



Class: II sem BCA 'C' section

UNIT – I: Introduction to Java

Basics of Java programming:

[Data types](#), Variables, [Operators](#), [Control structures](#) including [selection](#), [Looping](#), Java methods, [Overloading](#), [Math](#) class, [Arrays](#) in java. **Objects and Classes:** Basics of objects and classes in java, [Constructors](#), [Visibility](#) modifiers, Methods and objects, Inbuilt classes like [String](#), [String Buffer](#),

Questions

- 1.What is JVM? 2m p6
- 2.What is bytecode? Why java is platform independent? 2m p2
- 3.What are the default values of int,float and char? 2m p12
- 4.Explain features of java? 6m p3
- 5.Explain various access specifiers used in java. 6m p41
- 6.Explain the data types in java. 6m p10
- 7.Explain all types of operators with example. 10m p12
- 8.Explain general form of if and switch statements. 6m p16
- 9.Explain looping in java? 6m p19
- 10.What is class and object in java? Explain with an example. 10m. p32
- 11.Differentiate between break and continue. 2m p24
- 12.What is static data member and member functions? 2m 57
- 13.Explain method overloading with an example. 6m p25
- 14.Define constructor? How do we invoke in java? 2m p34
- 15.What is constructor overloading? Explain with an example. 6m p39
- 16.What are the different types of variables? Explain. 6m p9-10
- 17.What is Array? What are the types of arrays? Explain. 6m p11
- 18.Explain any 6 string methods with syntax and example.6m p49
- 19.How do we create instance of class in java? 2m

Ans: When we create an object, we are creating an instance of a class, therefore "instantiating" a class. Example: Student S = new Student();

Where Student is class and S is an object of Student.

- 20.What is System.in and System.out? 2m

System.in is the input stream connected to the console and System.out is the output stream connected to the console.

Ex: System.out.println("hello");

Ex: Scanner in = new Scanner(System.in);

- 21.What is the difference between string and string buffer class? 2m p55
- 22.What do you mean by command line arguments? 2m p34
- 23.Explain public static void main? 6m p4
- 24.What are separators? Describe various separators used in java. 5m p33

Introduction to java-Java technology is widely used currently. Let's start learning of java from basic questions like

What is Java? 2m- Java is a **programming language** and a **platform**.

Platform - Any hardware or software environment in which a program runs, known as a platform. Since Java has its own Runtime Environment (JRE) and API, it is called platform.

“Java is simple object-oriented , distributed , interpreted , robust , secure , architecture neutral , portable , multithreaded , high-performance and dynamic language”.

Where it is used?

- Desktop Applications such as acrobat reader, media player, antivirus etc.
- Web Applications such as irctc.co.in, javapoint.com etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games etc.

Java bytecode is the instruction set for the Java Virtual Machine. It acts similar to an assembler. As soon as a java program is compiled, java bytecode is generated. In more apt terms, java bytecode is the machine code in the form of a .class file. With the help of java bytecode we achieve platform independence in java.

Who invented Java?



James Gosling

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

- 1) **James Gosling, Mike Sheridan, and Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
- 2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.
- 3) Firstly, it was called "**Greentalk**" by James Gosling and file extension was .gt.
- 4) After that, it was called **Oak** and was developed as a part of the Green project.

How java differs from C++? 5m

How java differs from C++?	
C++	Java
Compiled to machine code	Compiled to byte code.
Supports pointer	Does not supports pointers
Global variables are allowed.	No global variable concept.
Multiple inheritance – A class can have two or more super classes.	Multiple inheritance is attained indirectly with the help of interface concept.
Supports operator overloading	Does not supports operator overloading.
ASCII – character set.	Unicode – character set.

Features of Java 5marks

1. Object Oriented

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

2. Platform Independent

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

3. Architecture-neutral

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

4. Portable

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

5. Robust

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

6. Multithreaded

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

7. Secure

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

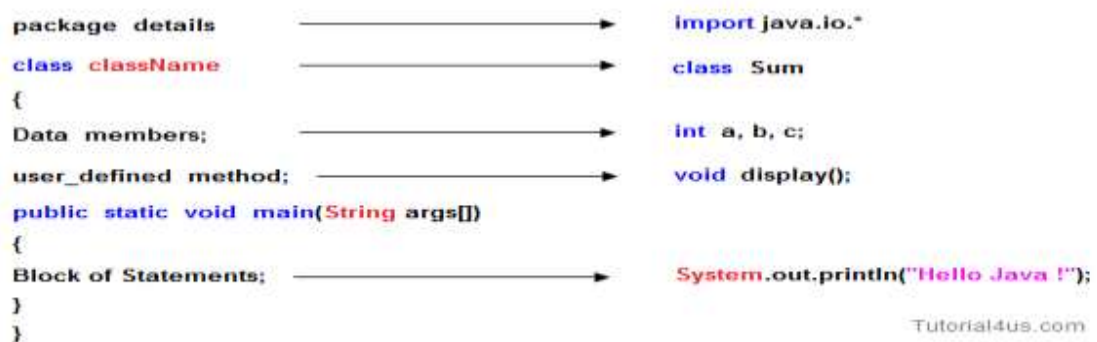
Explain Java program structure. 5m

Explain the line “public static void main” 5m

Structure of Java Program

Structure of a java program is the standard format released by Language developer to the Industry programmer.

Sun Micro System has prescribed the following structure for the java programmers for developing java application.



- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The

main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.

- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument. We will learn it later.

Simple Program of Java

```
class Simple
{
    public static void main(String args[])
    {
        System.out.println("Hello Java");
    }
}
```

save this file as Simple.java

To compile: javac Simple.java

To execute: java Simple

Output:Hello Java

Understanding first java program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

- To write the simple program, open notepad by **start menu -> All Programs -> Accessories -> notepad** and write simple program as displayed below:

How to set path in Java

There are 2 ways to set java path:

1. temporary
2. permanent

1) How to set Temporary Path of JDK in Windows

To set the temporary path of JDK, you need to follow following steps:

- Open command prompt
- copy the path of jdk/bin directory
- write in command prompt: path=copied_path

For Example:

```
path= C:\Program Files\Java\jdk1.6.0_23\bin
```

2) How to set Permanent Path of JDK in Windows

For setting the permanent path of JDK, you need to follow these steps:

