

MCA FIRST SEMESTER- 2021

MCA20103-OBJECT ORIENTED PROGRAMMING USING JAVA

UNIT – I

INTRODUCTION TO JAVA: Features of Java, The Java Virtual Machine, Parts of Java.

FIRST STEP TOWARDS JAVA PROGRAMMING: API Document, Starting a Java Program, Formatting the Output.

NAMING CONVENTIONS AND DATA TYPES: Naming Conventions in Java, Data Types in Java, Literals.

OPERATORS IN JAVA: Operators, Priority of Operators.

CONTROL STATEMENTS IN JAVA: if...else Statement, do...while Loop, while Loop, for Loop, switch Statement, break Statement, continue Statement, return Statement.

INPUT AND OUTPUT STREAMS IN JAVA: Accepting Input from the Keyboard, Reading Input with java.util.Scanner Class, Displaying Output with System.out.println(), Displaying Formatted Output with String.format()

(Chapters: 2, 3, 4, 5, 6, 7 of the Text Book)

Introduction To Java

1.Introduction to JAVA:-

1.1.History of Java: The James gosling, Mike Sheridan and Patrick naughton were working as professors at sun Micro systems in 1991. The sun micro system was university. The Applications for consumer electronic devices such as washing machines, microwave ovens was developed by c, c++ devices. In 1991, James gosling team got a new requirement. That is developing application for networked consumer electronic devices.

Networked consumer electronic devices:- The two electronic devices are communicated via either wireless network or wired network. The compiled data is transferred between devices across network. Therefore while travelling **security is required for data and data must be executed on another device irrespective of another device hardware.**

Example: Tv and remote.

Air conditioner and remote.

Both are two devices.

At that time, there were only c,c++ ,simula languages. But these languages were not suitable for developing such applications Because platform dependent languages.

James gosling need to develop the new programming language. After lot of work and 4 years, He invented new language. It did not have name. James gosling wanted to set a name to new language. While thinking, he came to window at office and opened the window shutter. He saw tree. It was OAK tree. It was big and strong tree and national tree in that tree. He selected tree name as new language name.

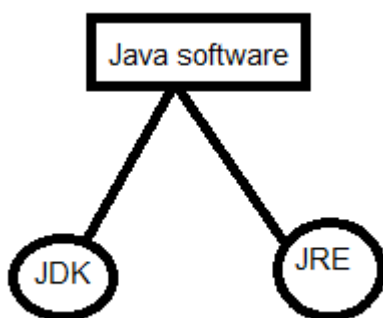
He was unable to register this PL with name OAK because with same name had been used by another company called OAK Technologies. Meanwhile, www was emerging into a market. In www so many verities of CPUs under different s/w and H/W environments(window,solaris...etc) will be connected. So james gosling knows that we require one language to develop platform independent applications/Internet based Business applications.

He and his team started enhancement of OAK. He name to enhancement project 'GREEN PROJECT'. This is dummy name. Later it is renamed to JAVA. This name was proposed by one of team member. There is no abbreviation for java. The name java specifically does not have any meaning rather it refers to **hot, aromatic drink coffee**. The name is just derived from coffee cup. That is reason Java programming language icon is coffee cup.

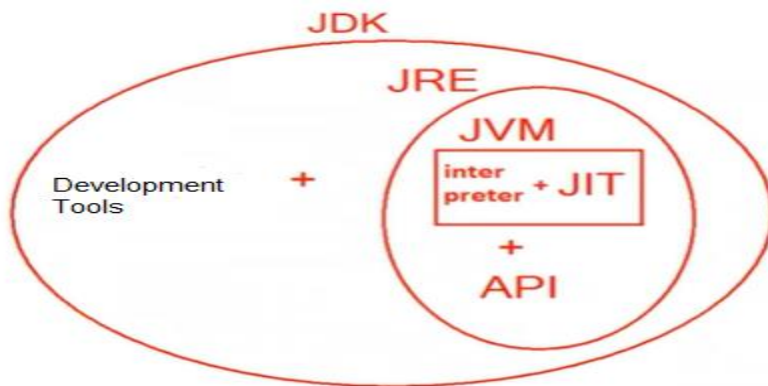
1.2.Types of Java Software:-The java software is divided into two types.

