* Java was developed by Sun micro System in 1995 by James Gosling.
* Before giving the name Java Earlier name was Oak which means symbol of strength and it is a national tree in countries like USA, Germany, France.
* Java is an Object Oriented, Class based, Secured and General Purpose Programming language.
* Object oriented means that provide many concepts such as inheritance, polymorphism, abstraction, etc.
* It is a real world entity and this entity has state and behaviour.
* Object contain an address and take up some space in memory.
* They can communicate without knowing the details of each others data or code.
* The only necessary thing is the type of message accepted and the type of response returned by the object.
* Example of object : dog -> State : name, colour -> Behaviour : barking,

**Classes**

* Collections of Objects is called class.
* It does not consume any space in memory.

**Features of Java**

* Java is object oriented
* Simple
  + In Java no explicit pointers are used and no operator overloading.
* Platform independent
* Secure
  + Because no pointers are used in Java and Java programs are run on JVM (Java Virtual Machine)
  + And if there is a problem in Java program then this exits in JVM and not in Operating System.
* Architectural Neutral
  + No need to check Operating System.
* Portable
  + Run Bytecode anywhere. (Write once Run Anywhere)
* Multithreaded
  + Many threads in one program enable us to Run the different methods/classes of a program side by side.

**Difference b/w JDK, JRE and JVM**

|  |  |  |
| --- | --- | --- |
| **JVM** | **JRE** | **JDK** |
| JVM is a Java virtual machine.  It provides Run time environment in which Java bytecode can be executed and it is platform dependent. | JRE is Java runtime environment. It is the actual implementation of JVM.  It is physically present. It contains libraries and files that JVM uses. | JDK stands for Java development kit also physically present. It contains JRE and other development tools. |
| * It loads the code * It verifies the code * It executes the code * It provides Run time Environment |  |  |

**Running a program**

class Test

{

}

compile -> javac Test.java

Run -> java Test

**In System.out.println(“”);**

System : Class name in java.lang package

Out : Its a static Variable of type printStream present in System class

println : Its a method to print Anything on console.

**In public static void main:**

compiling without main method:

NoMainMethodFound:

NoSychErrorMethod: main

Whether class contain main method or not and whether main method is declared or not according to requirement or not, these things are not checked by compiler, but at runtime JVM is responsible to check these things.

**-> public static void main(String[] args)**

**public ->** To call by JVM from anywhere.

**static ->** Without existing object also JVM has to call this method and main method has not way related to any object.

**void ->** main method won’t return anything to JVM.

**main ->** this is the name configured inside JVM.

**String[] args ->** This is used to pass command line arguments.

* This syntax is very strict and if we perform any changes we will get runtime exception saying “No such method Error”.
* Here :
  + main(String[] a)
  + main(String []a)
  + main(String a[]) are all allowed.
* The order of modifiers(public static) is not important.
* Instead of args we can use any valid java identifier.

**Types of variables in java**

1. PRIMITIVE VARIABLES
   * They are used to represent primitive values.
     + Eg. int, float.
2. REFERENCE VARIABLES
   * Used to refer objects of class.
     + Test t1 = new Test() // Test is class name and t1 is the object of class Test

Based on the purpose and relation variables are of 3 types :-

1. Instance variable
   * Value of that variable varies from object to object.
   * It should be declared within the class directly but outside any method, block, constructor.
2. Static variable
   * Value does not vary from object to object.
   * Also known as class/level variable
   * Declared as static, cannot be local, only one copy is created.
3. Local variable
   * It is created inside a block, constructor or method.
   * Its initialization is necessary before using it.
   * Not defined with static keyword.
   * Its scope is not outside the block, constructor or method.

**Identifiers**

It is any name either variable name, method name, class name and interface name.