- Go to MyComputer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok

Internal Details of Hello Java Program

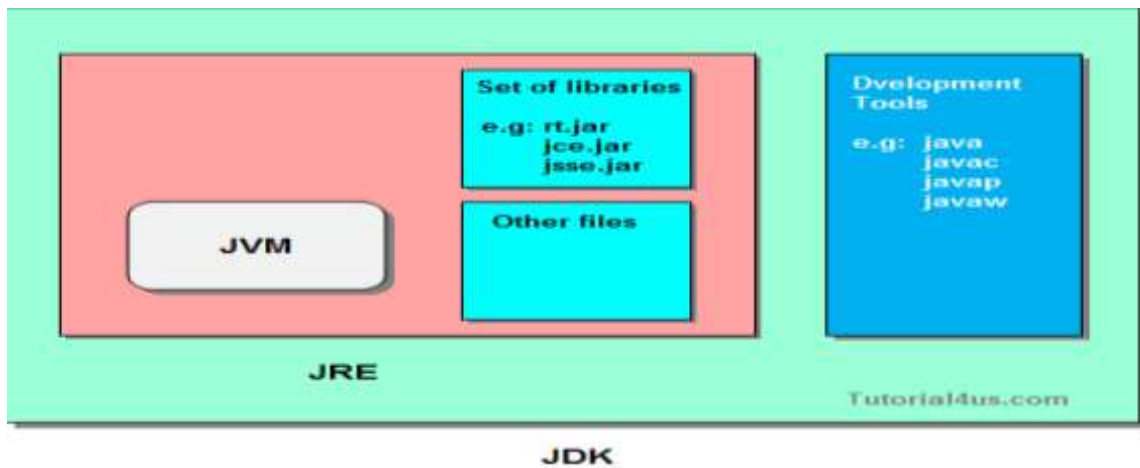
What happens at compile time?

At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.

To compile:	javac Hard.java
To execute:	java Simple

Difference between JDK, JRE and JVM

Sr. No.	Key	JDK	JRE	JVM
1	Definition	JDK (Java Development Kit) is a software development kit to develop applications in Java.	JRE (Java Runtime Environment) is the implementation of JVM and is defined as a software package that provides Java class libraries, along with JVM, and other components .	JVM (Java Virtual Machine) is an abstract machine that is platform-dependent, It is a specification that provides runtime environment in which java bytecode can be executed.
2	Prime functionality	used for code execution and has prime functionality of development.	Used for creating environment for code execution.	specifies all the implementations and responsible to provide these implementations to JRE.
3	Platform Independence	platform dependent i.e for different platforms different JDK required.	platform dependent.	platform dependent.
4	Tools	contains tools for developing, debugging and monitoring java application.	contain tools such as compiler or debugger etc.	JVM does not include software development tools.
5	Implementation	JDK = JRE+ Development tools	JRE =JVM + Libraries to run the application	JVM = Only Runtime environment for executing the Java byte code.



CONSTANTS, VARIABLES AND DATA TYPES

Constants

Constants in java are fixed values those are not changed during the Execution of program java
Types

- **Integer Constants**
- **Real Constants**
- **Single Character Constants**
- **String Constants**
- **Backslash Character Constants**

Integer Constants

Integer Constants refers to a Sequence of digits which Includes only negative or positive Values and many other things those are as follows:-

- An Integer Constant must have at Least one Digit.
- it must not have a Decimal value.
- it could be either positive or Negative.
- if no sign is Specified then it should be treated as Positive.
- No Spaces and Commas are allowed in Name.

Ex: valid 123, -321, 0

Invalid \$123, 1 23, ab12

Real Constants

- A Real Constant must have at Least one Digit.
- it must have a Decimal value.
- it could be either positive or Negative.
- if no sign is Specified then it should be treated as Positive.
- No Spaces and Commas are allowed in Name.

Ex: 251, 234.890 etc are Real Constants.

Single Character Constants

A Character is Single Alphabet a single digit or a Single Symbol that is enclosed within Single inverted commas.

Ex: 'S' , '3' etc.

String Constants

String is a Sequence of Characters Enclosed between double Quotes These Characters may be digits ,Alphabets

Ex: "Hello" , "1234" etc.

Backslash Character Constants

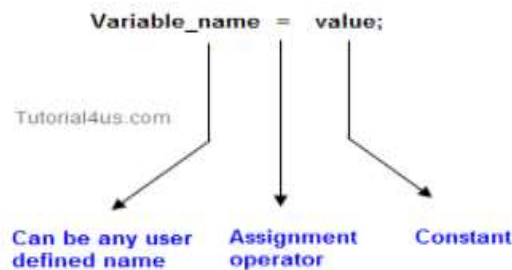
Backslash Constants those are used in output methods

Ex: \t For Tab (Five Spaces in one Time)

\b Back Space etc.

Variable

Variable is an identifier which holds data or another one variable is an identifier whose value can be changed at the execution time of program. Variable is an identifier which can be used to identify input data in a program.



`Variable_name = value;`

Explain Scope of Variables. 5m

In programming, a variable can be declared and defined inside a class, method, or block. It defines the scope of the variable i.e. the visibility or accessibility of a variable

There are three types of Variables Those are as follows:-

Variable Type	Scope	Lifetime
Instance variable	Troughout the class except in static methods	Until the object is available in the memory
Class variable	Troughout the class	Until the end of the program
Local variable	Within the block in which it is declared	Until the control leaves the block in which it is declared

```

class A
{
    int data=50;        //instance variable
    static int m=100;   //static or class variable

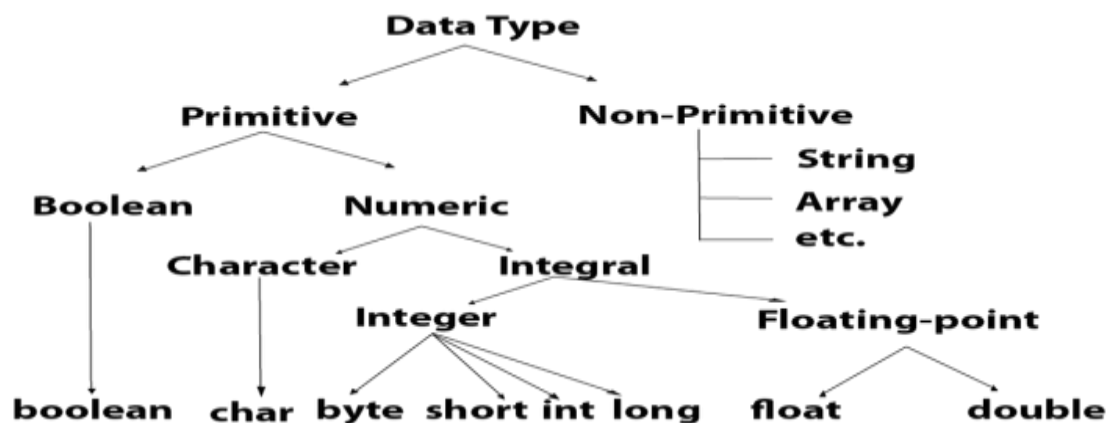
    void method()
    {
        int n=90;      //local variable
    }
}

//end of class

```

Explain Data Types in Java. 5m

- primitive data types
- non-primitive data types



PRIMITIVE DATA TYPE

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

NON-PRIMITIVE DATA TYPES

Class – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

Syntax

```
class <class_name>
{
    data member;
    method;
}
```

Arrays

is an object which contains elements of a similar data type.

1. dataType[] arr; (or)
2. dataType []arr; (or)
3. dataType arr[];

String is a sequence of characters. But in java, string is an object. String class is used to create string object.

