Course Content:

- Modules: Module 1,2,3,4
- Module 1: Html, CSS, JS
- Module 2: Core Java + Advance Java + Database + SQL
- Module 3: Spring Core + Spring Boot + Microservices+ maven + Git + Docker + Aws
- Module 4: Communication Skills + Interview Framing + Resume Prep + Self Confidence

Java Features

1. Simple

2. Platform Independent

3. Robust (Strong)

- Automatic Memory Management
- Exception handling (self-healing)

4. OOPS (Object Oriented Programming System)

5. Secure

Environment Setup

1. Download and install jdk 1.8 [2 folder]

- JDK (Java Development Kit)[provide set of tools to develop program]
- JRE (Java Runtime Environment)[jre gives us Platform to execute program,and jvm is inside jre]

2. set the path

- system variable : C:\Program Files\Java\jdk1.8.0_202
- user variable : %JAVA_HOME%\bin

3. verify

. java -version

Where to write code?

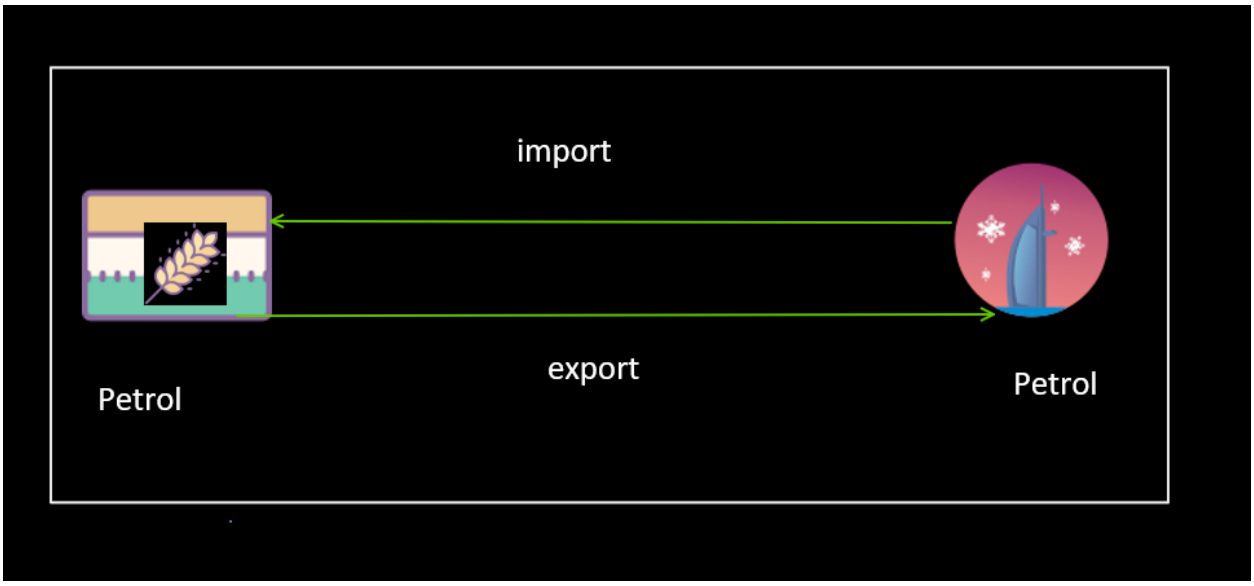
1. Notepad

2. Notepad++

Java program Structure

1. package statement

2. import statement [A--->B]



3. class declaration

4. Methods

5. variable

Translators

Translators are used to convert from one format to another format

We have 3 types of translators

1) Interpreter

2) Compiler

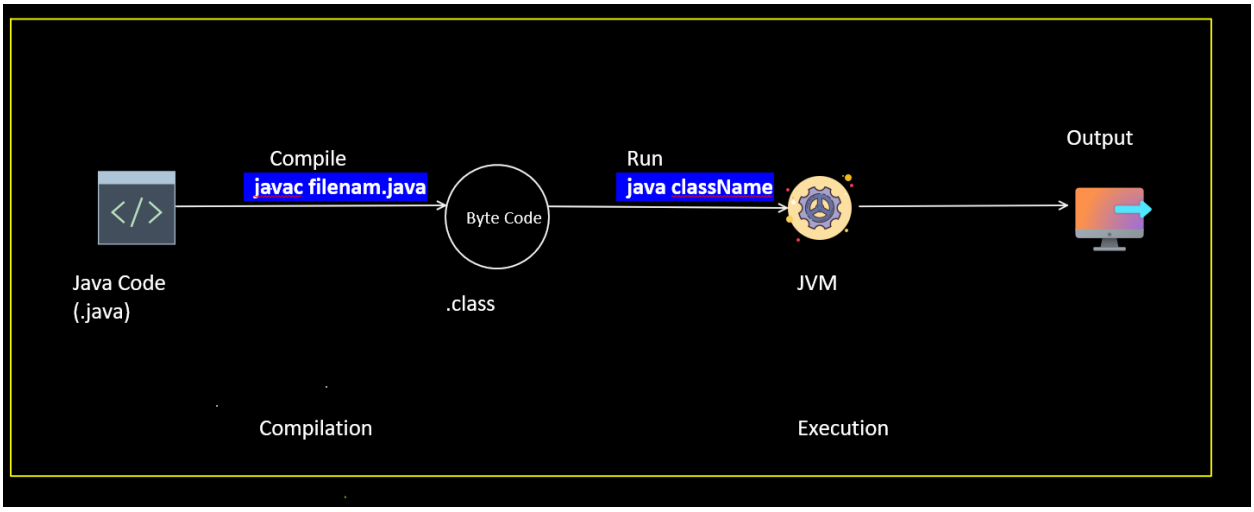
3) Assembler

Interpreter will convert the program line by line (performance is slow)

Compiler will convert all the lines of program at a time (performance is fast)

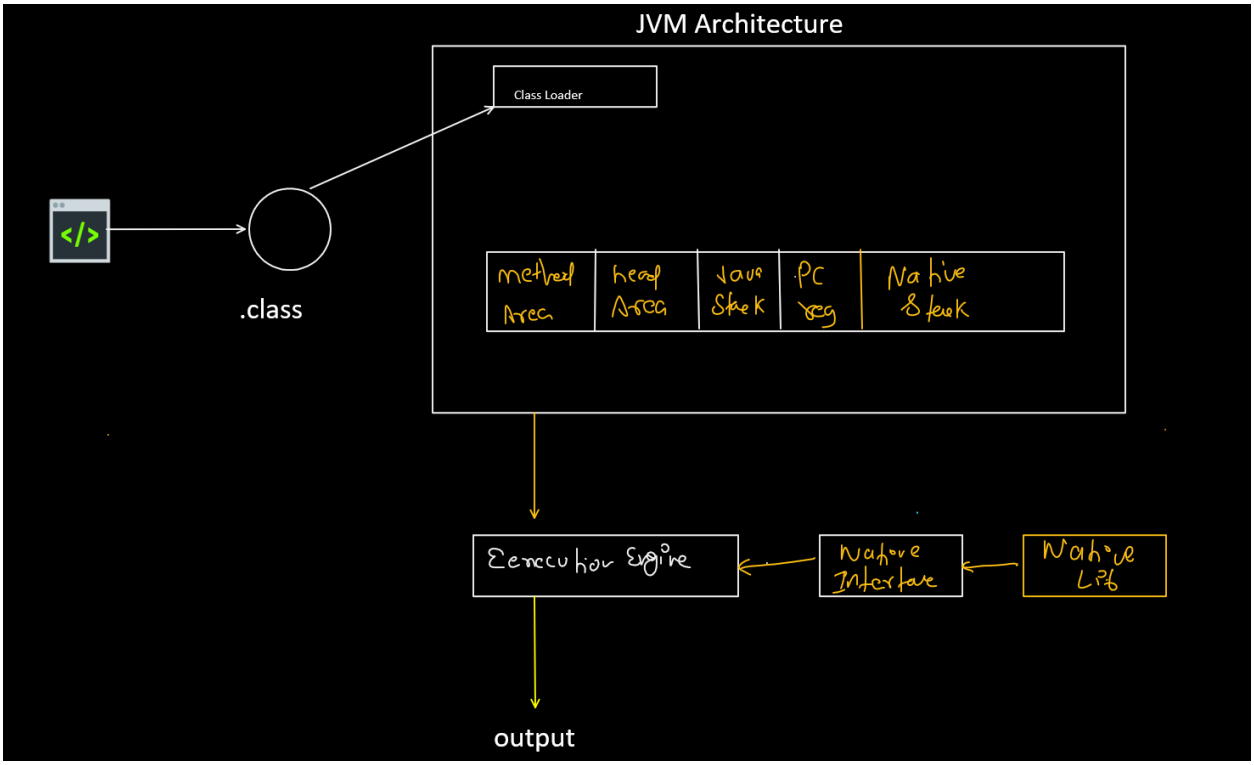
Assembler is used to convert assembler programming languages into machine language

