out.println("Exit From B");
                    }
          }
```

Code: https://github.com/hbpatel1976/Java/blob/main/ThreadTest.java

Threads: An example


```
class C extends Thread
          {public void run()
                    {for(int k=1; k<=5; ++k)
                              {System.out.println("\tFrom Thread C : k = " + k);
                              }
                    System.out.println("Exit From C");
                    }
          }


class ThreadTest
          {
          public static void main(String args[])
                    {
                    new A().start();
                    new B().start();
                    new C().start();
                    }
          }
```

Code: https://github.com/hbpatel1976/Java/blob/main/ThreadTest.java

Multithreading: Outcome

www.hbpatel.in

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>javac ThreadTest.java

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A
        From Thread B : j = 1
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 5
Exit From C

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread C : k = 1
        From Thread B : j = 1
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread C : k = 5
Exit From C
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 1
        From Thread C : k = 5
Exit From C
        From Thread A : i = 5
Exit From A
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>
```

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>javac ThreadTest

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A
        From Thread B : j = 1
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 5
Exit From C
```

Outcome #1

Multithreading: Outcome

www.hbpatel.in

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>javac ThreadTest.java

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A
        From Thread B : j = 1
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 5
Exit From C

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread C : k = 1
        From Thread B : j = 1
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread C : k = 5
Exit From C
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 1
        From Thread C : k = 5
Exit From C
        From Thread A : i = 5
Exit From A
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>
```

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread C : k = 1
        From Thread B : j = 1
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread C : k = 5
Exit From C
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A
```

Outcome #2

Multithreading: Outcome

www.hbpatel.in

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>javac ThreadTest.java

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A
        From Thread B : j = 1
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 5
Exit From C

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread C : k = 1
        From Thread B : j = 1
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread C : k = 5
Exit From C
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread A : i = 5
Exit From A

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 1
        From Thread C : k = 5
Exit From C
        From Thread A : i = 5
Exit From A
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>
```

```
C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>java ThreadTest
        From Thread A : i = 1
        From Thread A : i = 2
        From Thread A : i = 3
        From Thread A : i = 4
        From Thread C : k = 1
        From Thread C : k = 2
        From Thread C : k = 3
        From Thread C : k = 4
        From Thread B : j = 1
        From Thread C : k = 5
Exit From C
        From Thread A : i = 5
Exit From A
        From Thread B : j = 2
        From Thread B : j = 3
        From Thread B : j = 4
        From Thread B : j = 5
Exit From B

C:\Users\VSITR\eclipse-workspace\FirstJavaProjectForVSITRstudents\Temp\src>
```

Outcome #3

Runnable Interface

The runnable interface is used to write applications which can run in a separate thread. Any class that implements the runnable interface is called a thread. The code of the thread is executed by the interpreter after the thread is started. You can implement the runnable interface and create your own multithreading program. There are two ways to implement this interface. The first one is by using the thread subclass. The other one is by overriding the run() method.

Source: www.upgrad.com

Runnable Interface

www.hbpatel.in

```
class runClass implements Runnable
{
       public void run()
        {
            System.out.println("Thread is running");
        }
}
public class testingRunnable
{
    public static void main(String[] args)
    {
        runClass obj1 = new runClass();
        Thread t = new Thread(obj1);
        t.start();
    }
}
```

Output

```
Thread is running
```

Code: https://github.com/hbpatel1976/Java/blob/main/testingRunnable.java

Thread Synchronization

www.hbpatel.in

```
import java.io.*;
import java.util.*;
class classSender
{   public void send(String msg)
            {   System.out.println("Sending...." + msg);
                try {Thread.sleep(1000);}
            catch (Exception e) {System.out.println("Thread interrupted.");}
                System.out.println(msg + "....Sent");
            }
}
class ThreadedSend extends Thread
{   private String msg;
    classSender sender;
    ThreadedSend(String m, classSender obj)
            {   msg = m;
            sender = obj;
            }
            public void run()
            {   synchronized (sender)
                        {
                    sender.send(msg);
                        }
            }
}
```

Output

```
Sending....Welcome
Welcome....Sent
Sending....Bye
Bye....Sent
```

- Explain synchronization in thread with example. [5, May-2023] [5, Oct-2023]
- Explain synchronization with example. [5, Jun-2022]

Code: https://github.com/hbpatel1976/Java/blob/main/ThreadSend.java

Thread Synchronization

www.hbpatel.in

```
class threadSynchronization
{   public static void main(String args[])
            {   classSender send = new classSender();
            ThreadedSend S1 = new ThreadedSend("Welcome", send);
            ThreadedSend S2 = new ThreadedSend("Bye", send);
            S1.start();
            S2.start();
            try {
                    S1.join();
                    S2.join();
                }
            catch (Exception e) {System.out.println("Interrupted");}
            }
}
```