Interface

An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve abstraction.

```
interface <interface_name>
{
    // declare constant fields
    // declare methods that abstract
}
```

Why char uses 2 byte in java and what is \u0000 ? 2m

because java uses unicode system rather than ASCII code system. \u0000 is the lowest range of unicode system.

What are the STANDARD DEFAULT VALUES IN JAVA. 2m

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Explain the various Operators in Java 10m

Operator is a special symbol that tells the compiler to perform specific mathematical or logical operations.

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

Arithmetic Operators

Given table shows all the Arithmetic operator supported by Java Language. Lets suppose variable **A** hold 8 and **B** hold 3.

Operator	Example (int A=8, B=3)	Result
+	A+B	11
-	A-B	5
*	A*B	24
/	A/B	2
%	A%4	0

Relational Operators

Which can be used to check the Condition, it always return true or false. Lets suppose variable **A** hold 8 and **B** hold 3.

Operators	Example (int A=8, B=3)	Result
<	A<B	False
<=	A<=10	True
>	A>B	True
>=	A<=B	False
==	A== B	False
!=	A!=(-4)	True

Logical Operator

Which can be used to combine more than one Condition?. Suppose you want to combined two conditions **A<B** and **B>C**, then you need to use **Logical Operator** like (A<B) && (B>C). Here **&&** is Logical Operator.

Operator	Example (int A=8, B=3, C=-10)	Result
&&	(A<B) && (B>C)	False
	(B!=C) (A==B)	True
!	!(B<=-A)	True

Truth table of Logical Operator

C1	C2	C1 && C2	C1 C2	!C1	!C2
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

Assignment operators

Which can be used to assign a value to a variable. Lets suppose variable **A** hold 8 and **B** hold 3.

Operator	Example (int A=8, B=3)	Result
+=	A+=B or A=A+B	11
-=	A-=3 or A=A-3	5
=	A=7 or A=A*7	56
/=	A/=B or A=A/B	2
%=	A%=5 or A=A%5	3
=a=b	Value of b will be assigned to a	

Ternary or conditinal Operator

If any operator is used on three operands or variable is known as Ternary Operator. It can be represented with **? : .** It is also called as conditional operator

Advantage of Ternary Operator

Using **?:** reduce the number of line codes and improve the performance of application.

Syntax

```
expression-1 ? expression-2 : expression-3
```

In the above symbol expression-1 is condition and expression-2 and expression-3 will be either value or variable or statement or any mathematical expression. If condition will be true expression-2 will be execute otherwise expression-3 will be executed.

Syntax

```
a<b ? printf("a is less") : printf("a is greater");
```

Increment and Decrement Operators:

Expression	Process	Example	end result
A++	Add 1 to a variable after use.	int A=10,B; B=A++;	A=11 B=10
++A	Add 1 to a variable before use.	int A=10,B; B=++A;	A=11 B=11
A--	Subtract 1 from a variable after use.	int A=10,B; B=A--;	A=9 B=10
--A	Subtract 1 from a variable before use.	int A=10,B; B=--A;	A=9 B=9

Bit Wise Operators:

Bit wise operator execute single bit of their operands. Following table shows bit wise operator:

Operator	Importance/ significance
	Bitwise OR
&	Bitwise AND
&=	Bitwise AND assignment
=	Bitwise OR assignment
^	Bitwise Exclusive OR
<<	Left shift
>>	Right shift
~	One's complement

Special Operators

- Instance of operator
- dot operator

instance of operator is used to test whether the object is an instance of the specified type (class or subclass or interface).

Dot operator

This operator is used to access the variables and methods of a class.

Example 1

student.mark

Here we are accessing the variable “mark” of the “student” object

Example 2

student.getMarks()

Here we are accessing the method “getMarks” of the “student” object.

This operator also can be used to access classes, packages etc.

CONTROL STATEMENTS 5M

Decision Making Statements

if-else Statement The Java if statement is used to test the condition. It checks boolean condition: true or false. There are various types of if statement in java.

- if statement and if-else
- if-else-if ladder
- nested if statement
- switch

if Statement The Java if statement tests the condition. It executes the if block if condition is true.

syntax

if(condition)

{

//code to be executed

}

Java if-else Statement The Java if-else statement also tests the condition. It executes the if block if condition is true otherwise else block is executed.

Syntax:

if(condition)

{

//code if condition is true

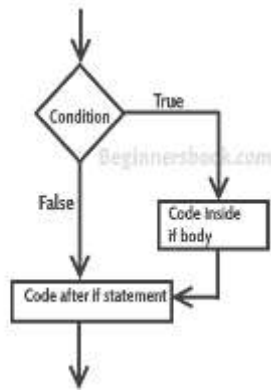
}

else

{

//code if condition is false

}



```

class IfExample
{
    public static void main(String[] args)
    {
        int age=20;
        if(age>18)
        {
            System.out.print("eligible for vote");
        }
        Else
        {
            System.out.print("Not eligible for vote");
        }
    }
}

```

Java if-else-if ladder Statement

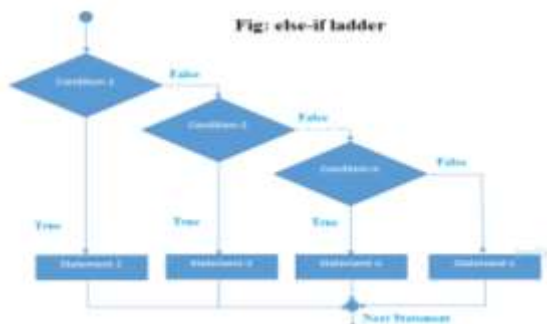
if-else-if statement is used when we need to check multiple conditions. In this statement we have only one “if” and one “else”, however we can have multiple “else if”. It is also known as **if else if ladder**.

Syntax:

```

if(condition1)
{ //code to be executed if condition1 is true
}
else if(condition2)
{ //code to be executed if condition2 is true
}
else if(condition3)
{ //code to be executed if condition3 is true
}
...
else
{ //code to be executed if all the conditions are false
}

```



```

public class Test {
    public static void main(String[] args) {
        int A=30;
        if (A>50) {
            System.out.println("A is greater than 50");
        }
        else if (A>20) {
            System.out.println("A is greater than 20 and less than 50");
        }
        else if (A>0) {
            System.out.println("A is greater than 0 and less than 20");
        }
        else {
            System.out.println("A is either 0 or less than 0");
        }
    }
}

```

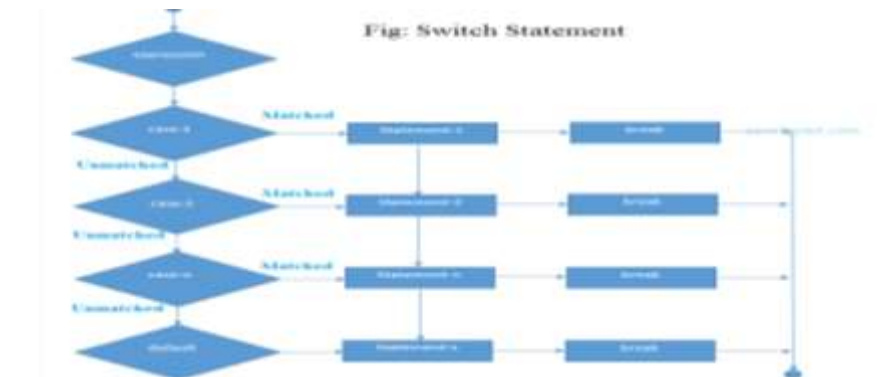
OUTPUT:
A is greater than 0 and less than 20