How Java Code Compiles



JVM architecture

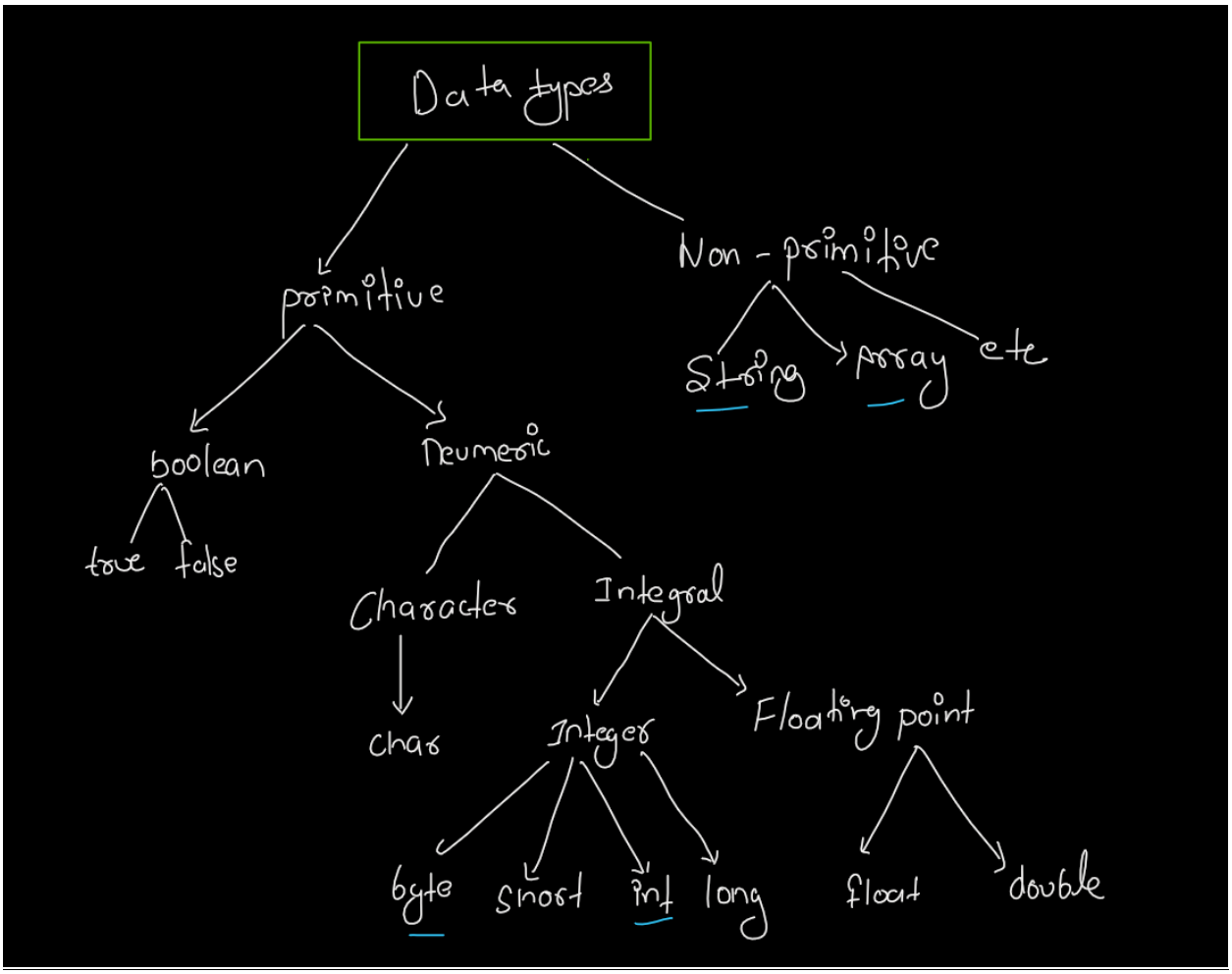
- Class loader: It will load .class file into JVM
- Method Area: Class code will be stored here
- Heap area: Objects will be stored into heap area
- Java Stack: Method execution information will be stored here
- PC Register: It will maintain next line information to execute
- Native Stack: It will maintain non-java code execution information
- Native Interface: It will load native libraries into JVM
- Native Libraries: Non-java libraries which are required for native code execution
- Execution Engine: It is responsible to execute the program and provide output/result. It will
- use Interpreter and JIT for execution.



Variables

- variables are used to store the data during program execution
- We need to specify type of the variable to store the data
- To specify type of data we will use 'data types'
- To declare the variable in Java, we can use following syntax

Data Type



Type of comment in java

1. single line comment --> //

2. multi line commmnet --> /* */

3. documentation commmnet -->

Java Question Paper -1

1) Explain software project layers

2) What is database and why we need it ?

3) What is programming language & why we need programming language ?

4) What are the features of java ?

5) What is JAVA ?

6) What is the difference between C and Java ?

7) What type of applications we can develop using java & brief them ?

8) What is the difference between JDK, JRE and JVM ?

9) What is the execution flow of java program ?

10) What is the difference between interpreter and compiler?

12) Write JVM architecture

13) write a java program to print hello

Java Identifiers

Identifiers are the names given to class Method, interface.

Rules for namming Indetifires:

1. valid char : lower case , Upper case, digit, _, $
2. Start with : letter,_, $. it cannot start with digit
3. case Sesitivity : car, CAR ,cAR,Car
4. Reserved KeyWords : (int, for,if,class) cannot be used

Convention For Naming Identifiers(best for Identifiers):

1. CamelCase: for classes, method and variable eg: carName, cityName
2. MeaingFul : String cityName = "10202" ,cityName="Agra"

Eg: Identifiers

age ----------> valid

_marks --------> valid

$value --------> valid

8marks ----------> invalid

class ----------> invalid

my-name ---------> invalid

@cityName -------> invalid

calCulater ------> valid

CHAPTER 2

Operators

- Arithmetic
- Relational
- Logical
- Assignment
- New
- Dot (.) Operator

Control Statements

- Conditional Statements
- Looping Statements
- Transfer Statements

Operators

-> Operator is a symbol which tells to the compiler to perform some operation.

-> Java provides a rich set of operators do deal with various types of operations.

-> Sometimes we need to perform arithmetic operations then we use plus (+) operator for

addition, multiply (*) for multiplication etc.

-> Operators are always essential part of any programming language.

-> Java operators can be divided into following categories:

- Arithmetic operators
- Relation operators
- Logical operators
- Assignment operators
- Conditional operators
- Misc. operators

Arithmetic operators

Arithmetic operators are used to perform arithmetic operations like: addition, subtraction etc and helpful to solve mathematical expressions.

The below table contains Arithmetic operators.