1.2.1)JDK.

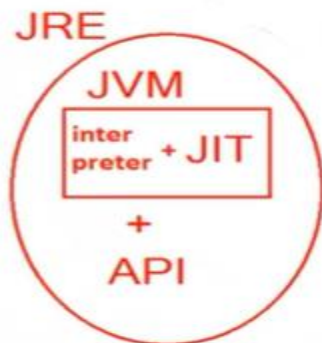
1.2.2)JRE



1.2.1.JDK:- when we install JDK,JRE is also installed automatically. JDK has both Development tools(compiler, documentation generation program, tools, jar creation program, debugger...etc) and JRE. Developers develop the new programs and Developers compile and execute the new programs in development Environment . Therefore JDK has to be installed in development environment.



1.2.2.JRE:- JRE software is available as separate pack, so we can install JRE alone. JRE has only JVM. Hence using JRE we can only execute already developed applications. JRE has only to be installed in Testing environment. Because testers just execute the application for testing. The JRE contains JVM and API(Library)
Note:- Sun micro system(oracle) developed **individual JVM** for every operating system.



The Interpreter, JIT and associated files jointly is called JVM.

1.3.Java Versions: The version is number. It is used to identify the available features in the software. Sun Micro system first released Java on Jan 23, 1996. Java has two types of versions.

1. Major Version:- It is main version of software contains new features.
2. Minor version:- It is sub version of main version contains bug fixes.

Example:

Main version: JDK 9.0

Minor Versions: JDK 9.0.1, JDK 9.0.2...etc.

| Java Language Name | Java Softwares Names |
|---------------------|----------------------|
| java 1.0 | jdk1.0, jre1.0 |
| java 1.1 | jdk1.1, jre 1.1 |
| Java 2 platform 1.2 | j2sdk1.2, j2sre1.2 |
| java2 platfrom 1.3 | j2sdk1.3, j2sre1.3 |
| java2 platform 1.4 | j2sdk1.4, j2sre1.4 |
| java 5 | jdk 1.5, jre1.5 |
| java 6 | jdk1.6, jre 1.6 |
| java7 | jdk1.7, jre 1.7 |
| java 8 | jdk 1.8, jre 1.8 |
| java 9 | jdk9 , jre 9 |
| --- | |
| -- | |
| java 18 | JDK18, JRE18 |

LTS(Long Term support) versions:- when new version is released into market, the sun micro system(oracle) announces that new version is LTS version or non LTS version. If it is LTS version, the new version can be downloaded and used. Other wise it is non LTS version.

Example: java8.0, java11.0, java 17.0

When we try to download non LTS version into local machine, browser automatically redirect us to LTS version.

2.Features of JAVA:-

2.1) Simple:-

| English Language | Programming Language |
|------------------------------------|---|
| 1. Alphabet/Letters[26] | 1. Programming Elements[8] |
| 2. words. No. of words is infinity | 2. Reserve words. Java has around 70 reserve words. |
| 3. Grammar [8 parts of speech] | 3. syntax . Java has only 20 simple syntaxes. |
| 4. sentences[infinity] | 4. statements[infinity] |
| 5. paragraph[infinity] | 5. block [infinity] |
| 6. page [infinity] | 6. program[infinity] |
| 7. Book[infinity] | 7. Project [infinity] |

The programming language(java) is very easy than English language. we can know this, when we see the above table.

Java is easy to learn and its syntax is quite simple and easy to understand. The confusing and difficult concepts (pointers,operator overloading)of c++ are left out java.