Output

```
Sending....Welcome
Welcome....Sent
Sending....Bye
Bye....Sent
```

Code: https://github.com/hbpatel1976/Java/blob/main/threadSynchronization.java

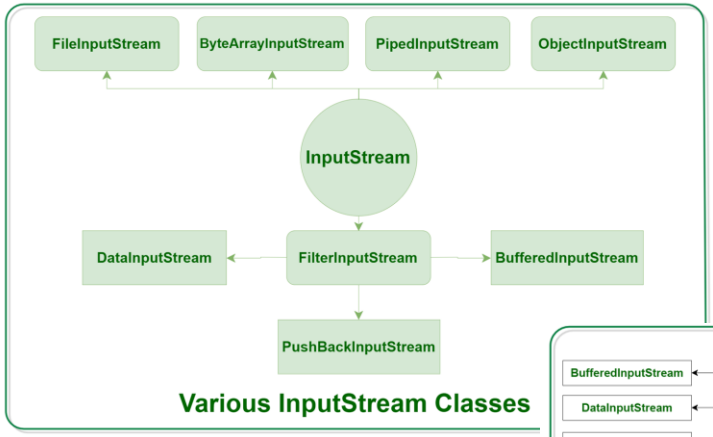
Java Programming: Modules

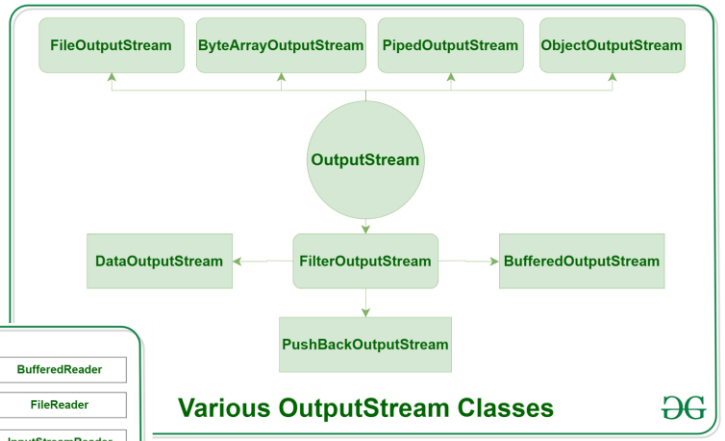
www.hbpatel.in

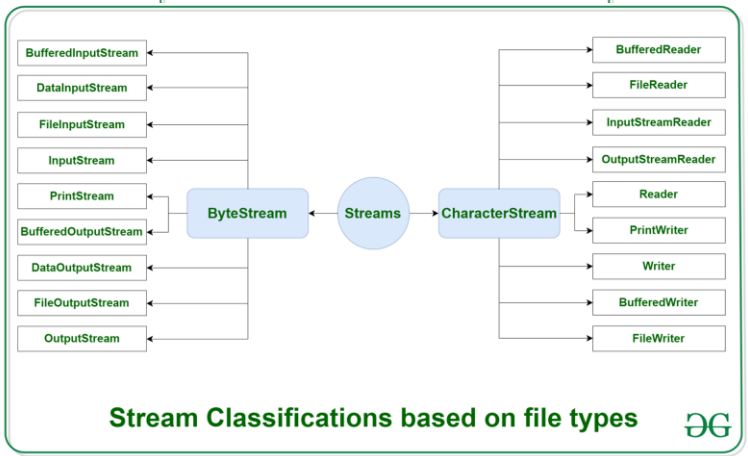
- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

I/O programming

www.hbpatel.in







Explain the various Character I/O classes with example. [5, May-2023] [5, Oct-2023]

Image Source: www.geeksforgeeks.org

throw Vs throws

www.hbpatel.in

Factors	Throw	Throws
Purpose	The 'throw' keyword is used to explicitly throw an exception from within a block of code or a method.	The 'throws' keyword is used in the method signature to declare the exceptions that a method can potentially throw.
Implementation	The 'throw' keyword can only throw a single exception at a time. As such, it is not possible to throw multiple exceptions simultaneously with 'throw'.	The 'throws' keyword allows for the declaration of multiple exceptions that a function could throw.
Exception Type	The 'throw' keyword can only propagate unchecked exceptions, meaning that checked exceptions cannot be propagated with 'throw'.	The 'throws' keyword can be used to declare both checked and unchecked exceptions. For checked exceptions, the 'throws' keyword must be followed by the specific name of the exception class.
Syntax	The 'throw' keyword is followed by the instance variable.	The 'throws' keyword is followed by the names of the exception classes.
Usage	The 'throw' keyword should be used within the body of a method.	The 'throws' keyword should be used within the method signature.

Source: testbook.com

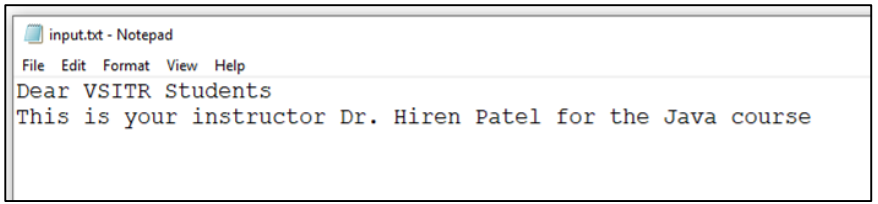
- What are the keywords throw and throws used for? [5, Oct-2023]