Operator	Description
+	adds two operands
-	subtract second operands from first
*	multiply two operand
/	divide numerator by denumerator
%	remainder of division
++	Increment operator increases integer value by one
--	Decrement operator decreases integer value by one

```java
public class Demo {

    public static void main(String[] args) {

        int a = 10;
        int b = 2;

        System.out.println(a + b);

        System.out.println(a - b);

        System.out.println(a * b);

        System.out.println(a / b);

        System.out.println(a % b);

    }
}
```

Increment & Decrement Operators

In programming (Java, C, C++, JavaScript etc.), the increment operator ++ increases the

value of a variable by 1. Similarly, the decrement operator -- decreases the value of a

variable by 1.

Post-Increment (Ex: a ++): Value is first used for computing the result and then incremented.

Pre-Increment (Ex: ++ a): Value is incremented first and then the result is computed.

Post-decrement (Ex: a--): Value is first used for computing the result and then decremented

Pre-decrement (Ex: --a): Value is decremented first and then the result is computed.

Imp Points about Increment and Decrement operators

- Can only be applied to variables only
- Nesting of both operators is not allowed
- They are not operated over final variables
- Increment and Decrement Operators cannot be applied to Boolean

Relational operators

Relational operators are used to test comparison between operands or values. It can be

used to test whether two values are equal or not equal or less than or greater than etc.

The following table shows all relation operators supported by Java.

Operator	Description
==	Check if two operand are equal
!=	Check if two operand are not equal.
>	Check if operand on the left is greater than operand on the right
<	Check operand on the left is smaller than right operand
>=	check left operand is greater than or equal to right operand
<=	Check if operand on left is smaller than or equal to right operand

```
class Operations {

public static void main(String as[]) {
        int a, b;

        a=40;
        b=30;

        System.out.println("a == b = " + (a == b) );
        System.out.println("a != b = " + (a != b) );
        System.out.println("a > b = " + (a > b) );
        System.out.println("a < b = " + (a < b) );
        System.out.println("b >= a = " + (b >= a) );
        System.out.println("b <= a = " + (b <= a) );

    }
}
```

Operations.java

Logical operators

Logical Operators are used to check conditional expression. For example, we can use logical

operators in if statement to evaluate conditional based expression. We can use them into

loop as well to evaluate a condition. Java supports following 3 logical operators. Suppose we have two variables whose values

are: a=true and b=false.

Operator	Description	Example
&&	Logical AND	(a && b) is false
||	Logical OR	(a || b) is true
!	Logical NOT	(!a) is false

```
class LogicalOperators {

   public static void main(String as[]) {

        boolean a = true;
        boolean b = false;

        System.out.println("a && b = " + (a&&b));
        System.out.println("a || b = " + (a||b) );
        System.out.println("!(a && b) = " + !(a && b));

    }
}
```

LogicalOperators.java

Assignment Operators

Assignment operators are used to assign a value to a variable. It can also be used combine

with arithmetic operators to perform arithmetic operations and then assign the result to the

variable. Assignment operator supported by Java are as follows:

Operator	Description	Example
`=`	assigns values from right side operands to left side operand	`a = b`
`+=`	adds right operand to the left operand and assign the result to left	`a+=b` is same as `a=a+b`
`-=`	subtracts right operand from the left operand and assign the result to left operand	`a-=b` is same as `a=a-b`
`*=`	mutiply left operand with the right operand and assign the result to left operand	`a*=b` is same as `a=a*b`
`/=`	divides left operand with the right operand and assign the result to left operand	`a/=b` is same as `a=a/b`
`%=`	calculate modulus using two operands and assign the result to left operand	`a%=b` is same as `a=a%b`

Conditional operator

It is also known as ternary operator because it works with three operands. It is short

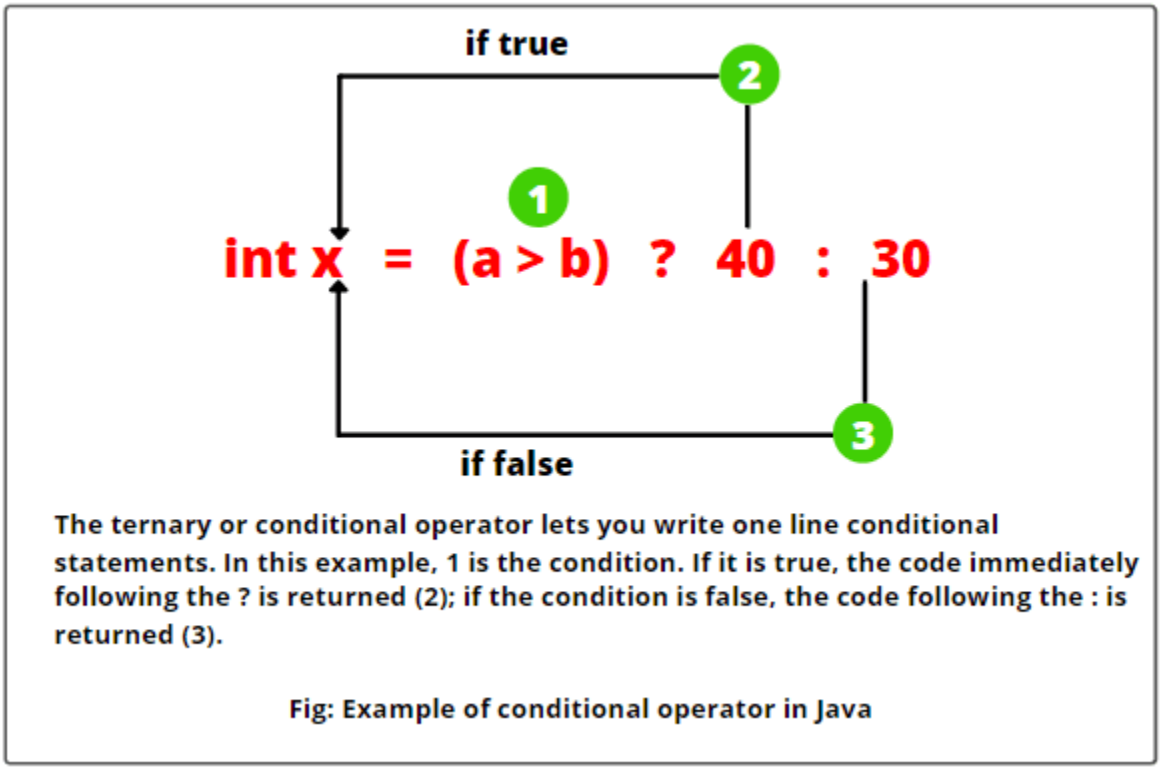
alternate of if-else statement. It can be used to evaluate Boolean expression and return

either true or false value

Syntax : epr1 ? expr2 : expr3

In ternary operator, if epr1 is true then expression evaluates after question mark (?) else

evaluates after colon (:). See the below example.



The ternary or conditional operator lets you write one line conditional statements. In this example, 1 is the condition. If it is true, the code immediately following the ? is returned (2); if the condition is false, the code following the : is returned (3).

Fig: Example of conditional operator in Java

instanceOf operator

It is a java keyword and used to test whether the given reference belongs to provided type

or not. Type can be a class or interface. It returns either true or false.

Example:

Here, we created a string reference variable that stores “Ashok IT”. Since it stores string

value so we test it using instance operator to check whether it belongs to string class or not.