Java Switch Statement

- The switch statement is a multi-way branch statement.
- It executes one statement from multiple conditions.
- It is like an if-else-if ladder statement.
- It provides an easy way to dispatch execution to different parts of code based on the value of the expression

Syntax:

```
switch(expression)
{
case value1:    //code to be executed;
               break; //optional
case value2:    //code to be executed;
               break; //optional
.....
               default:    code to be executed if all cases are not matched;
}
```



```
switch (ch)
{
    case 1: System.out.print("Monday");
           break;
    case 2: System.out.print("Tuesday");
           break;
    case 3: System.out.print("Wednesday");
           break;
    case 4: System.out.print("Thursday");
           break;
    case 5: System.out.print("Friday");
           break;
    case 6: System.out.print("Saturday");
           break;
}
```

```

    default: System.out.print("invalid");
}

```

Loops in Java

Loops are used to execute a set of statements repeatedly until a particular condition is satisfied. In Java we have three types of basic loops: **for**, **while** and **do-while**.

In Java. There are three types of for loops in java.

- Simple For Loop
- For-each or Enhanced For Loop
- Labeled For Loop

Java Simple For Loop

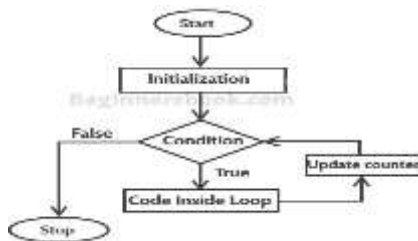
We can initialize variable, check condition and increment/decrement value.

Syntax:

```

for(initialization; condition ; increment/decrement)
{ statement(s);}

```



Example:

```

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

```

Java for-each Loop

The for-each loop is used to traverse array or collection in java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

It works on elements basis not index. It returns element one by one in the defined variable.

Syntax:

```
for(Type var : array)
{
    //code to be executed
}
```

Example:

```
int arr[]={ 12,23,44,56,78 };
for(int i:arr)
{
    System.out.print(i);
}
```

OUTPUT

12 23 44 56 78

Java Labeled For Loop

We can have name of each for loop. we use label before the for loop. It is useful if we have nested for loop so that we can break/continue specific for loop.

Normally, break and continue keywords breaks/continues the inner most for loop only.

Syntax:

labelname:

```
for(initialization;condition;incr/decr)
{
    //code to be executed
}
```

Example:

```
// the for loop is labeled as first
first:
for( int i = 1; i < 5; i++)
{

    // the for loop is labeled as second
second:
```

```

for(int j = 1; j < 3; j ++ )
{

    System.out.println("i = " + i + "; j = " +j);

    // the break statement terminates the loop labeled as second
    if ( i == 2)
        break second;
}
}

```

Java Infinitive For Loop

If you use two semicolons ;; in the for loop, it will be infinitive for loop.

Syntax:

```

for(;;)
{
    //code to be executed
}

```

Example:

```

public class ForExample
{
    public static void main(String[] args)
    {
        for(;;)
        {
            System.out.println("infinitive loop");
        }
    }
}

```

Output:

```

infinitive loop
infinitive loop
infinitive loop
infinitive loop

```

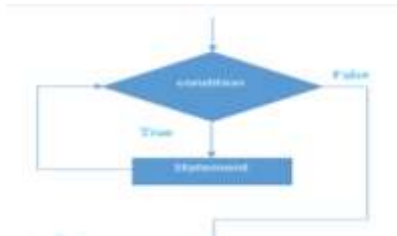
infinite loop
ctrl+c

Java While Loop

The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

Syntax:

```
while(condition)
{
    //code to be executed
}
```



Example:

```
int i=1;
while(i<=10)
{
    System.out.println(i);
    i++;
}
```

Infinite While Loop

If you pass **true** in the while loop, it will be infinite while loop.

Syntax:

```
while(true)
{
    //code to be executed
}
```

Example:

```

while(true)
{
    System.out.println("infinite while loop");
}

```

Java do-while Loop

The Java do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

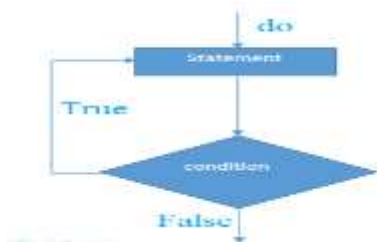
The Java do-while loop is executed at least once because condition is checked after loop body.

Syntax:

```

do{
    //code to be executed
}while(condition);

```

**Example:**

```

int i=1;
do
{
    System.out.println(i);
    i++;
}while(i<=10);

```

Java Break Statement

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

The Java break is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

Syntax:

jump-statement;

break;



Java Break Statement with Loop

Example:

```

for(int i=1;i<=10;i++)
{
    if(i==5)
    {
        break;
    }
}
  
```

Java Continue Statement

The continue statement is used in loop control structure when you need to immediately jump to the next iteration of the loop. It can be used with for loop or while loop.

The Java continue statement is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

Syntax:

continue;

Example:

```

public class ContinueExample
{
  
```



```

public static void main(String[] args)
{
    for(int i=1;i<=10;i++)
    {
        if(i==5)
            continue;

        System.out.println(i);
    }
}

```

Output:

```

1
2
3
4
6
7
8
9
10

```

OVERLOADING

- In Java, **two or more methods may have the same name if they differ in parameters** .
- These methods are called overloaded methods and this feature is called .
- If a class has multiple methods having same name but difference in parameters, it is known as Method Overloading.
- Overloading concept helps in achieving the compile-time polymorphism. This is called early or static binding.

```

void func() { ... }
void func(int a) { ... }
float func(double a) { ... }
float func(int a, float b) { ... }

```

- Here, the func() method is overloaded. These methods have the same name but accept different arguments.

Method overloading If a class has multiple methods having same name but difference in parameters, it is known as **Method Overloading**. Overloading concept helps in achieving the compile-time polymorphism. This is called early or static binding.