I/O programming

www.hbpatel.in

- Write a Java Program to copy content of file1.txt to file2.txt using Java file handling. [5, Nov-2022]
- How can we open and read a text file in Java? Explain your answer with example. [5, Nov-2022]

```
import java.io.*;
public class ioProgramming01
{
public static void main(String[] args) throws IOException
    {
        FileInputStream source = null;
        FileOutputStream target = null;
        try {
                source = new FileInputStream("C:\\Users\\VSITR\\Documents\\NetBeansProjects\\input.txt");
                target = new FileOutputStream("C:\\Users\\VSITR\\Documents\\NetBeansProjects\\output.txt");
                int temp;
                while ((temp = source.read()) != -1) target.write((byte)temp);
            }
        finally {
                if (source != null)source.close();
                if (target != null) target.close();
                }
    }
}
```



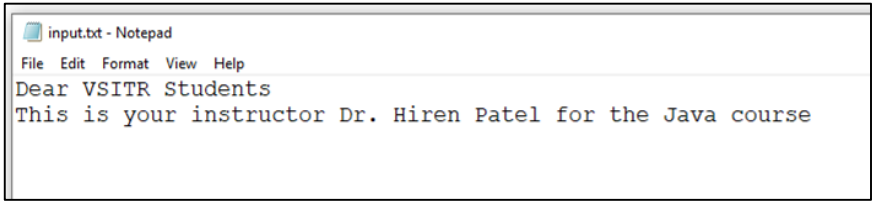
Code: https://github.com/hbpatel1976/Java/blob/main/ioProgramming01.java

I/O programming

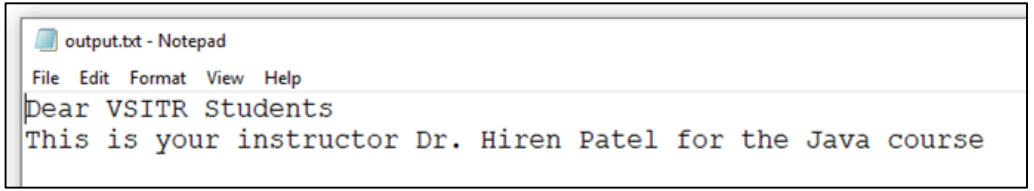
www.hbpatel.in

```
Command Prompt

C:\Users\VSITR\Documents\NetBeansProjects>dir *.txt
 Volume in drive C has no label.
 Volume Serial Number is 2AA7-57DD

 Directory of C:\Users\VSITR\Documents\NetBeansProjects

12/30/2023  11:04 AM                82 input.txt
               1 File(s)             82 bytes
               0 Dir(s)  235,778,445,312 bytes free

C:\Users\VSITR\Documents\NetBeansProjects>type input.txt
Dear VSITR Students
This is your instructor Dr. Hiren Patel for the Java course

C:\Users\VSITR\Documents\NetBeansProjects>dir *.txt
 Volume in drive C has no label.
 Volume Serial Number is 2AA7-57DD

 Directory of C:\Users\VSITR\Documents\NetBeansProjects

12/30/2023  11:04 AM                82 input.txt
12/30/2023  11:09 AM                82 output.txt
               2 File(s)            164 bytes
               0 Dir(s)  235,778,310,144 bytes free

C:\Users\VSITR\Documents\NetBeansProjects>type output.txt
Dear VSITR Students
This is your instructor Dr. Hiren Patel for the Java course

C:\Users\VSITR\Documents\NetBeansProjects>
```

Before Running the Program



After Running the Program



I/O programming

www.hbpatel.in

```
import java.io.*;
public class ioProgramming02
{
public static void main(String[] args) throws IOException
    {
        FileReader sourceStream = null;
        FileWriter targetStream = null;
        try {
                sourceStream = new FileReader("C:\\Users\\VSITR\\Documents\\NetBeansProjects\\input.txt");
                targetStream = new FileWriter("C:\\Users\\VSITR\\Documents\\NetBeansProjects\\output.txt");
                int temp;
                while ((temp = sourceStream.read()) != -1) targetStream.write((byte)temp);
            }
       catch(Exception e) {System.out.println(e);}
       finally {
                if (sourceStream != null)sourceStream.close();
                if (targetStream != null) targetStream.close();
                }
    }
}
```

Code: https://github.com/hbpatel1976/Java/blob/main/ioProgramming02.java

I/O programming

www.hbpatel.in

```
import java.io.*;
import java.nio.CharBuffer;
import java.util.Arrays;

class ioProgramming03
{
            public static void main(String[] args) throws IOException
            {
                        Reader r = new FileReader("C:\\Users\\VSITR\\Documents\\NetBeansProjects\\file.txt");
                        PrintStream out = System.out;
                        char c[] = new char[20];
                        CharBuffer cf = CharBuffer.wrap(c);
                        if (r.ready())
                        {
                                    r.read(c, 0, 15);
                                    out.println(Arrays.toString(c));
                                    r.read(cf);
                                    out.println(Arrays.toString(cf.array()));
                        }
                        r.close();
            }
}
```

Output

```
[V, i, d, u, s, h,  , S, o, m, a, n, y,  , I, , , , , ]
[n, s, t, i, t, u, t, e,  , o, f,  , T, e, c, h, n, o, l, o]
```