**-> Rules for identifiers**

1. space cannot be used
2. only \_ and $ are used
3. reserved(predefined words/key words) words cannot be used. (true, false protected, private)
4. integer cannot be used at first place

**Naming conventions**

* Variables : emp, emp\_no, emp\_no\_chitkara
* Method : main(), mainJava(), main JavaProgram()
* class : Main, MainJava, MainJavaProgram

**Java Datatypes**

There are total 8 datatypes in java divided into :-

1. Primitive
   1. Boolean
      1. boolean (bool)
   2. Numerical
      1. charactar (char)
      2. integral
         1. integer
            1. int
            2. long
            3. short
         2. float
            1. float
            2. double
2. Non-primitive
   1. String
   2. Array
   3. class
   4. Structure

**new** helps to declare an object

To create an object of a class there are three way: new, clone(), newInstance()

**Super and final Keyword**

Super is a reference variable that is used for immediate parent class object.

super() is used to :-

1. To invoke immediate parent class constructor. (
2. To invoke Parent class method. ( super.methodname
3. To refer immediate parent class variable. (super.varaible name)

class Parent

{

int i=10;

}

class child extends Parent

{

int i=20;

void m1()

{

System.out.println(“value of i is child” + i);

System.out.println(“value of i is parent” + i);

}

}

class Supertest

{  
 public static void main(String args[])

{

child c1 = new Child();

c1.m1();

}

}

**Overloading**

When many methods in a class have the same name but different signature or different no of arguments.

**Overriding**

To overwrite the functionality of an existing method.

**Arrays**

Limitations:-

1. Fixed in size i.e. once we create an array there is no chance of decrease or increase the size based on our requirements. Hence, its compulsory to know the size of an array in advance.
2. Only homogeneous

An array is an indexed collection of fixed no of homogeneous data elements. The main advantage of array is we can represent huge no of values by using single variable so that readability of the code will be improved.

**Array declaration**

**1D:**

int[] x;

int []x;

int x[];

At the time of declaration we can’t define the size of array otherwise we’ll get compiled time Error.

eg. int[6] x;

**2D:**

int[][] x;

int [][]x;

int x[][];

int[] []x;

int []x[];

int[] x[];

int[] a, b; both are 1D array

int[] a[], b; a->2D, b->1D

int[] a[], b[]; both are 2D

int[] []a, b; both are 2D

int[] []a, b[]; a->2D, b->3D

int[] []a, []b; compile time error

**3D:**

1. int[][][] a;
2. int [][][]a;
3. int a[][][];
4. int[] [][]a;
5. int [][]a[];
6. int[] a[][];
7. int[] []a[];
8. int[][] []a;
9. int[][] a[];
10. int []a[][];

**Constructor**

* Constructor is a block of code similar to method.
* It is called when an instance of class is created.
* At the time of calling constructor memory for the object is allocated in the memory.
* It is a special type of method which is used to initialize the object.
* Every time an object is created using the new keyword () at least one constructor is called.
* It calls a default constructor if there is no constructor available in the class. In that case java compiler provides a default constructor by default.

**Rules for creating Java constructor :-**

1. Constructor name must be the same as its class name.
2. Constructor must have no explicit return type.
3. Java constructor cannot be abstract, static, final and synchronized.

Eg. (Java program to create and call default constructor.)

class Test

{

Test() // Create a constructor

{

System.out.println(“Test is created”);

}

public static void main(String[] args)

{

Test t1 = new Test();

}

}

**Purpose of default constructor:-**

Default constructor is used to provide the default values to the object, for e.g. 0, null depending on the type of variable.

class Student

{

int id;

String name;

void display()

{

System.out.println(id+” “+name);

}

public static void main(String[] args)

{

Student s1 = new Student();

Student s2 = new Student();

s1.display();

s2.display();

}

}

**Parametrized constructor**

A constructor which have specific no of parameters is called parametrized constructor.

It is used to provide different values to distinct objects.

**Constructor Overloading**

Its a technique in Java having more than one constructor with different parameter list. They are arranged in a way that each constructor performs different task. They are differentiated by the compiler by the no of parameters

**Final KeyWord**

We can use final keyword with either variable, method or class. Its value cannot be changed during the scope of the program.