Advantages:

1. Method overloading **increases the readability of the program.**
2. This provides flexibility to programmers so that they can call the same method for different types of data.
3. This makes the code look clean.
4. This reduces the execution time because the binding is done in compilation time itself

//LAB5 - Write a java program to add two integers and two float numbers Use Method overloading.

```
class calculate
{
    void sum (int a, int b)
    {
        System.out.println("sum is" +(a+b)) ;
    }
    void sum (float a, float b)
    {
        System.out.println("sum is" +(a+b));
    }
    Public static void main (String[] args)
    {
        Calculate cal = new Calculate();
        cal.sum (8,5);    //sum(int a, int b) is method is called.
        cal.sum (4.6f, 3.8f); //sum(float a, float b) is called.
    }
}
```

EXPLAIN MATH CLASS FUNCTIONS 5M

Math Class methods helps to perform the numeric operations like square, square root, cube, cube root, exponential and trigonometric operations

MATHEMATICAL FUNCTIONS

- ▶ JAVA supports basic math functions through Math class defined in the java.lang.package.
- ▶ The functions should be used as follows:-

Math.function_name();

→ java.lang.Math

- ▶ Example: `y = Math.sqrt(x);`

Function
name

MATHEMATICAL FUNCTIONS

FUNCTIONS	ACTION
sin(x)	Returns the sine of the angle x in radians
cos(x)	Returns the cosine of the angle x in radians
tan(x)	Returns the tangent of the angle x in radians
asin(x)	Returns the angle whose sine is y
atan2(x,y)	Returns the angle whose tangent is x/y
pow(x,y)	Returns x^y
exp(x)	Returns e^x
log(x)	Returns the natural logarithm of x
sqrt(x)	Returns square root of x
abs(x)	Returns absolute of x
max(a,b)	Returns maximum of a and b

WAP TO DEMONSTRATE MATH FUNCTIONS 5M

WAP TO DEMONSTRATE MATH FUNCTIONS

```
import java.lang.Math.*;
class Math1
{
    public static void main(String args[])
    {
        int i1 = 2;
        int i2 = -4;
```

```

double d1 = 84.6;
double d2 = 2.4;
System.out.println("Absolute value of i1 " + Math.abs(i1));
System.out.println(Math.abs(i2));
System.out.println(Math.round(d1));
System.out.println(Math.round(d2));
System.out.println(Math.ceil(d1));
System.out.println(Math.floor(d1));
System.out.println(Math.min(i1, i2));
System.out.println(Math.max(i1, i2));
System.out.println(Math.exp(d2));
System.out.println(Math.log(d2));
System.out.println(Math.pow(5.0, 3.0));
System.out.println(Math.sqrt(16));
}
}

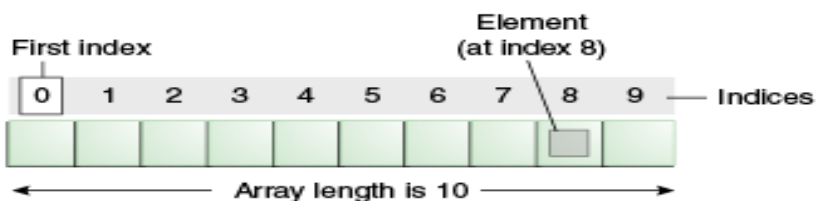
```

Java Array

Array is a collection of similar type of elements that have contiguous memory location.

Java array is an object that contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

Array in java is index based, first element of the array is stored at 0 index.



Benefits

- Easier access to any element using the index.
- Easy to manipulate and store large data.

Disadvantages

- Fixed size. Can not be increased or decrease once declared.
- Can store a single type of primitives only.

Explain the types of arrays in Java. 5m

There are two types of arrays in Java they are –

Single dimensional array – A single dimensional array of Java is a normal array where, the array contains sequential elements (of same type) –

```
int[] myArray = { 10, 20, 30, 40 }
```

Example:

```
class Single
{
    public static void main(String args[])
    {
        int a[]=new int[5];//declaration and instantiation
        a[0]=10;//initialization
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;
        for(int i=0;i<a.length;i++)
            System.out.println(a[i]);
    }
}
```

Multi-dimensional array – A multi-dimensional array in Java is an array of arrays. A two dimensional array is an array of one dimensional arrays and a three dimensional array is an array of two dimensional arrays.

```
class Multi
{
    public static void main(String[] args)
    {
        int[][] m = { { 1,2},{2,3}, {3,4} };

        for(int i = 0 ; i < 3 ; i++)
        {
            for(int j = 0 ; j < 2; j++)
            {
                System.out.print(m[i][j] + " ");
            }
        }
    }
}
```

```

        System.out.println();
    }
}
}

```

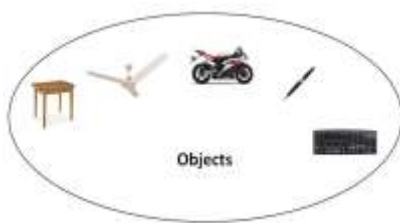
Java OOPs Concepts

Object Oriented Programming is a paradigm that provides many concepts such as inheritance, data binding, polymorphism etc.

Simula is considered as the first object-oriented programming language. The programming paradigm where everything is represented as an object, is known as truly object-oriented programming language.

Smalltalk is considered as the first truly object-oriented programming language.

OOPs (Object Oriented Programming System)



Object means a real word entity such as pen, chair, table etc.

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Object

Any entity that has state and behavior is known as an object.

For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

Class

Collection of objects is called class. It is a logical entity.

Inheritance *When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.*



Polymorphism

When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meow, dog barks woof etc.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.

Encapsulation

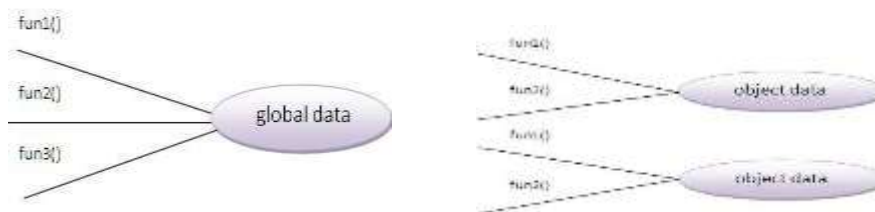
Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.



Advantage of OOPs over Procedure-oriented programming language

- 1) OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
- 2) OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
- 3) OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.



Object and Class in Java

- **Object** – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
- **Class** – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.
- Syntax to declare a class:

```
class <class_name>
{
    data member;
    method;
}
```

Example:

```
class Dog
{
    String breed;
    int age;
    String color;

    void barking()
    {
    }
    void hungry()
    {
    }
}
```



```
}  
  
}
```

An object has three characteristics:

- state: represents data (value) of an object.
- behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
- identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Java is a Strongly typed Language. why? 2m

Every variable and expression has a strongly defined type. All assignments are checked for type compatibility. Java compiler checks all expressions and parameters to ensure that the types are compatible.

A strongly-typed programming language is one in which each type of data (such as integer, character, etc.) is predefined as part of the programming language and all constants or variables defined for a given program must be described with one of the data types.

In Java, when a variable is declared, it must be informed to the compiler what data type the variable stores like integer, float, double or string etc.

For Example, consider the simple code snippet :

```
int age = 20;  
String name = "Tridib";  
boolean ans = true;
```

Separators in Java

Symbols used to indicate groups of code which are divided and arranged. They basically define the shape and function of our code.