Code: https://github.com/hbpatel1976/Java/blob/main/ioProgramming03.java

I/O programming

www.hbpatel.in

```
import java.io.IOException;
import java.io.RandomAccessFile;
public class ioProgramming04
{
    public static void main(String[] args) throws IOException
        {
        RandomAccessFile file = new RandomAccessFile("C:\\Users\\VSITR\\Documents\\NetBeansProjects\\file.txt", "rw");
        file.writeChar('V');
        file.writeInt(1023);
        file.writeDouble(49.22);
        file.seek(0);
        System.out.println(file.readChar());  // 0
        System.out.println(file.readInt());   // 1
        System.out.println(file.readDouble());// 2
        file.seek(2);
        System.out.println(file.readInt());  // 2
        file.seek(file.length());            // 3
        file.writeBoolean(true);             // 4
        file.seek(4);
        System.out.println(file.readBoolean());  // 4
        file.close();
        }
}
```

Output

```
V
1023
49.22
1023
true
```

Code: https://github.com/hbpatel1976/Java/blob/main/ioProgramming04.java

Binary I/O Vs. Text I/O

www.hbpatel.in

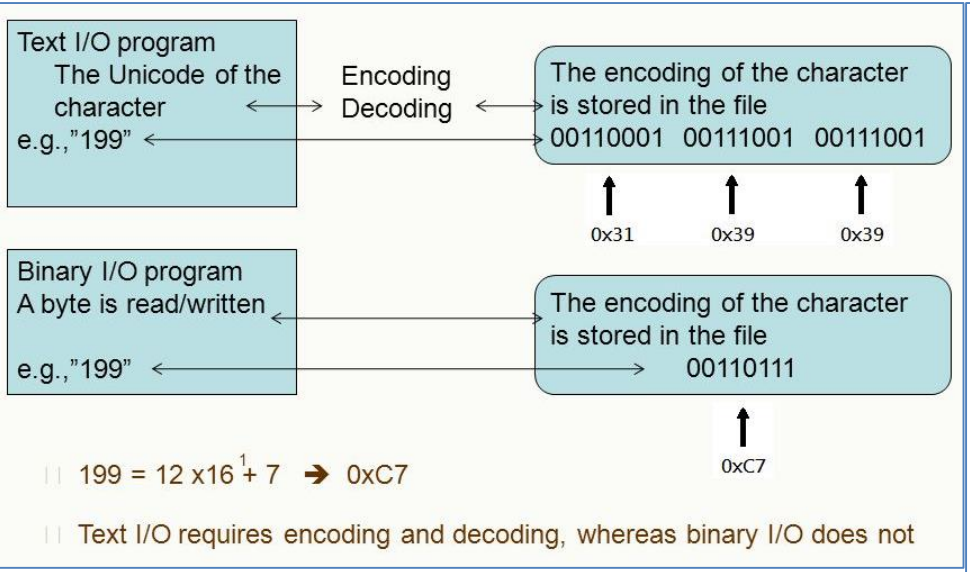


Image Source: https://slideplayer.com/slide/5039780/

Text Files vs. Binary Files

- Number: 127 (decimal)
  - Text file
    - Three bytes: "1", "2", "7"
    - ASCII (decimal): 49, 50, 55
    - ASCII (octal): 61, 62, 67
    - ASCII (binary): 00110001, 00110010, 00110111
  - Binary file:
    - One byte (`byte`): 01111110
    - Two bytes (`short`): 00000000 01111110
    - Four bytes (`int`): 00000000 00000000 00000000 01111110

Image Source: https://slideplayer.com/slide/5039780/

Compare Binary I/O with Text I/O. [5, May-2023] [5, Oct-2023]

Java Programming: Modules

www.hbpatel.in

- Module 0: Installing Java and Running First Java Application
- Module 1: Introduction to Java
- Module 2: Basics of objects and classes
- Module 3: Inheritance and Polymorphism
- Module 4: Introduction to Collection
- Module 5: Exception Handling
- Module 6: Multithreading
- Module 7: I/O programming
- Module 8: Event and GUI programming