See the below example

new operator

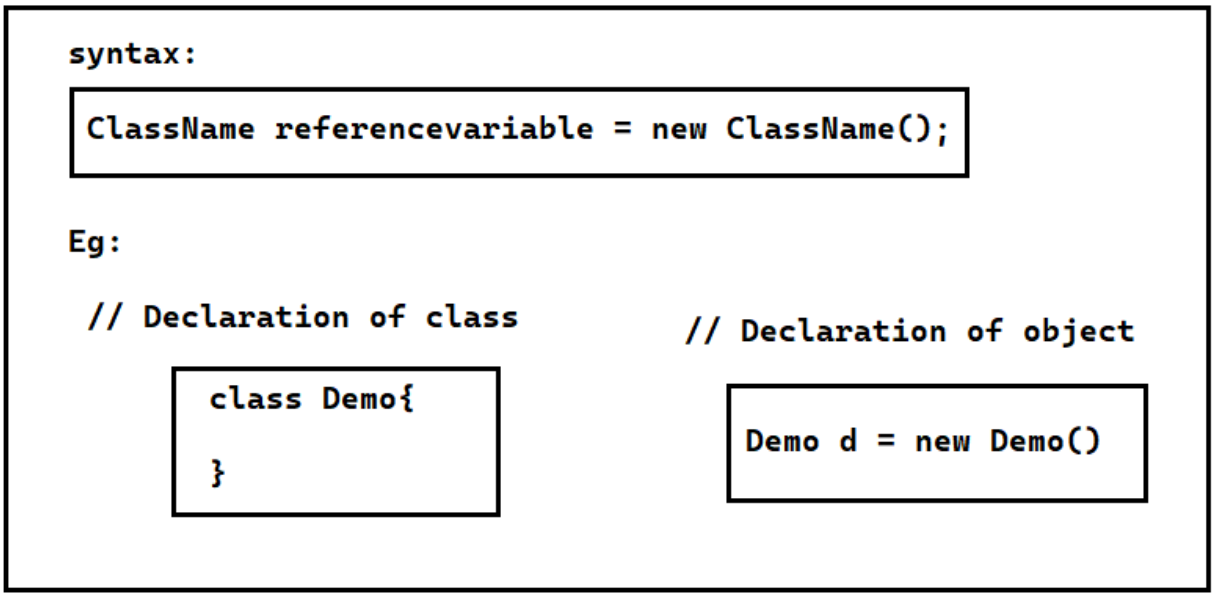
▪ new is java keyword or operator which is used to create the object

▪ we can create an object for both user defined classes and predefine classes

▪ creating an object nothing but allocating the memory so that we can use in the

application.

▪ once an object created that will be located inside the Heap memory of JVM.



. (Dot) operator

-> This operator used to access the members of the class using reference or column like

follows

Eg:

reference.variable

reference.method()

-> This operator can also be used to refer or identify the class from a package

Eg:

java.lang.NoSuchMethodError

java.lang.String

Control statements (flow control)

-> In java we can write any number of statements which are executed in sequence order by default

-> But if we want to execute java statements according to our requirement then we have to use control statements

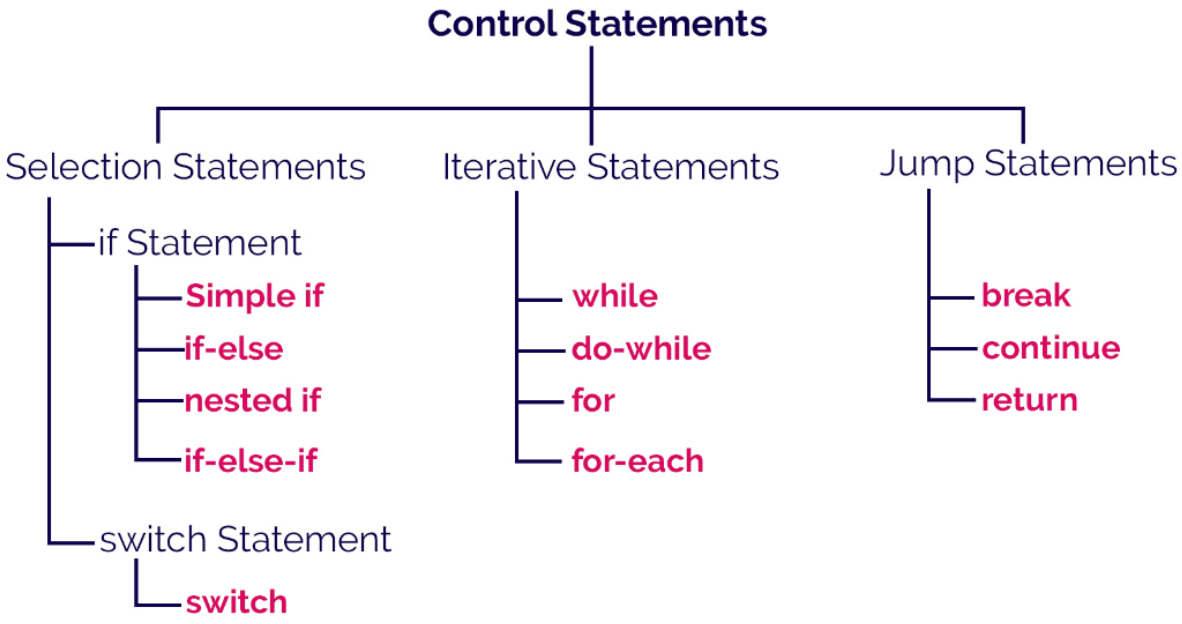
-> These statements decide whether a specific part of the code will be executed or not.

Types of Control Flow Statements

-> There are different types of control statements available in Java for different situations or conditions.

-> In java, we can majorly divide control statements into three major types:

1. Selection / Conditional statements
2. Loop statements
3. Jump / Branching / Transfer statements



Conditional / Selection Statements

-> Conditional statements are used to execute group of statements based on condition

-> Conditional statements will evaluate boolean expression to make the decision Simple 'if' statement in java

-> In java, we use if statement to test a condition and decide the execution of a block of statements based on that condition result.

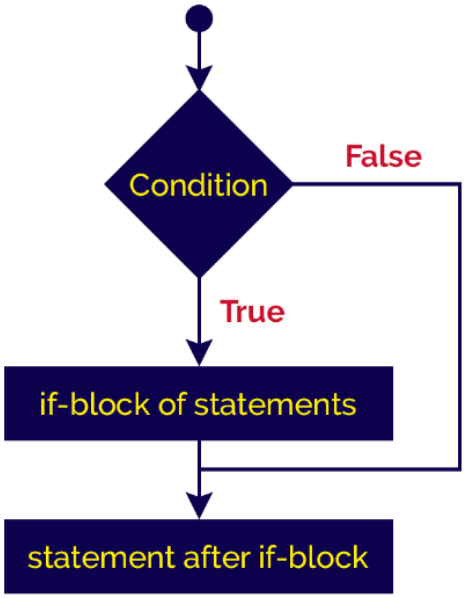
-> If the condition is True, then the block of statements is executed and if it is false, then the block of statements will be ignored.

The syntax and execution flow of if the statement is as follows.

Syntax

```
if(condition){
        if-block of statements;
        ...
}
statement after if-block;
...
```

Flow of execution



if-else statement in java

-> In java, we use the if-else statement to test a condition and pick the execution of a block of statements out of two blocks based on that condition result.

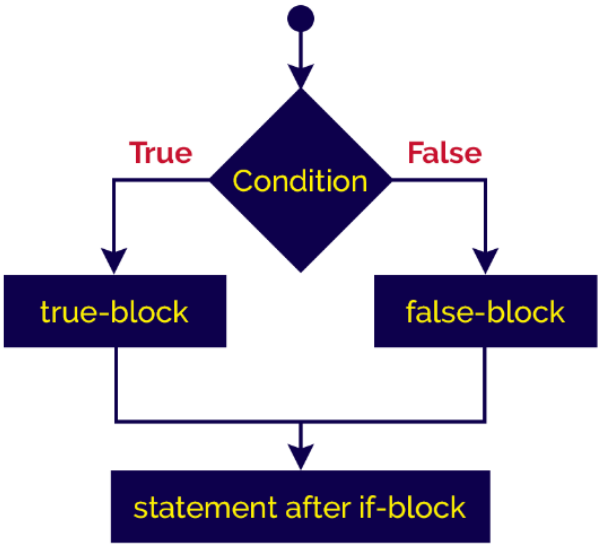
-> The if-else statement checks the given condition then decides which block of statements to be executed based on the condition result.

-> If the condition is True, then the true block of statements is executed and if it is False, then the false block of statements is executed.