Separator	Name	Use
.	Period	It is used to separate the package name from sub-package name & class name. It is also used to separate variable or method from its object or instance.
,	Comma	It is used to separate the consecutive parameters in the method definition. It is also used to separate the consecutive variables of same type while declaration.
;	Semicolon	It is used to terminate the statement in Java.
()	Parenthesis	This holds the list of parameters in method definition. Also used in control statements & type casting.
{ }	Braces	This is used to define the block/scope of code, class, methods.
[]	Brackets	It is used in array declaration.
Separators in Java		

Java Command Line Arguments


Command Line Arguments in Java

If any input value is passed through the command prompt at the time of running of the program is known as **command line argument** by default every command line argument will be treated as string value and those are stored in a string array of main() method.

Syntax for Compile and Run CMD programs

```
Compile By -> javac Mainclass.java
Run By -> java Mainclass value1 value2 value3 .....
```

```
javac Mainclass.java
java Mainclass value1 value2 value3.....
```



Command Line Arguments

CONSTRUCTORS IN JAVA

Important qns

Define constructor. How do we invoke constructor in java? 2m

What is the difference between constructor and method. 2m,5m

What is constructor overloading? Wap to demonstrate the same. 5m

What is default constructor and parameterized constructor? 2m

In Java, constructor is a block of codes similar to method. It is called when an instance of object is created and memory is allocated for the object.

It is a special type of method which is used to initialize the object.

It will be invoked at the time of object creation.

SYNTAX:

```
<class_name>()  
  
{  
  
}
```

When is a constructor called

Everytime an object is created using new() keyword, atleast one constructor is called. It is called a default constructor.

Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

Rules for creating java constructor

There are basically two rules defined for the constructor.

1. Constructor name must be same as its class name
2. Constructor must have no explicit return type

Types of java constructors

There are two types of constructors in java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

DEFAULT CONSTRUCTOR	PARAMETERIZED CONSTRUCTOR
A constructor that is automatically generated by the compiler in the absence of any programmer-defined constructors	A constructor that is created by the programmer with one or more parameters to initialize the instance variables of a class
Has no parameters	Has one or more parameters
If the programmer has not written a constructor, the default constructor is automatically called	Programmer should write his own constructor when writing a parameterized constructor
	Visit www.PEDIAA.com

Java Default Constructor

A constructor is called "Default Constructor" when it doesn't have any parameter. Everytime an object is created using new() keyword, atleast one constructor is called. It is called a default constructor.

SYNTAX:

```
<class_name>()
```

```
{}
```

Example of default constructor .

```
class Bike1
{
    Bike1() //default constructor
    {
        System.out.println("Bike is created");
    }
    public static void main(String args[])
    {
        Bike1 b=new Bike1();
    }
}
```

```
}
```

Rule: If there is no constructor in a class, compiler automatically creates a default constructor.

What is the purpose of default constructor? 2m Default constructor is used to provide the default values to the object like 0, null etc. depending on the type.

Example of default constructor that displays the default values

```
class Student3
{
    int id;
    String name;

    void display()
    {
        System.out.println(id+" "+name);
    }

    public static void main(String args[])
    {
        Student3 s1=new Student3();
        Student3 s2=new Student3();
        s1.display();
        s2.display();
    }
}
```

Output

```
0 null
0 null
```

Java parameterized constructor A constructor which has a specific number of parameters is called parameterized constructor. Parameterized constructor is used to provide different values to the distinct objects.

Example of parameterized constructor

In this example, we have created the constructor of Student class that have two parameters. We can have any number of parameters in the constructor.

```
class Student4
{
```

```

    int id;
    String name;

    Student4(int i,String n)
    {
        id = i;
        name = n;
    }
    void display()
    {
        System.out.println(id+ " "+name);
    }

    public static void main(String args[])
    {
        Student4 s1 = new Student4(111,"Karan");
        Student4 s2 = new Student4(222,"Aryan");
        s1.display();
        s2.display();
    } //end main
} //end class

```

Java Copy Constructor

There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

- By constructor
- By assigning the values of one object into another
- By clone() method of Object class

```
// Write a program to demonstrate Copy Constructor.
```

```
class Rectangle
{
    int length;
    int breadth;

    Rectangle(int l, int b) //constructor to initialize length and breadth of rectangle
    {
        length = l;
        breadth = b;
    }

    Rectangle(Rectangle obj) //copy constructor
    {
        System.out.println("Copy Constructor Invoked");
        length = obj.length;
        breadth = obj.breadth;
    }

    int area() //method to calculate area of rectangle
    {
        return (length * breadth);
    }
}

//class to create Rectangle object and calculate area
class CopyConstructor
{
    public static void main(String[] args)
    {
        Rectangle A = new Rectangle(5,6);
        Rectangle B = new Rectangle(A);
        System.out.println("Area of First Rectangle : "+ A.area());
        System.out.println("Area of First Second Rectangle : "+ B.area());
    }
}
```

Constructor Overloading in Java

Java Constructor overloading is a technique in which a class can have any number of constructors that differ in parameter list. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

Examples of valid constructors for class Account are

```
Account(int a);
Account (int a,int b);
Account (String a,int b);
```

OR

- Just like [method overloading](#), constructors also can be overloaded.
- Same constructor declared with different parameters in the same class is known as **constructor overloading**.
- Compiler differentiates which constructor is to be called depending upon the number of parameters and their sequence of data types.

Example of Constructor Overloading

```
class Student5
{
    int id,age;
    String name;

    Student5(int i,String n)
    {
        id = i;
        name = n;
    }

    Student5(int i,String n,int a)
    {
        id = i;
        name = n;
        age=a;
    }
    void display()
    {
        System.out.println(id+" "+name+" "+age);
    }
}
```



```

public static void main(String args[])
{
    Student5 s1 = new Student5(111,"Karan");
    Student5 s2 = new Student5(222,"Aryan",25);
    s1.display();
    s2.display();
}
} //end class

```

Constructor vs Method		
	Constructor	Method
Name	Constructor's name must be same as the name of the class.	Method's name can be anything.
Return Type	Constructor doesn't have a return type.	Method must have a return type.
Call	Constructor is invoked implicitly by the system.	Method is invoked by the programmer.
Main Job	Constructor can be used to initialize an object.	Method consists of Java code to be executed.
Overload	Constructor can be Overload.	Method also can be overload.

Access specifiers/ modifiers OR visibility control

Private Access Modifier - private:

- Methods, Variables and Constructors that are declared private can only be accessed within the declared class itself.
- Private access modifier is the most restrictive access level. Class and interfaces cannot be private.
- Using the private modifier is the main way that an object encapsulates itself and hide data from the outside world.

Public Access Modifier - public:

- A class, method, constructor, interface etc declared public can be accessed from any other class.
- However if the public class we are trying to access is in a different package, then the public class still need to be imported.

```
public static void main(String[] arguments)
```

```
{
    // ...
}
```

Protected Access Modifier - protected:

Variables, methods and constructors which are declared as protected in a superclass can be accessed only by the subclasses .

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected.