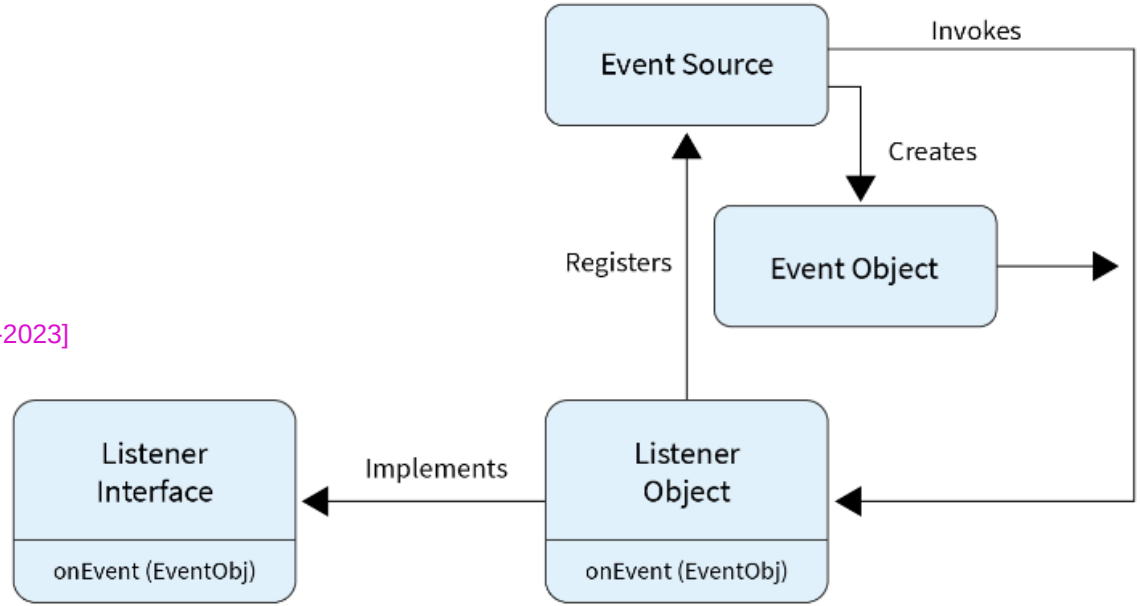
Event and GUI programming

Event handling in Java is the procedure that controls an event and performs appropriate action if it occurs. The code or set of instructions used to implement it is known as the Event handler.

It consists of two major components:

1. The event source and
2. The event listener.

Explain event handling in Java with suitable example. [5, Oct-2023]



Source: www.scaler.com

Event and GUI programming

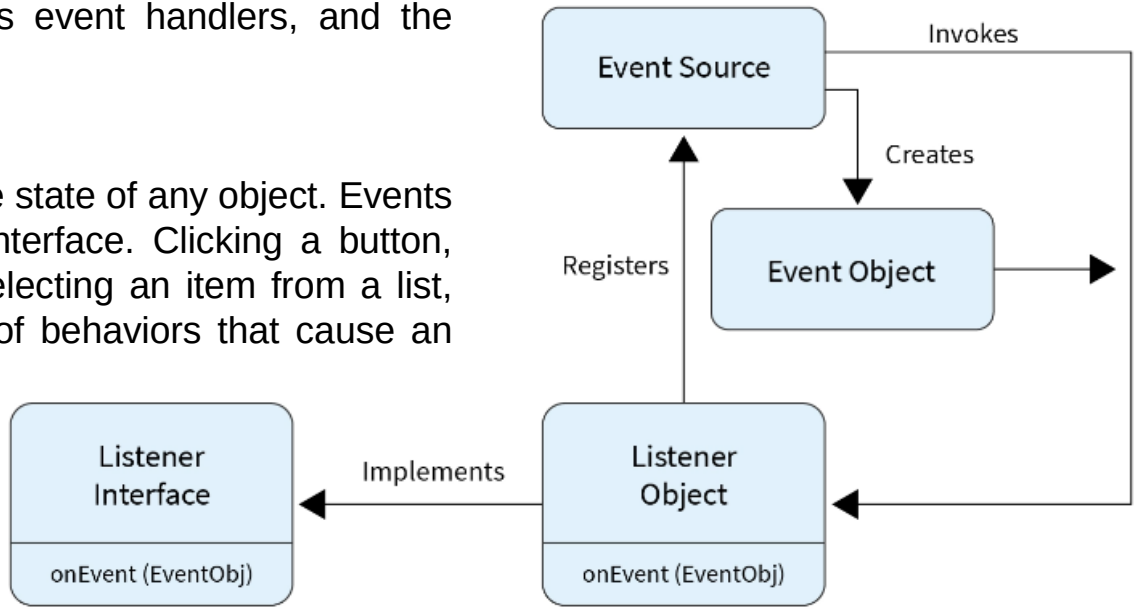
www.hbpatel.in

What is Event Handling in Java?

Many event listeners are frequently used; recall the click of a button that takes you to another website or the mouse scroll? All of this is accomplished through the use of various event handlers, and the mechanism is known as event handling.

Events in Java

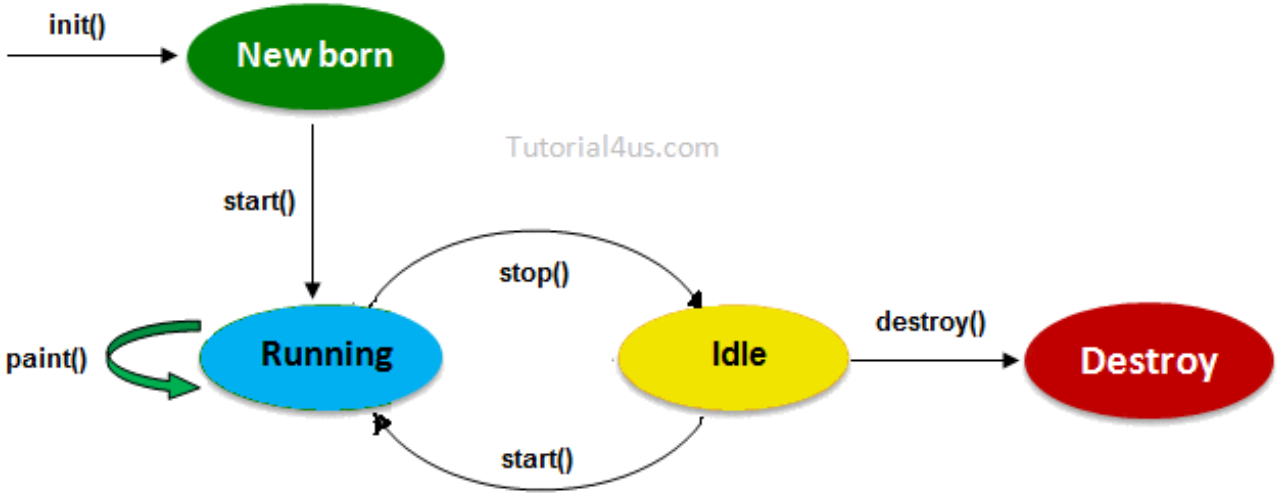
Events in Java represent the change in the state of any object. Events occur when the user interacts with the interface. Clicking a button, moving the mouse, typing a character, selecting an item from a list, and scrolling the page are all examples of behaviors that cause an event to occur.