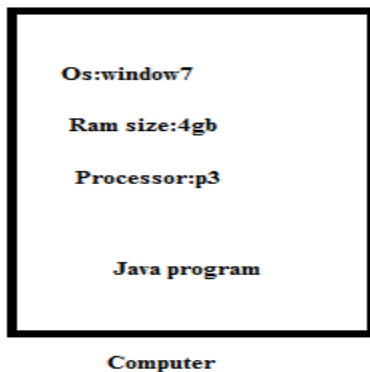
2.2) Object Oriented:- The java strongly supports OOPs concepts due to which it is called object-oriented language. The concepts are

- class
- object
- Encapsulation
- Inheritance
- Abstraction.
- polymorphism.

The java is not fully object-oriented programming language Because of following reasons.

1. java has predefined primitive data types(which are not objects).
2. You can access the static member of class with out creating the object.

2.3) Architectural –Neutral:-



Windows7 was installed in computer. Its current ram size is 2gb. Its processor is p3. Due to some reasons, I updated current system architecture. I replaced the windows7 with windows10. I increased the ram size. I removed existing processor. I placed latest processor.

Eventhough I made changes in existing computer architecture, The java program can run successfully with out any modification. This feature is architectural neutral.

2.4)Java is a platform independent language

Compiler (javac) converts source code (.java file) to the byte code (.class file). As mentioned above, JVM executes the byte code produced by compiler. This byte code can run on any platform such as Windows, Linux, and Mac OS etc. This means a program that is compiled on windows can run on Linux and vice-versa.

Each operating system has different JVM; however the output they produce after execution of byte code is same across all operating systems. That is why we call java as platform independent language.

2.5)Multithreading

Java supports [multithreading](#). Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.

3.Java Virtual Machine (JVM) & its Architecture:-

Java Virtual Machine (JVM) is an engine that provides runtime environment to drive the Java Code or applications. It converts Java byte code into machines language. JVM is a part of Java Run Environment (JRE). In other programming languages, the compiler produces machine code for a particular system. However, Java compiler produces code for a Virtual Machine known as Java Virtual Machine. Java applications are called WORA (Write Once Run Anywhere).

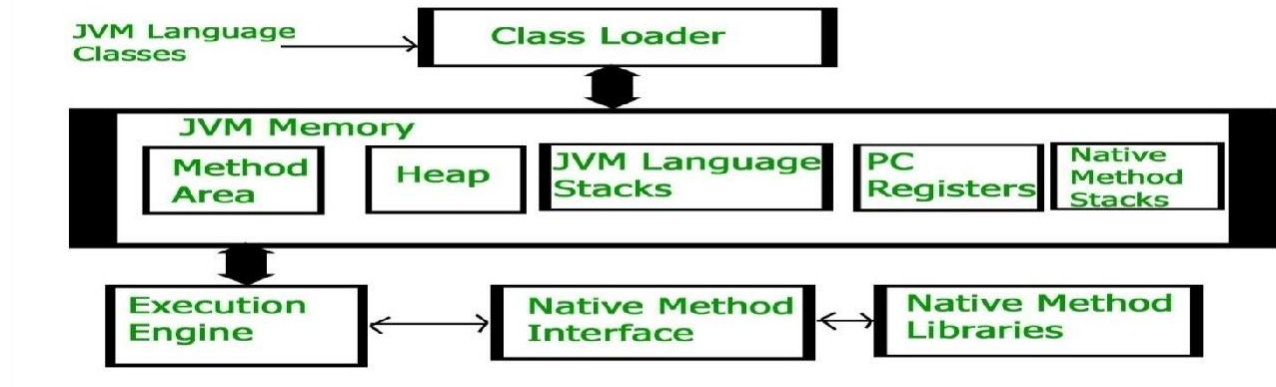
Here is how JVM works

First, Java code is compiled into byte code. This byte code gets interpreted on different machines Between host system and Java source, Byte code is an intermediary language.

JVM is responsible for allocating memory space.



JVM Architecture: - JVM contains class loader, memory area, execution engine etc. the following diagram illustrates the JVM Architecture.



1) Class Loader

The class loader is a subsystem used for loading class files. It performs three major functions viz. Loading, Linking, and Initialization.

2) Method Area

JVM Method Area stores class structures like metadata, the constant runtime pool, and the code for methods.