Eg.

class final1

{

final int i =10;

final1()

{

i=i+10; // Error cannot assign value to final variable

}

}

class finalKeyword

{

public static void main(String[] args)

{

final1 f1 = new final1();

}

}

**Final method cannot be overridden.**

**Final class cannot be inherited.**

**Declaration of 1D array**

Every array in Java is an object hence we can create only using new() keword.

int[] a=new int[3].

For every array we have an inbuilt class available and these classes are part of java.lang package.

* At the time of creation compulsory we should specify the size or else we’ll get error.
* Its valid or legal to have an array with 0.
* If we are trying to specify array size negative then we’ll get runtime exception: negative array size exception.

Int[] -> [I

int[]

Array Creation 2D

In 2d array matrix approaches not follow but array of arrays is follow for multidimensional array creation.

The main advantage of this approach is memory utilization is improved.

Array Initialization

Once we create an array every array element by default initialize with default value.

Whenever we are trying to print any reference variable internally toString() method will be called which is implemented by default to return the string in the form classname@hashcode\_in\_Hexadecimal form.

Int[] x= new int[3];

Sop(x); ( classname@hashcode)

Sop(x[0]); (0)

2D Array Initialization

int [][] x= new int[2][3];

sop(x); ( classname@hashcode)

sop(x[0]); ( classname@hashcode)

sop(x[0][0]); (0)

Null Pointer Exception: If we are trying to form any operation on null then we will get null pointer exception.

Once we create an array by default initialize with default value if we are not satisfied with default values then we can override these values with the customized values.

ArrayIndexOutOfBound Exception: If we are trying access array element with out of range (either positive or negative int value) then we will get run time exception saying ArrayIndexOutOfBound Exception.

Int[] x; (declaration)

x=new int[3]; (creation)

x[0]=50; (initialization)

x[1]=60; (initialization)

x[2]=70; (initialization)

int [] x={10,20,30};

int[][] x= {{10,20,30},{40,50},{60,70}};

int[][][] x= {{{10,20,30},{50,60}},{{70,80},{90,100,110}}};

We can use this shortcut for multidimensional array also.

Jagged Array

Jagged array is also same as 2D array.

Int[][] x;

X= new int[2][0];

X[0]=new int[2];

X[1]=new int[4];

X[0][0] =10;

X[1][1] =20;

X[1][3] =30;

Q1. Take 10 int value from user and store them in array and print them on the screen?

Q2. Take 20 inputs from the user and print the following:-

1. No. of positive no.

2. No. of negative no.

3. No. of odd no.

4. No. of even no.

5. No. of Zero

Q3. Take 10 int input from the user and store them in the array now copy all the elements in an another array but in reverse order?

Q4. Initialize and print all the elements of 2D array?

Q5. Find largest and smallest element of an array?

Q6. Rotate 1D array k times.

Q7. Print 2D array column wise or row wise.

Q8. Print boundary values.

Q9. Print in zig-zag format.

**String:**

String is basically an object that represents sequence of char values. An array of characters works same as java String.

Char []ch = {‘j’, ‘a’, ‘v’, ‘a’};

String s = new String(ch);

SOP(ch) 🡪 java

Java String class provides some methods to perform operations on String such as:

* 1. compare()
  2. concat()
  3. equals()
  4. split()
  5. length()
  6. replace()
  7. compareTo()
  8. substring()

The java.lang.string class implements serializable, comparable and char sequence interfaces.

**CharSequence interface** is used to represent the sequence of characters.

string, stringBuffer and stringBuilder implement charSequence interface.

The java String is **immutable**, which means it cannot be changed, whenever we change any String a new instance is created.

For **mutable** strings we use stringBuffer and stringBuilder classes.

2 ways to create String object:-

* + 1. By stringLiteral
       1. String s1 = “Java”;
       2. Each time you create the String literal, JVM checks the string constant pool first.
       3. If the string is already in the pool a reference to the tool instance is returned. If the string does not exist in the pool, a new instance is created and placed in the pool.
       4. String s1 = “Java”;

String s2 = “Java”;

* + 1. By new keyword