Syntax

```
if(condition){
        true-block of statements;
        ...
}
else{
        false-block of statements;
        ...
}
statement after if-block;
...
```

Flow of execution



Nested if statement in java

```
if ( condition_1 ) {

    if(condition_2){
        inner if-block of statements;
        ...
    }
    ...
}
```

if...else if...else statement

if...else if statements will be used when we need to compare the value with more than 2 conditions.

They are executed from top to bottom approach. As soon as the code finds the matching condition, that block will be executed. But if no condition is matching then the last else statement will be executed.

```
if ( condition_1 ) {

   Statement 1; //if condition_1 becomes true then this will be executed

} else if ( condition_2 ) {

   Statement 2; // if condition_2 becomes true then this will be executed

} else {

   Statement 3; //executed when no matching condition found

}
```

Switch statement

-> Java switch statement compares the value and executes one of the case blocks based on the condition.

-> It is same as if…else if ladder. Below are some points to consider while working with

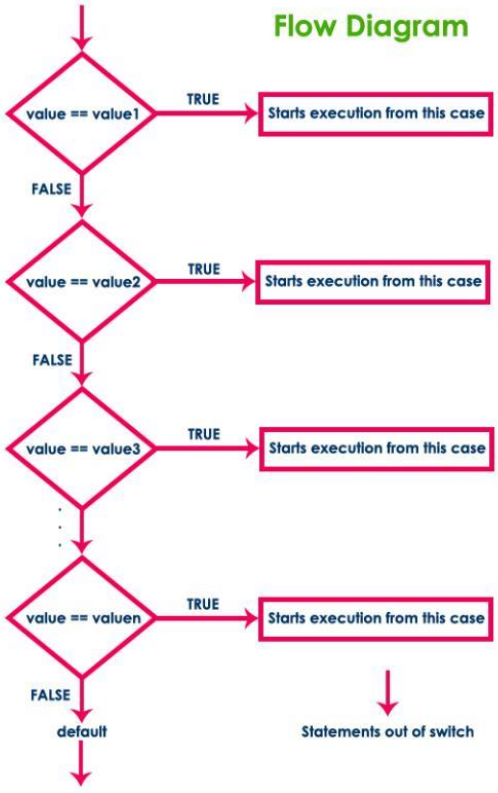
switch statements:

- There can be one or N number of cases
- The values in the case must be unique
- case value must be of the same type as expression used in switch statement
- each case statement can have break statement (it is optional)

Syntax

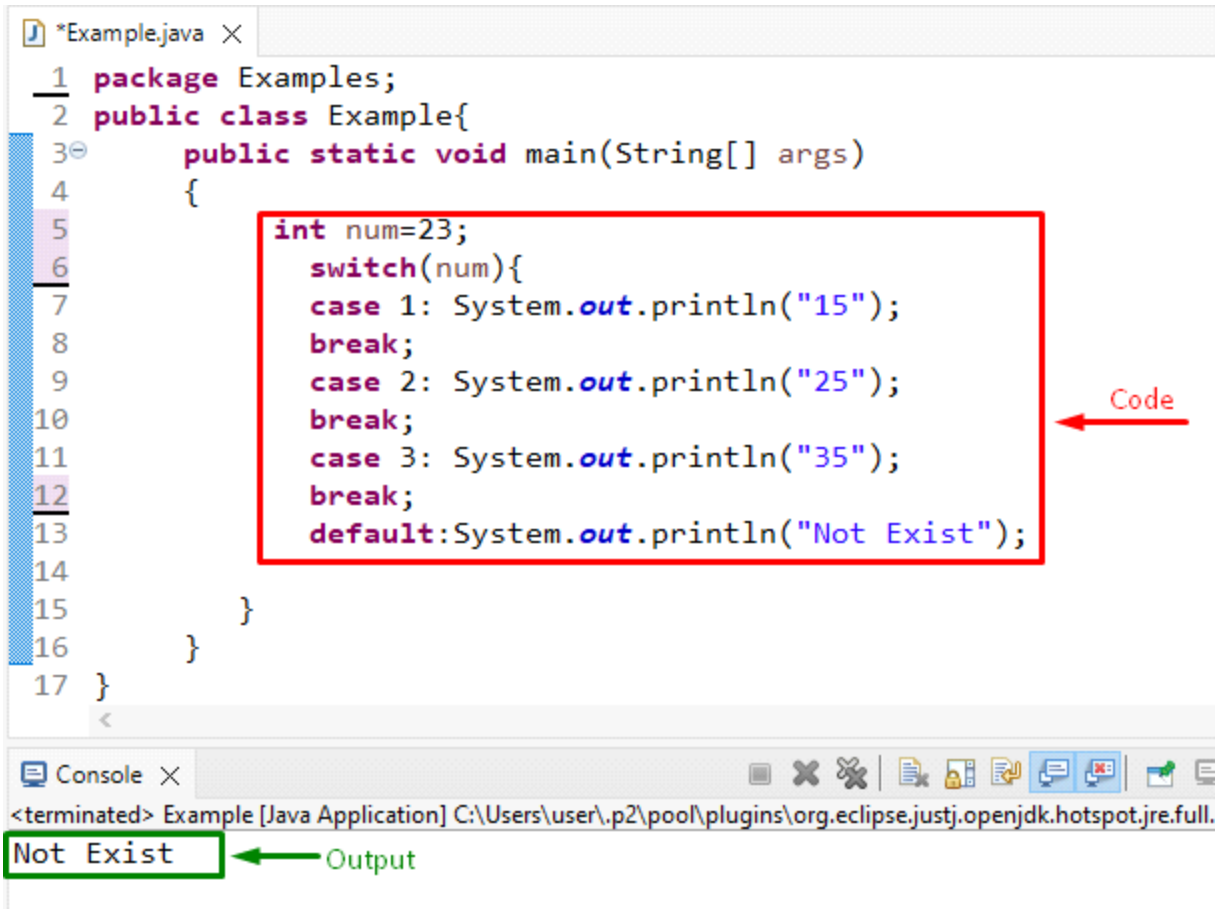
```
switch ( expression or value )
{
        case  value1:  set of statements;
                       ....
        case  value2:  set of statements;
                       ....
        case  value3:  set of statements;
                       ....
        case  value4:  set of statements;
                       ....
        case  value5:  set of statements;
                       ....
        .
        .
        default: set of statements;
}
```

Flow Diagram



```
switch(expression) {
case value1:
   //code for execution;
   break;  //optional
case value2:
 // code for execution
 break;  //optional
......
......
......
......
Case value n:
// code for execution
 break;  //optional

default:
 code for execution when none of the case is true;
}
```



Looping Statements in Java

-> Loop is an important concept of a programming that allows to iterate over the sequence of statements.

-> Loop is designed to execute particular code block till the specified condition is true or all the elements of a collection (array, list etc) are completely traversed.

-> The most common use of loop is to perform repetitive tasks. For example, if we want to

print table of a number then we need to write print statement 10 times. However, we can do the same with a single print statement by using loop.

-> Loop is designed to execute its block till the specified condition is true.

Java provides mainly three loops based on the loop structure.

1. for loop

2. while loop

3. do while loop

for loop in java

The for loop is used to execute a single statement or a block of statements repeatedly as long as the given condition is TRUE.