Default:

When no access modifier is specified for a class , method or data member – It is said to be having the **default** access modifier by default.

	Within Same Class	Within same package	Outside the package- (Subclass)	Outside the package- (Global)
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes (only to <u>derived class</u>)	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

STRINGS

String is a sequence of characters. But in java, string is an object. String class is used to create string object.

How to create String object?

There are two ways to create string object:

1. By string literal
2. By new keyword

1) String literal:

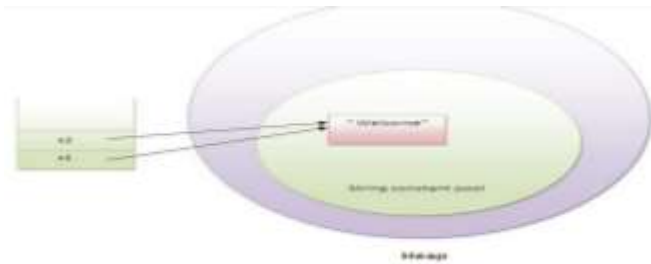
String literal is created by double quote.

ex:

```
String s="hello";
```

Suppose, String s1="Welcome";

`String s2="Welcome";` //no new object will be created



Why Java uses the concept of String literal?

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

1. By new keyword :

Java String is created by using a keyword “new”.

For example: `String s=new String(“Welcome”);`

In such case, **JVM** will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool.

string object will always take more time to execute than string literal because it will construct a new string every time it is executed.

STRING LITERAL	STRING OBJECT
Set of characters that is created by enclosing them inside a pair of double quotes	Set of characters that is created using the new() operator
If the String already exists, the new reference variable will be pointing to the already existing literal	Even the String already exists or not, a new String object will be created
<code>String s = "Hello World";</code>	<code>String s = new String ("Hello World!");</code>
	Visit www.PEDIAA.com

String Handling

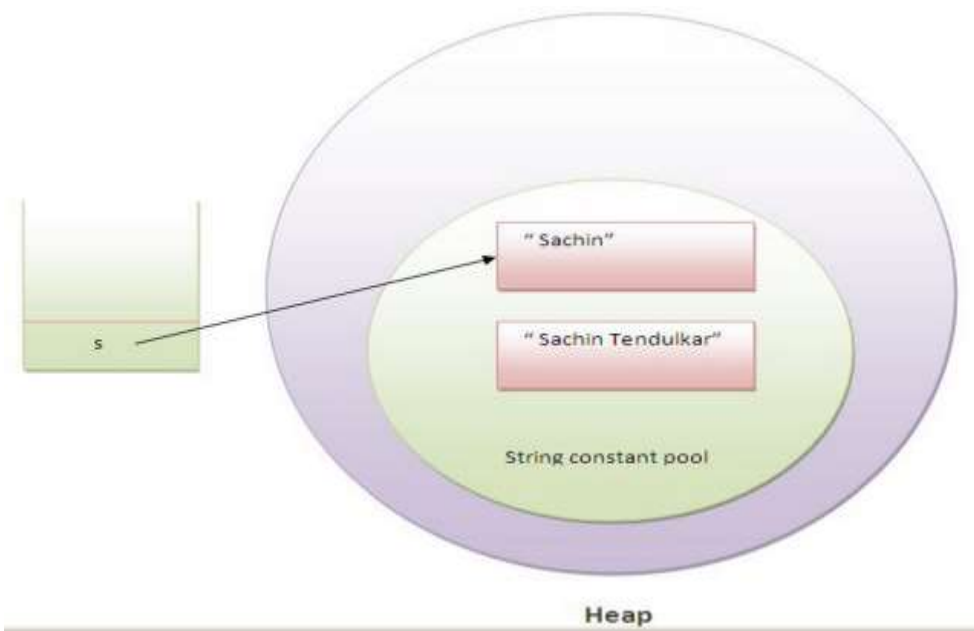
- Concept of String
- Immutable String
- String comparison
- String concatenation
- concept of Substring
- StringBuffer class
- StringBuilder class

Immutable String

In java, strings are immutable (unmodifiable) objects.

```
class Simple
{
    public static void main(String args[])
    {
        String s="sachin";
        s.concat("Tendulkar") ; //concat() method appends the string at the end
        System.out.println(s); // will print sachin bcoz strings are immutable
    }
}
```

output: sachin



Why string objects are immutable in java?

Because java uses the concept of string literal. Suppose there are 5 reference variables, all refers to one object "sachin". If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

We can concat using the following code

```
class concat
{
    public static void main(String args[]){
        String s1="Sachin ";
        String s2="Tendulkar";
        String s3=s1.concat(s2);
        System.out.println(s3);//Sachin Tendulkar
    }
}
```

String comparison

There are three ways to compare strings

1. By **equals()** method
2. By **==** operator
3. By **compareTo()** method

1. equals() method compares the values of string for equality.

class A

```
{
    public static void main(String [] args)
    {
        String s1="sachin";
        String s2="sachin";
        String s3=new String("sachin");
        String s4="saurav";

        System.out.println(s1.equals(s2)); //true
        // System.out.println(s1.equalsIgnoreCase(s2)); //true
        System.out.println(s1.equals(s3)); //true
        System.out.println(s1.equals(s4)); //false
    }
}
```

2. By == operator

The == operator compares references not values.

class A

```
{
    public static void main(String [] args)
    {
        String s1="sachin";
        String s2="sachin";
        String s3=new String("sachin");
        String s4=new String("sachin");

        System.out.println(s1==s2); //true (because both refer to same instance)

        System.out.println(s3==s4); //false (s3 refers to instance created in nonpool)
    }
}
```

3. By compareTo() method:

It compares values and returns an int which tells if the values compare less than, equal or greater than.

ie s1 == s2 : 0

s1 > s2 : positive value

s1 < s2 : negative value

class A

```
{
    public static void main(String [] args)
    {
        String s1="sachin";
        String s2="sachin";
        String s3="ratan"

        System.out.println(s1.compareTo(s2)); // 0
        System.out.println(s1.compareTo(s3)); // 1
        System.out.println(s3.compareTo(s1)); // -1
    }
}
```

String Concatenation:

There are two ways to concat string objects:

1. By + (string concatenation) operator
2. By concat() method

1. By + (string concatenation) operator

It is used to add strings.

class A

```
{
    public static void main(String [] args)
    {
        String s="sachin" + "Tendulkar";

        System.out.println(s); // sachinTendulkar
    }
}
```

2) By concat() method

It concatenates the specified string to the the end of current string.

class A

```
{
    public static void main(String [] args)
    {
        String s1="sachin";
        String s2="Tendulkar";
        String s3=s1.concat(s2);
```

```
System.out.println(s3);  
s3.concat("cricketer"); //will not combine  
System.out.println(s3);  
}  
}
```

Output: sachinTendulkar

Substring

We can extract substring from the given string by using substring method.