Source: www.scaler.com

Applet Lifecycle

www.hbpatel.in



- Explain the event handling in Applet with example [5, May-2023, 5, Jun-2022]
- Explain the lifecycle of Applet. [5, May-2023], [5, Nov-2022] [5, Jun-2022] [5, Oct-2023]

Source: tutorial4us.com

Layout Manager

The LayoutManagers are used to arrange components in a particular manner. The Java LayoutManagers facilitates us to control the positioning and size of the components in GUI forms. LayoutManager is an interface that is implemented by all the classes of layout managers. There are the following classes that represent the layout managers:

```
java.awt.BorderLayout          java.awt.FlowLayout
java.awt.GridLayout            java.awt.CardLayout
java.awt.GridBagLayout         javax.swing.BoxLayout
javax.swing.GroupLayout        javax.swing.ScrollPaneLayout
javax.swing.SpringLayout
```

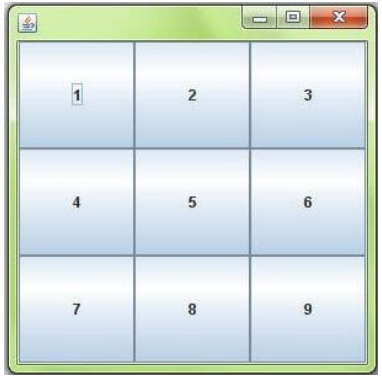
- Explain Java Boarder Layout with suitable example. [5, Nov-2022]
- Explain Java Grid Layout with suitable example. [5, Nov-2022]
- What is Layout? Explain various Layout managers in Java. [5, Jun-2022]
- Explain various Layouts in Java with proper examples. [5, Oct-2023]

Source: www.javatpoint.com

Layout Manager

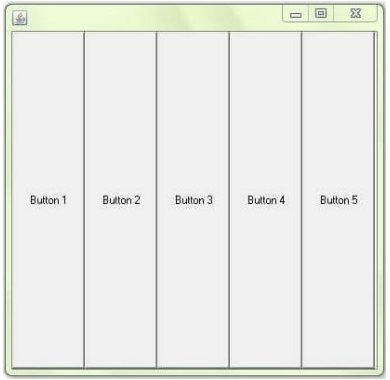
www.hbpatel.in

Boarder Layout



Grid Layout

Flow Layout



Box Layout

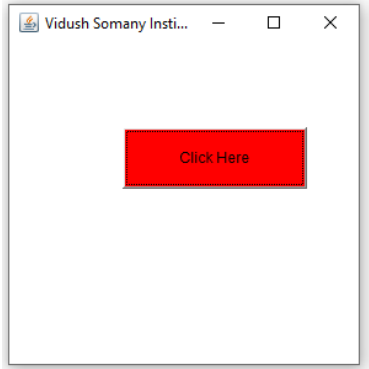
- Explain Java Boarder Layout with suitable example. [5, Nov-2022]
- Explain Java Grid Layout with suitable example. [5, Nov-2022]
- What is Layout? Explain various Layout managers in Java. [5, Jun-2022]

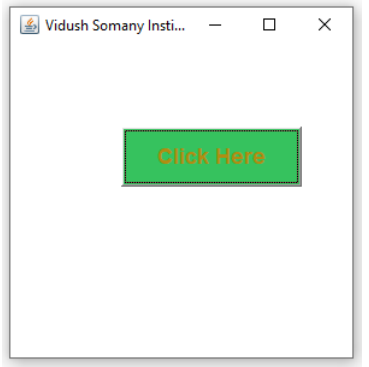
Source: https://www.javatpoint.com/java-layout-manager

Event and GUI programming

www.hbpatel.in

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.Random;

public class eventButtonClick
{
    public eventButtonClick(){

        Frame frame = new Frame("Vidush Somany Institute of Technology and Research");

        Button btn = new Button("Click Here");
        btn.setBounds(100,100,150,50);
        btn.setBackground(Color.red);
        btn.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e) {
                Random rand = new Random();
                float r = rand.nextFloat();
                float g = rand.nextFloat();
                float b = rand.nextFloat();
```

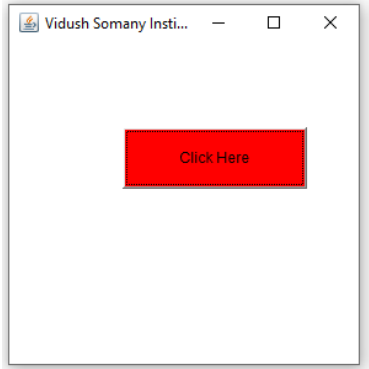


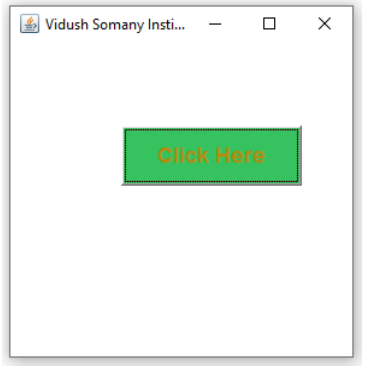


Code: https://github.com/hbpatel1976/Java/blob/main/eventButtonClick.java

Event and GUI programming