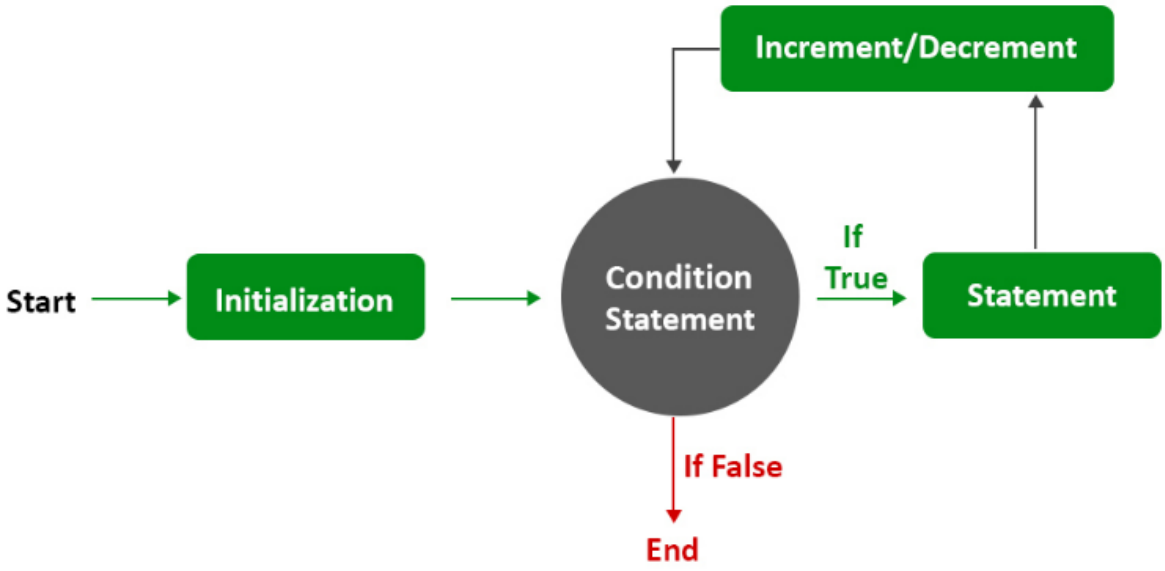
To create a for loop, we need to set the following parameters.

1) Initialization: It is the initial part, where we set initial value for the loop. It is executed only once at the starting of loop. It is optional, if we don't want to set initial value.

2) Condition: It is used to test a condition each time while executing. The execution continues until the condition is false. It is optional and if we don't specify, loop will be infinite.

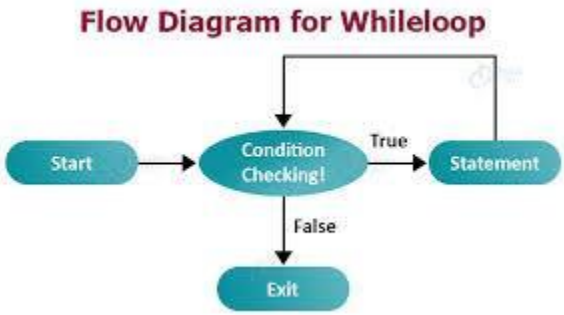
3) Statement: It is loop body and executed every time until the condition is false.

4) Increment/Decrement: It is used for set increment or decrement value for the loop.



while statement in java

The while loop is used to execute a single statement or block of statements repeatedly as long as the given condition is TRUE. The while statement is also known as Entry control looping statement. The syntax and execution flow of while statement is as follows.



WhileDemo.java

```java
public class WhileDemo {

    public static void main(String[] args) {

        int i = 1;

        while (i <= 10) {
            System.out.println(i);
            i++;
        }
    }
}
```

do-while statement in java

The do-while loop is used to execute a single statement or block of statements repeatedly as long as given the condition is TRUE. The do-while statement is also known as the Exit control looping statement. The do-while statement has the following syntax

DoWhileDemo.java

```java
public class DoWhileDemo {

    public static void main(String[] args) {
        int i = 1;

        do {
            System.out.println(i);
            i++;

        } while (i <= 10);
    }

}

```

Nested For Loop

Writing a loop inside another loop is called as Nested For Loop

NestedForLoopDemo.java

```java
public class NestedForLoopDemo {

    public static void main(String[] args) {

        for (int i = 1; i <= 5; i++) {

            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }

            System.out.println();
        }
    }
}

```

for-each loop in java

-> The Java for-each statement was introduced since Java 5.0 version.

-> It provides an approach to traverse through an array or collection in Java.

-> The for-each statement also known as enhanced for statement.

-> The for-each statement executes the block of statements for each element of the given array or collection

Branching / Transfer Statements

-> Transferring statements are the control statements which are used to transfer the control position from 1 location to another location

-> In Java we have following 3 types of transfer statements

1) break

2) continue

3) return

Break statement

-> In Java, break is a statement that is used to break current execution flow of the program.

-> We can use break statement inside loop, switch case etc.

-> If break is used inside loop then it will terminate the loop.

-> If break is used inside the innermost loop then break will terminate the innermost loop only and execution will start from the outer loop.

-> If break is used in switch case then it will terminate the execution after the matched case.

BreakDemo.java

```java
public class BreakDemo {

    public static void main(String[] args) {

        for (int i = 1; i <= 10; i++) {
            if (i == 8) {
                break;
            }
            System.out.println(i);
        }
    }
}
```

Continue Statement

-> In Java, the continue statement is used to skip the current iteration of the loop. It jumps

to the next iteration of the loop immediately.

-> We can use continue statement with for loop, while loop and do-while loop as well.

ContinueDemo.java

```java
public class ContinueDemo {

    public static void main(String[] args) {

        for (int i = 1; i <= 10; i++) {

            if (i == 5) {
                continue;
            }

            System.out.println(i);
        }
    }
}
```

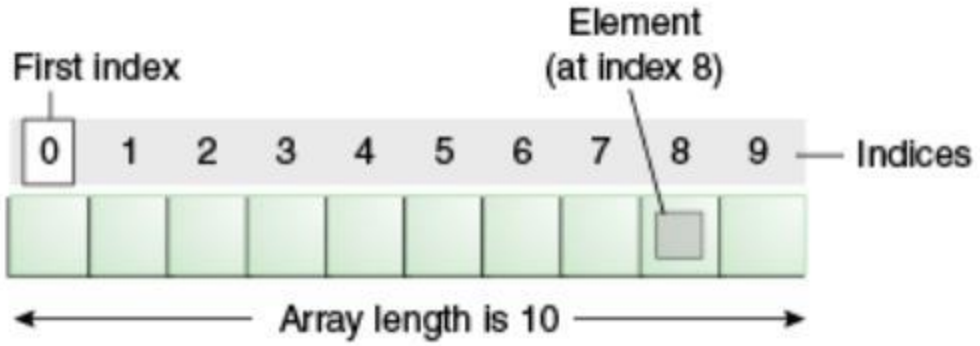
Return statement

-> return is a transferring statement which is used to stop the continuity of method execution

```
public String plot()
{
    return "Khushi coffee point";
}
```

Arrays

- Array is an object which contains elements of similar data type
- Array is a container object that hold values of homogeneous type.
- Array is also known as static data structure because size of an array must be specified at the time of its declaration.
- Array starts from zero index and goes to n-1 where n is length of the array.
- Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.
- In Java, array is treated as an object and stores into heap memory. It allows to store primitive values or reference values.



Syntax to Declare an Array in Java

```
dataType[] arr; (or)

dataType []arr; (or)

dataType arr[];
```

Instantiation of an Array in Java

```
arrayRefVar = new datatype[size];
```

-> Instantiation is a process of allocating memory to an array. At the time of instantiation, we specify the size of array to reserve memory area.

ArrayDemo.java

```java
public class ArrayDemo {

    public static void main(String[] args) {

        int[] arr = new int[5];

        for (int x : arr) {
            System.out.println(x);
        }
    }
}

```

-> In the above example, we created an array arr of int type and can store 5 elements. We iterate the array to access its elements and it prints five times zero to the console. It prints zero because we did not set values to array, so all the elements of the array initialized to 0 by default

Set Array Elements

We can set array elements either at the time of initialization or by assigning direct to its index.

```
int[] arr = {10,20,30,40,50};
```

Here, we are assigning values at the time of array creation. It is useful when we want to store static data into the array.