3) Heap

All the Objects, their related instance variables, and arrays are stored in the heap. This memory is common and shared across multiple threads.

4) JVM language Stacks

Java language Stacks store local variables, and it's partial results. Each thread has its own JVM stack, created simultaneously as the thread is created. A new frame is created whenever a method is invoked, and it is deleted when method invocation process is complete.

5) PC Registers

PC register store the address of the Java virtual machine instruction which is currently executing. In Java, each thread has its separate PC register.

6) Native Method Stacks

Native method stacks hold the instruction of native code depends on the native library. It is written in another language instead of Java.

7) Execution Engine

It is a type of software used to test hardware, software, or complete systems. The test execution engine never carries any information about the tested product.

8) Native Method interface

The Native Method Interface is a programming framework. It allows Java code which is running in a JVM to call by libraries and native applications.

9) Native Method Libraries

Native Libraries is a collection of the Native Libraries(C, C++) which are needed by the Execution Engine.

4.Basic Structure of Java Program(or) parts of Java:-

JAVA Program Structure

4.1.Documentation Section: Documentation Section:- This section contains set of comment lines giving the

name of program, name of the author and other details.

4.1.1)comment :- The comment is description of program elements (module, package, interface, class, variable, blocks, constructor, method, inner class..etc).

Java supports 3 types of comments.

1. Single line comment(Inline comments)
2. Multiline comment(Block comments)
3. Document comment.

4.1.1.1) Single line comment: // symbols are used to create single line comment. The description is only in single line.

Syntax: // statement

Example: // Salary variable stores the employ salary

Float salary;

4.1.1.2) Multiline comment : /* */ symbols are used to create multiline comment.

Syntax: /* statements */

Example:

```
/* Author-Name: Sukumar
   Date:04-05-2020
   Program-Name: Addition.
   Company-name: Raos'Degree college.
  */
```

4.1.1.3) Document Comment:- The java provides document comment symbols which is /** and */. The document comment starts with /** and ends with */. The documentation comment may precede the class header, interface header, method header, field declaration and other component. To make source code Documentation without distributing the source code, we need some tool. That tool is javadoc. The tool does opposite of what compiler does. It collects, formats and organize the java documents in source code.

4.2. Package Section/Package Statement:-

Syntax:

Package packagename;

This is an optional statement. This section contains only one package statement. If u put more than one package statement in this section, compiler displays error message.

Example:

```
package p1;
package p2;
class A
{
    public static void main(String arg[])throws Exception
    {
        char a='N';
        System.out.println("character is:"+a);
    }
}
```

Output:

D:\>javac A.java

A.java:2: class, interface, or enum expected

package p2;

4.3.Import Section/Import Statement: This section is optional section. It contains one or more than one import statements. if we want to use a class[es] of another package, then you can do this by importing it directly into our program.

Syntax:1

Import packagename.classname;

Syntax:2

Import packagename.*;

4.4.Interface Section/interface Statements: This section is optional section. It contains one or more than one interface definitions.

4.5.Class Section:-This section is also optional section. It can also contain more than one class definitions where as class names must be unique.

Basic class Syntax

```
class ClassName{  
    variable declaration section  
    block definition section  
    constructors  
    methods  
    public static void main(String arg[])  
    {  
  
    }  
}
```

4.6. Main() method:- The main() is the starting point for JVM to start execution of a Java program.

public: It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

static: You can make a method as static by using the keyword static. JVM should call the main() method without creating an object. Static methods are the method which invokes without creating the objects, so JVM do not need any object to call the main() method.

void: In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

Main: It takes only one argument. That argument data type is string array.