1. **substring(int startIndex)** - It returns the substring from the specified startIndex (inclusive).
2. **substring(int startIndex, int endIndex)** - It returns the substring from startIndex to endIndex.

program

```
class A  
{  
    public static void main(String [] args)  
    {  
        String s="Sachin Tendulkar";  
        System.out.println(s.substring(6)); // Tendulkar  
        System.out.println(s.substring(0,6)); //Sachin  
    }  
}
```


Methods of String class

- **length()**
- **charAt(index)**
- **toUpperCase()**
- **toLowerCase()**
- **concat()**
- **equals()**
- **equalsIgnoreCase()**
- **compareTo()**
- **substring()**
- **indexOf()**
- **lastIndexOf()**
- **replace()**

length()

length(): This method is used to get the number of character of any string.

Example

```
class StringHandling
{
    public static void main(String arg[])
    {
        int l;
        String s=new String("Java");
        l=s.length();
        System.out.println("Length: "+l);
    }
}
```

Output

Length: 4

charAt(index)

charAt(): This method is used to get the character at a given index value.

Example

```
class StringHandling
```

```

{
public static void main(String arg[])
{
char c;
String s=new String("Java");
c=s.charAt(2);
System.out.println("Character: "+c);
}
}

```

Character: v

toUpperCase()

toUpperCase(): This method is use to convert lower case string into upper case.

Example

```

class StringHandling
{
public static void main(String arg[])
{
String s="Java";
System.out.println("String: "+s.toUpperCase());
}
}

```

Output

String: JAVA

toLowerCase()

toLowerCase(): This method is used to convert lower case string into upper case.

Example

```

class StringHandling
{
public static void main(String arg[])

```

```

{
String s="JAVA";
System.out.println("String: "+s.toLowerCase());
}
}

```

Output

String: java

concat()

concat(): This method is used to combined two string.

Example

```

class StringHandling
{
public static void main(String arg[])
{
String s1="Presidency";
String s2="College";
System.out.println("Combined String: "+s1.concat(s2));
}
}

```

Output

Combined String: PresidencyCollege

equals()

equals(): This method is used to compare two strings, It return true if strings are same otherwise return false. It is case sensitive method.

Example

```

class StringHandling
{
public static void main(String arg[])

```

```

{
String s1="Presidency";
String s2="College";
String s3="Presidency";
System.out.println("Compare String: "+s1.equals(s2));
System.out.println("Compare String: "+s1.equals(s3));
}
}

```

Compare String: false

Compare String: true

equalsIgnoreCase()

equalsIgnoreCase(): This method is case insensitive method, It return true if the contents of both strings are same otherwise false.

Example

```

class StringHandling
{
public static void main(String arg[])
{
String s1="Presidency";
String s2="PRESIDENCY";
String s3="College";
System.out.println("Compare String: "+s1.equalsIgnoreCase(s2));
System.out.println("Compare String: "+s1.equalsIgnoreCase(s3));
}
}

```

Output

Compare String: true

Compare String: false

compareTo()

compareTo(): This method is used to compare two strings by taking unicode values, It return 0 if the string are same otherwise return +ve or -ve integer values.

Example

```
class StringHandling
{
    public static void main(String arg[])
    {
        String s1="Hitesh";
        String s2="Raddy";
        int i;
        i=s1.compareTo(s2);
        if(i==0)
        {
            System.out.println("Strings are same");
        }
        else
        {
            System.out.println("Strings are not same");
        }
    }
}
```

Output

Strings are not same

substring()

substring(): This method is used to get the part of given string.

Example

```
class StringHandling
{
```

```
public static void main(String arg[])
{
    String s="Java is programming language";
    System.out.println(s.substring(8)); // 8 is starting index
}
```

Output

```
programming language
```

indexOf()

indexOf(): This method is used find the index value of given string. It always gives starting index value of first occurrence of string.

Example

```
class StringHandling
{
    public static void main(String arg[])
    {
        String s="Java is programming language";
        System.out.println(s.indexOf("programming"));
    }
}
```

Output

```
8
```

lastIndexOf()

lastIndexOf(): This method used to return the starting index value of last occurrence of the given string.

Example

```

class StringHandling
{
    public static void main(String arg[])
    {
        String s1="Java is programming language";
        String s2="Java is good programming language";
        System.out.println(s1.lastIndexOf("programming"));
        System.out.println(s2.lastIndexOf("programming"));
    }
}

```

Output

```

8
13

```

replace()

replace(): This method is used to return a duplicate string by replacing old character with new character.

Note: In this method data of original string will never be modify.

Example

```

class StringHandling
{
    public static void main(String arg[])
    {
        String s1="java";
        String s2=s1.replace('j', 'k');
        System.out.println(s2);
    }
}

```

Output Kava

No.	String	StringBuffer
1)	The String class is immutable.	The StringBuffer class is mutable.
2)	String is slow and consumes more memory when we concatenate too many strings because every time it creates new instance.	StringBuffer is fast and consumes less memory when we concatenate t strings.
3)	String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method.	StringBuffer class doesn't override the equals() method of Object class.
4)	String class is slower while performing concatenation operation.	StringBuffer class is faster while performing concatenation operation.
5)	String class uses String constant pool.	StringBuffer uses Heap memory

StringBuffer class:

It is used to create mutable (modifiable) string.

Commonly used methods of StringBuffer class

1. **append(String s)** - is used to append the specified string with the string.
2. **insert(int offset, String s)** - is used to insert the specified string with this string at the specified position.
3. **replace(int startIndex, int endIndex,String str)** - is used to replace the string from specified index.
4. **delete(int startIndex, int endIndex)** - delete the string from specified index.

class B

```

{
    public static void main(String [] args)
    {
        StringBuffer sb = new StringBuffer("Hello");
        sb.append("java");

        System.out.println(sb);
    }
}

```

output Hellojava

StringBuilder class

It is used to create mutable string.

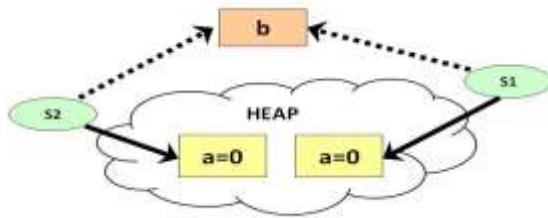
It is same as StringBuffer class except that it is non-synchronized.

It is available since JDK1.5.

Note: StringBuffer class is thread-safe. ie multiple threads cannot access it simultaneously. So it is safe and will result in an order. Whereas StringBuilder class is not.

STATIC MEMBERS

- Static variables are used in a situation where single variable's value need to be shared by all objects.



In Java, static members are those which belongs to the class and you can access these members without instantiating the class.

The static keyword can be used with methods, fields, classes (inner/nested), blocks.

Static Methods – You can create a static method by using the keyword *static*. Static methods can access only static fields, methods. To access static methods there is no need to instantiate the class, you can do it just using the class name as –

```
class A
{
    static int a;
    public static void sample()
    {
        a++;
        System.out.println("value of a is +a");
    }
    public static void main(String args[]){
        A.sample();
        A.sample();
    }
}
```

```
}  
}
```