-> We can set value based on index position also

```
arr[1] = 105
```

ArrayDemo.java

```java
public class ArrayDemo {

    public static void main(String[] args) {

        int[] arr = { 10, 20, 30, 40, 50 };

        for (int x : arr) {
            System.out.println(x);
        }

        // assigning a value
        arr[1] = 105;

        System.out.println("element at first index: " + arr[1]);
    }
}
```

Accessing array element

We can access array elements by its index value. Either by using loop or direct index value.

We can use loop like: for, for-each or while to traverse the array elements.

ArrayDemo.java

```java
public class ArrayDemo {

    public static void main(String[] args) {

        int[] arr = { 10, 20, 30, 40, 50 };

        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }

        System.out.println("element at first index: " + arr[1]);
    }
}
```

Length Of Array In Java

The length of an array indicates the number of elements present in the array. Unlike C/C++, where we use 'sizeof' operator to get the length of the array, Java array has 'length' property. We will explore more on this property later.

ArrayDemo.java

```java
public class ArrayDemo {

    public static void main(String[] args) {
        int[] myArray = { 1, 3, 5, 7 };
        System.out.println("Size of the given array : " + myArray.length);
    }
}
```

Strings

-> String is an object that represents sequence of characters.

Ex: "hello", "ashwani"

-> In Java, String is represented by String class which is available java.lang package

-> One important thing to notice about string object is that string objects are immutable that means once a string object is created it cannot be changed.

How to create String object in Java?

-> To handle string data in Java, we need an object of string class. Basically, there are three ways to create a String object in Java.

1) By string literal.

2) By new keyword.

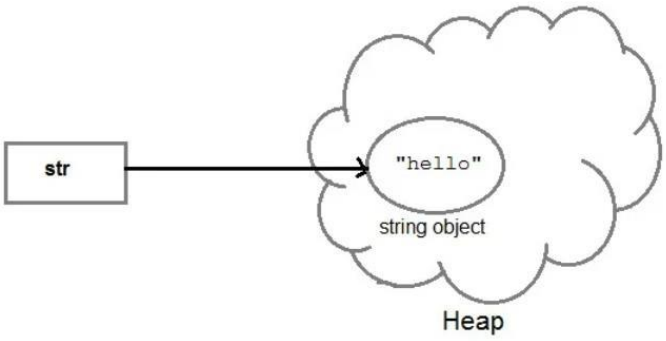
3) By converting character arrays into strings

Working with String literal

-> String literal in Java is created by using double quotes.

String str = "hello";

-> The string literal is always created in the string constant pool.



-> In Java, String constant pool is a special area that is used for storing string objects.

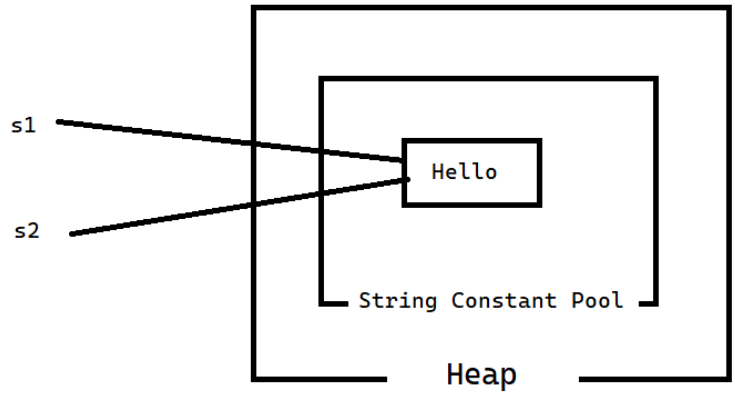
-> Whenever we create a string literal in Java, JVM checks string constant pool first. If the string already exists in string constant pool, no new string object will be created in the string pool by JVM.

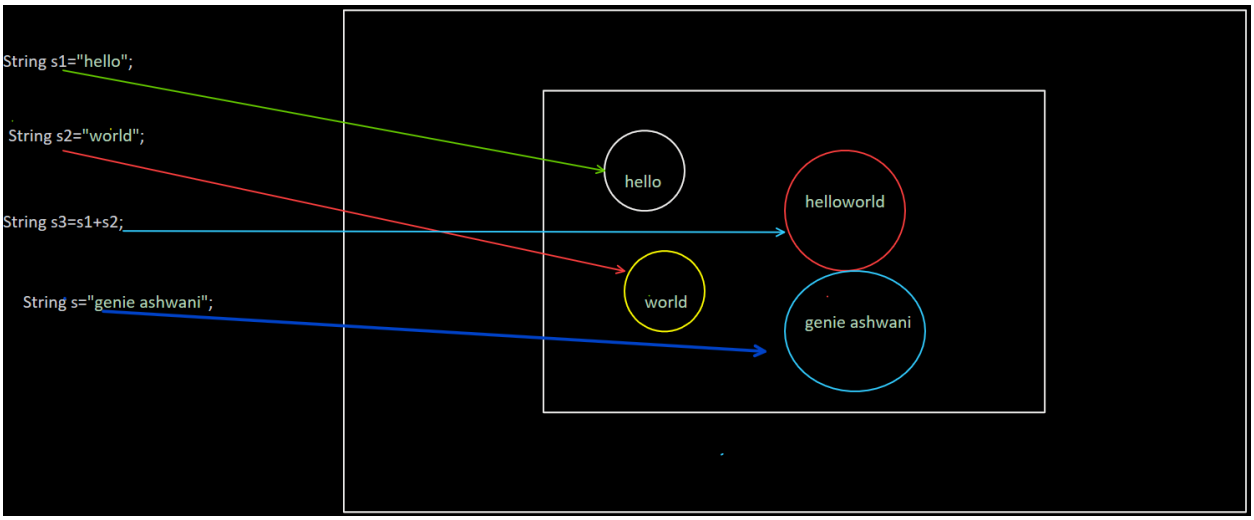
-> JVM uses the same string object by pointing a reference to it to save memory. But if string does not exist in the string pool, JVM creates a new string object and placed it in the pool.

For example:

String s1 = "Hello";

String s2 = "Hello";





Creating String Object by using new Keyword

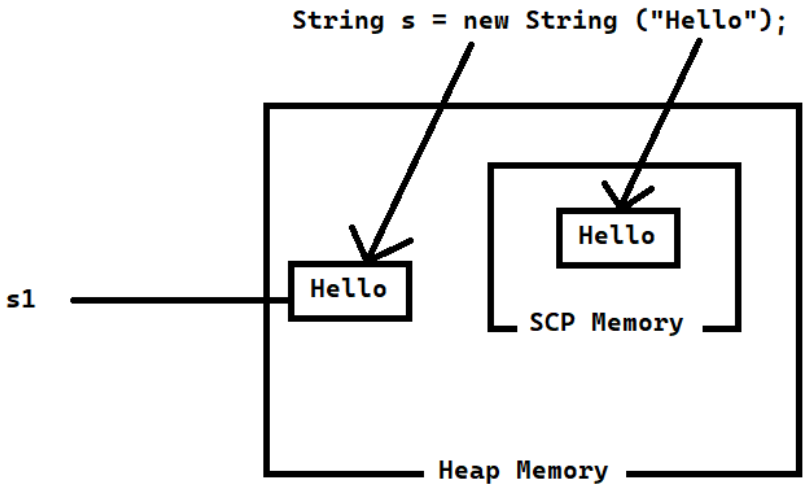
-> The second way of creating an object to string class is by using new operator.