```
                btn.setBackground(new Color(r,g,b));
                r = rand.nextFloat();
                g = rand.nextFloat();
                b = rand.nextFloat();

                btn.setForeground(new Color(r,g,b));

                btn.setFont(new Font("Arial", Font.BOLD, 18));
            }
        });
        frame.add(btn);
        frame.setSize(300,300);
        frame.setLayout(null);
        frame.setVisible(true);

        frame.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e) {frame.dispose();}
        });
    }
    public static void main(String[] args) {eventButtonClick f = new eventButtonClick();}
}
```



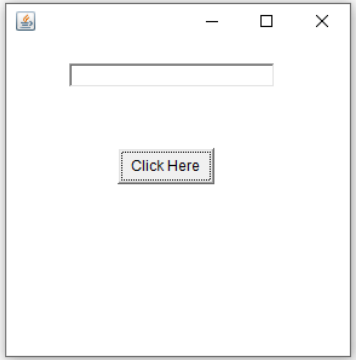


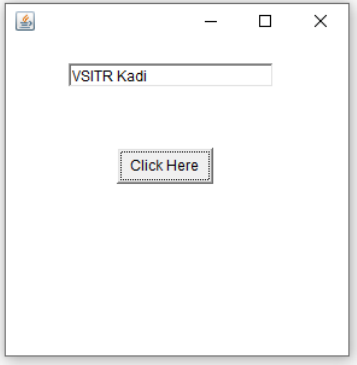
Code: https://github.com/hbpatel1976/Java/blob/main/eventButtonClick.java

Event and GUI programming

www.hbpatel.in

```
import java.awt.*;
import java.awt.event.*;
class eventButtonTextField extends Frame implements ActionListener
{
    TextField textBox;
    eventButtonTextField()
    {
        textBox=new TextField();
        textBox.setBounds(60,50,170,20);
        Button btn=new Button("Click Here");
        btn.setBounds(100,120,80,30);

        btn.addActionListener(this);//passing current instance

        add(btn);add(textBox);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e){textBox.setText("VSITR Kadi");}
    public static void main(String args[]){new eventButtonTextField();}
}
```

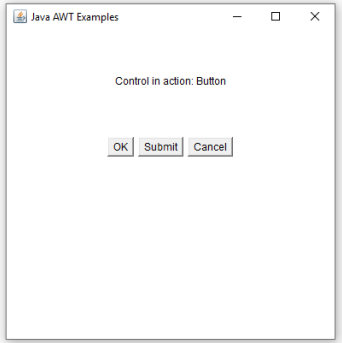


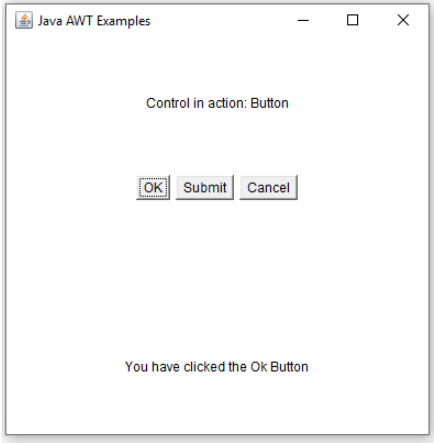


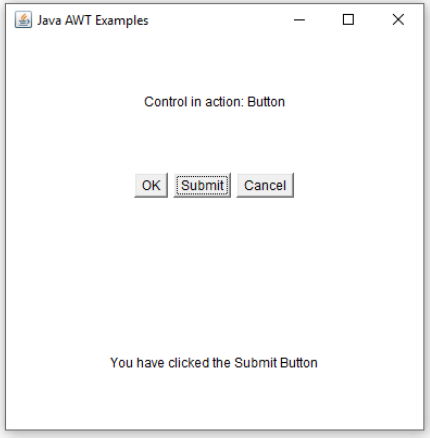
Code: https://github.com/hbpatel1976/Java/blob/main/eventButtonTextField.java

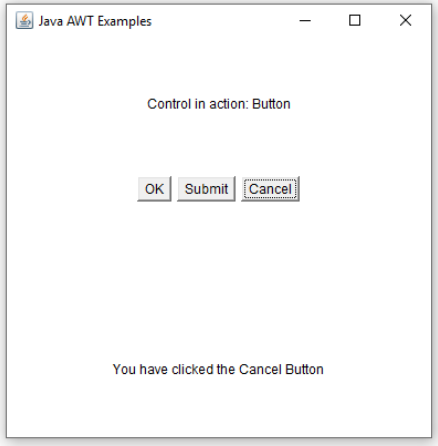
Event and GUI programming

www.hbpatel.in









Event and GUI programming

www.hbpatel.in