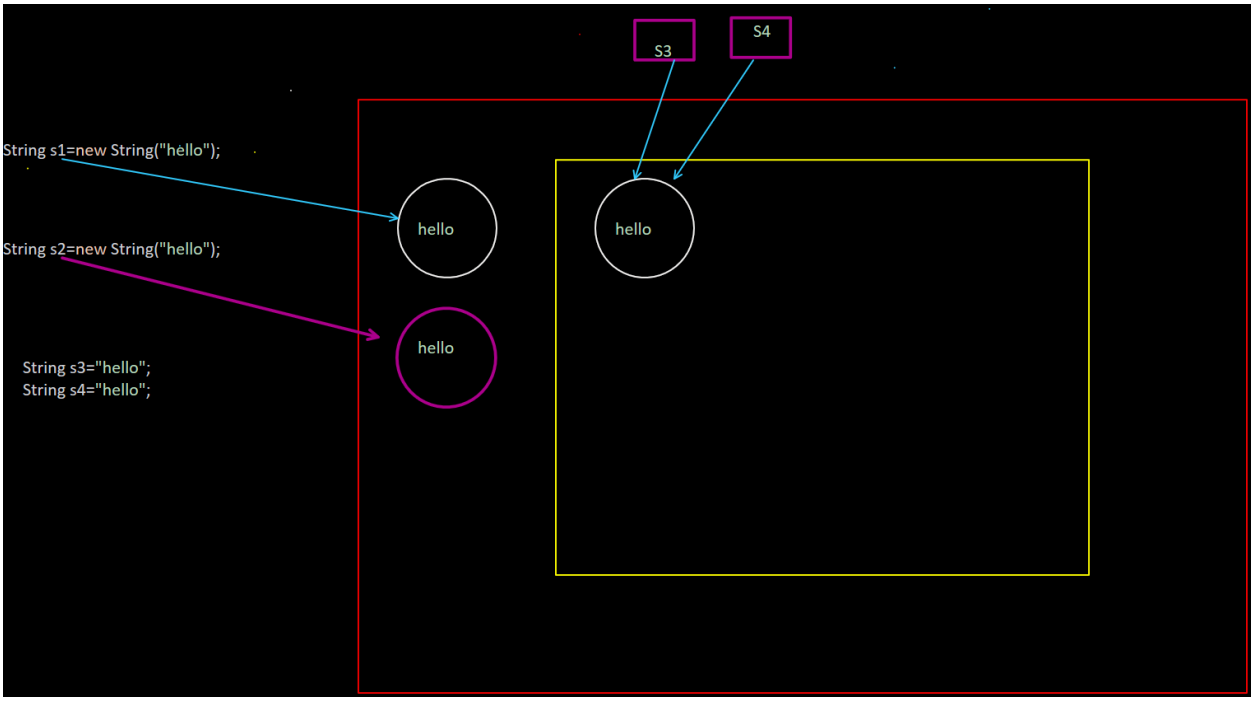
-> It is just like creating an object of any class. It can be declared as follows:

String s = new String("Hello");

-> Whenever we will create an object of string class using the new operator, JVM will create two objects. First, it will create an object in the heap area and store string “Hello” into the object.

-> After storing data into memory, JVM will point a reference variable s to that object in the heap. Allocating memory for string objects is shown in the below figure





Why String Objects are given as immutable objects

An object can be referred by multiple reference variables in this case if string objects are mutable objects, then we change the content of object automatically other references get

also modified so that string objects are given as immutable objects it means whenever any operation is done on strings it will create new object

String Methods

```
package Dec232023;

import java.util.Arrays;

public class StringFunctions {
    public static void main(String[] args) {

        //length
        String s="ashwani genie";
        System.out.println(s.length());

        //concatention
        String s1="ashwnai";
        String s2="upadhyay";
```

```
        String s3 = s1+s2;
        System.out.println(s3);

        //toCharArray()
        String s4 ="Ashwani";
        char arr[]= s4.toCharArray();
        System.out.println("to char array = "+ Arrays.toString(arr));

        //charAt()
        String s11="leela";
        System.out.println("chart at = "+s11.charAt(1));

        //compareTo()

        String s12="hello";
        String s13="hello";

       int res = s12.compareTo(s13);
        System.out.println(res);

        //== refernce com equals()

        System.out.println(s12.equals(s13));

        //contains()
        String s15="hello ashwani welcome to the development batch spark
1.0";
        System.out.println(s15.contains("ashwani"));
        System.out.println(s15.contains("Deep"));




        //indexOf()
        System.out.println(s15.indexOf('a'));

        //replace()\
        String ss="hello world";
        String res1=ss.replace("hello","ashwani");
        System.out.println(res1);

        //subString()
        String r1="hello ashwani welcome to the development batch spark 1.0";
        String r2= r1.substring(0,7); //o --- end-1
        String r3= r1.substring(4);//o --- end-1

        System.out.println(r2);
        System.out.println(r3);

    }
}
```

StringBuffer Class

-> StringBuffer class is used to create a mutable string object. It means, it can be changed after it is created.

-> It is similar to String class in Java both are used to create string, but stringbuffer object can be changed.

-> StringBuffer class is used when we have to make lot of modifications to our string.

-> It is also thread safe i.e multiple threads cannot access it simultaneously.

StringBuffer defines 4 constructors.

StringBuffer(): It creates an empty string buffer and reserves space for 16 characters.

StringBuffer(int size): It creates an empty string and takes an integer argument to set capacity of the buffer.

StringBuffer(String str): It creates a stringbuffer object from the specified string.

StringBuffer(charSequence []ch): It creates a stringbuffer object from the charsequence array.

Creating StringBuffer Object

```
public class StringBufferDemo {

    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Ashwani");
        sb.append("upadhyay");
        System.out.println(sb);
    }
}
```

Difference Between String & StringBuffer

```
public class StringAndStringBuffer {

    public static void main(String[] args) {

        //String
        String s1="hello";
        String s2 = s1.concat("world");
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s1==s2);

        StringBuffer sb=new StringBuffer("raj");
        sb.append("yadav");
        System.out.println(sb);


    }
}
```

Note: In the above program Output is such because String objects are immutable objects. Hence, if we concatenate on the same String object, it won't be altered But StringBuffer creates mutable objects. Hence, it can be altered.

StringBuffer class methods

```
package Dec242023;

public class StringBufferFunctions {
    public static void main(String[] args) {

        //append()
        StringBuffer sb=new StringBuffer("Archi");
        sb.append("jain");
        System.out.println(sb);

        //insert()
        sb.insert(2,123);
        System.out.println(sb);

        //reverse()
        sb.reverse();
        System.out.println(sb);

        //replace()
        sb.replace(6,11,"hello");
```

```
        System.out.println(sb);

        StringBuffer sb1=new StringBuffer("sdsdsdsdsd");
        System.out.println(sb1.capacity());
    }
}
```

StringBuilder class

-> StringBuilder is identical to StringBuffer except for one important difference that it is not synchronized, which means it is not thread safe.

-> StringBuilder also used for creating string object that is mutable and non synchronized.

-> The StringBuilder class provides no guarantee of synchronization.

-> StringBuffer and StringBuilder both are mutable but if synchronization is not required then it is recommended to use StringBuilder class.

-> This class is located into java.lang package and signature of the class is as:

StringBuilder Constructors

StringBuilder (): creates an empty StringBuilder and reserves room for 16 characters.

StringBuilder (int size): create an empty string and takes an integer argument to set capacity of the buffer.

StringBuilder (String str): create a StringBuilder object and initialize it with string str.

StringBuilder (CharSequence seq): It creates stringbuilder object by using CharSequence object.

Working with StringBuilder class

```
public class StringBuilderDemo {

    public static void main(String[] args) {

        StringBuilder obj = new StringBuilder("welcome to ");
        obj.append("genei ashwani");
        System.out.println(obj);
    }
}
```

Note:

-> When we want a mutable String without thread-safety then StringBuilder should be used

-> When we want a mutable String with thread-safety then StringBuffer should be used

-> When we want an Immutable object then String should be used.

Java Program with Command Line Arguments

Demo.java

```
1  public class Demo {
2
3      public static void main(String[] args) {
4
5          // convert into integer type
6          int number1 = Integer.parseInt(args[0]);
7          System.out.println("First Number: " + number1);
8
9          // convert into integer type
10         int number2 = Integer.parseInt(args[1]);
11         System.out.println("Second Number: " + number2);
12
13         int result = number1 + number2;
14         System.out.println("Addition of two numbers is: " + result);
15     }
16 }
17
```