```
import java.awt.*;
import java.awt.event.*;
public class multipleButtons
{
   private Frame mainFrame;
   private Label headerLabel;
   private Label statusLabel;
   private Panel controlPanel;
   public multipleButtons(){prepareGUI();}
   public static void main(String[] args)
   {
      multipleButtons  multipleButtons = new multipleButtons();
      multipleButtons.showEventDemo();
   }

   private void prepareGUI()
   {
      mainFrame = new Frame("Java AWT Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));
      mainFrame.addWindowListener(new WindowAdapter() {public void windowClosing(WindowEvent
windowEvent){System.exit(0);}
    });
```

Code: https://github.com/hbpatel1976/Java/blob/main/multipleButtons.java

Event and GUI programming

www.hbpatel.in

```
headerLabel.setAlignment(Label.CENTER);
    statusLabel = new Label();
    statusLabel.setAlignment(Label.CENTER);
    statusLabel.setSize(350,100);

    controlPanel = new Panel();
    controlPanel.setLayout(new FlowLayout());

    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
   }

   private void showEventDemo(){
      headerLabel.setText("Control in action: Button");

      Button okButton = new Button("OK");
      Button submitButton = new Button("Submit");
      Button cancelButton = new Button("Cancel");

      okButton.setActionCommand("OK");
      submitButton.setActionCommand("Submit");
      cancelButton.setActionCommand("Cancel");
```

Code: https://github.com/hbpatel1976/Java/blob/main/multipleButtons.java

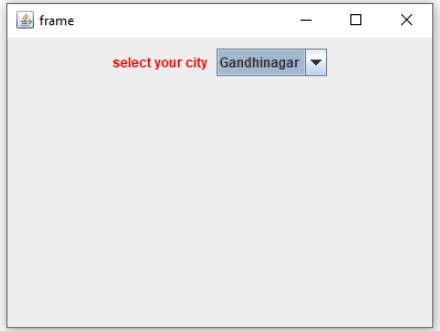
```
okButton.addActionListener(new ButtonClickListener());
      submitButton.addActionListener(new ButtonClickListener());
      cancelButton.addActionListener(new ButtonClickListener());

      controlPanel.add(okButton);
      controlPanel.add(submitButton);
      controlPanel.add(cancelButton);

      mainFrame.setVisible(true);
   }

   private class ButtonClickListener implements ActionListener{
      public void actionPerformed(ActionEvent e) {
         String command = e.getActionCommand();
         if( command.equals( "OK" ))  {
            statusLabel.setText("You have clicked the Ok Button");
         }
         else if( command.equals( "Submit" ) )  {
            statusLabel.setText("You have clicked the Submit Button");
         }
         else  {
            statusLabel.setText("You have clicked the Cancel Button");
         }
      }
   }
}
```

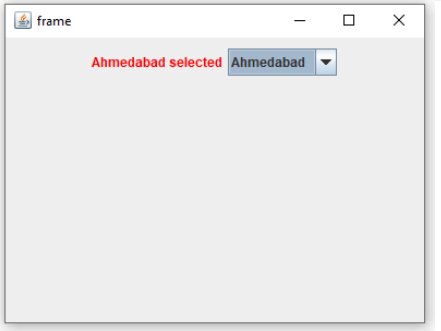
Code: https://github.com/hbpatel1976/Java/blob/main/multipleButtons.java

Event and GUI programming

www.hbpatel.in

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
class citySelection extends JFrame implements ItemListener
{             static JFrame f;
              static JLabel l;
              static JComboBox c;
              public static void main(String[] args)
              {             f = new JFrame("frame");
                            citySelection s = new citySelection();
                            f.setLayout(new FlowLayout());
                            String s1[] = { "Gandhinagar", "Kadi", "Visnagar", "Ahmedabad", "Surat"};
                            c = new JComboBox(s1);c.addItemListener(s);
                            l = new JLabel("select your city ");
                            l.setForeground(Color.red);
                            JPanel p = new JPanel();
                            p.add(l); p.add(c); f.add(p);
                            f.setSize(400, 300);
                            f.show();
              }
              public void itemStateChanged(ItemEvent e)
              {             if (e.getSource() == c) {l.setText(c.getSelectedItem() + " selected");}
              }
}
```





Code: https://github.com/hbpatel1976/Java/blob/main/citySelection.java

Application Vs. Applet

www.hbpatel.in

Parameters	Java Application	Java Applet
Definition	Applications are just like a Java program that can be executed independently without using the web browser.	Applets are small Java programs that are designed to be included with the HTML web document. They require a Java-enabled web browser for execution.
main () method	The application program requires a main() method for its execution.	The applet does not require the main() method for its execution instead init() method is required.
Compilation	The “javac” command is used to compile application programs, which are then executed using the “java” command.	Applet programs are compiled with the “javac” command and run using either the “appletviewer” command or the web browser.
File access	Java application programs have full access to the local file system and network.	Applets don’t have local disk and network access.
Access level	Applications can access all kinds of resources available on the system.	Applets can only access browser-specific services. They don’t have access to the local system.
Execution	Applications can execute the programs from the local system.	Applets cannot execute programs from the local machine.
Program	An application program is needed to perform some tasks directly for the user.	An applet program is needed to perform small tasks or part of them.
Run	It cannot run on its own; it needs JRE to execute.	It cannot start on its own, but it can be executed using a Java-enabled web browser.

Source: geeksforgeeks.org Differentiate Application and Applet. [5, May-2023] [5, Jun-2022] [5, Oct-2023]