FIRST STEP TOWARDS JAVA PROGRAMMING**1.API DOCUMENT:**

- The API document generated from the java program is similar to a help file where all the features are available with their descriptions.
 - The user can refer to any feature in this file and get some knowledge recording how we can use it in our program
 - To create an API document, we should use a special compiler called javadoc compiler
 - An API document is a .html file that contains description of all the features of a software, a product or a technology
 - The input of the javadoc command is .java (source file) file and the output of javadoc command is .html (API document) file.
 - Javadoc creates description for only public classes, public methods and public variables.
- Syntax:**Javadoc programname.java

```
/**Description about package  
packagename;  
/**description about class*/  
Class Classname  
{  
Class code;  
/*description about method*/  
Void methodname()  
{  
Method code  
}  
\\**description about method*/  
Void methodname()  
{  
Method code  
}  
}
```

```
Package packagename;  
Description about package  
  
class Classname  
Description about class  
  
Void methodname( )  
Description about method  
  
void methodname( )  
Description about method
```

EXAMPLE:

```
/**SAMPLE PROGRAM*/
import java.lang.System;
import java.lang.String;
/** This is class A*/
public class A
{
    public int x,y;
    /** read x and y*/
    public void get Data()
    {
        x=10;
        y=20;
    }
    /**displays x and y*/
    public void dispData()
    {
        System.out.println(x);
        System.out.println(y);
    }
}
/** this is class sample*/
public class sample
{
    public static void main(String args[])
    {
        A obj=new A();
        obj.dispData();
        obj.getData();
        obj.dispData();
    }
}
```

2.STARTING A JAVA PROGRAM:

/* Author: A.sukumar

Date: 08-01-2023

Functionality: This program display “hello world” message on Console.

***/**

Import java.io.*;

Import java.util.*;

Class Sample{

public static void main(String args[])

{

System.out.println(“HelloWorld”);

}

}

Task1: Java programs must be saved in a file whose name ends with the .java extension.

Filename: Sample.java

Task2: Compile the java program.

Syntax: javac filename.java

Example: > javac Sample.java

Task3: Run the Java program.

Syntax: java filename/classname

Example: java Sample

Output:

| |
|-------------|
| Hello World |
|-------------|

3.Formatting the output:

Java also supports backslash codes to format the output. These are also called escape sequence characters. Every escape sequence character starts with (\) following by some characters. The Java supports following escape sequence characters.

| Backslash code | Meaning |
|----------------|----------------------|
| \n | Next line/new line |
| \t | Horizontal tab space |
| \b | Backspace |
| \f | Form feed |
| \\ | Displays \ |
| \" | Displays “ |
| \’ | Displays ‘ |

Example:1 Develop the program to display ‘suku’ in the following pattern by using single SOPln.

S

NAMING CONVENTIONS AND DATA TYPES**1.NAMING CINVENTIONS IN JAVA:**

The naming convention is not rule. It is just **agreement**/guidelines among java developers for easy readability of code.

1.1.Class:-

- a. The className should be common noun.
- b. whe you create class name with several words. You should capitalize each word.the uppertime letter acts as a separator.

Example:

Employee,TwoStairBuilding,Customer,ElectricCooker.

1.2. Interface:-

- a. The Interface should be Adjective.
- b. When you create Interface name with several words, You should capitalize each word.the uppertime letter acts as separator.

Example:- Runnable, Printable,Accessible ...etc.

1.3.Method:-

- a. The Method name should be verb.
- b. If method name contains multiple words, the first word is in small letter then from second word onwards each new word starts with capital letter.
- c. If method name contains single word, that word is small letter.

Example: print , display, readLine

1.4. Variable:-

- a. If variable name contains multiple words, the first word is in small letter then from second word onwards each new word starts with capital letter.
- b. If variable name contains single word, that word is small letter.
- c. Avoid using one-character variables such as x,y,z ..etc.

Example:

empName,StudentRollNum,CarNo

1.5. Package:-

- a. name of package in java is written in lowercase.
- b. Ifname contains multiple words, It should be separated by(.

Example:-

Java.util

Java.io ...etc

1.6. Constants:-

- a. It should be in uppercase letters.
- b. If name contains multiple words, It shouldbe separated by underscore(_).

Example:

RED,YELLOW, MIN_AGE,MAX_AGE.

Camelcase:- If name is combined with multiple words, except firstword, remaining all words starts with

uppercase letter. This is camelcase.

Java follows camel-case syntax for naming classes, Interface, method and variable.

□

2.Data Types in Java:

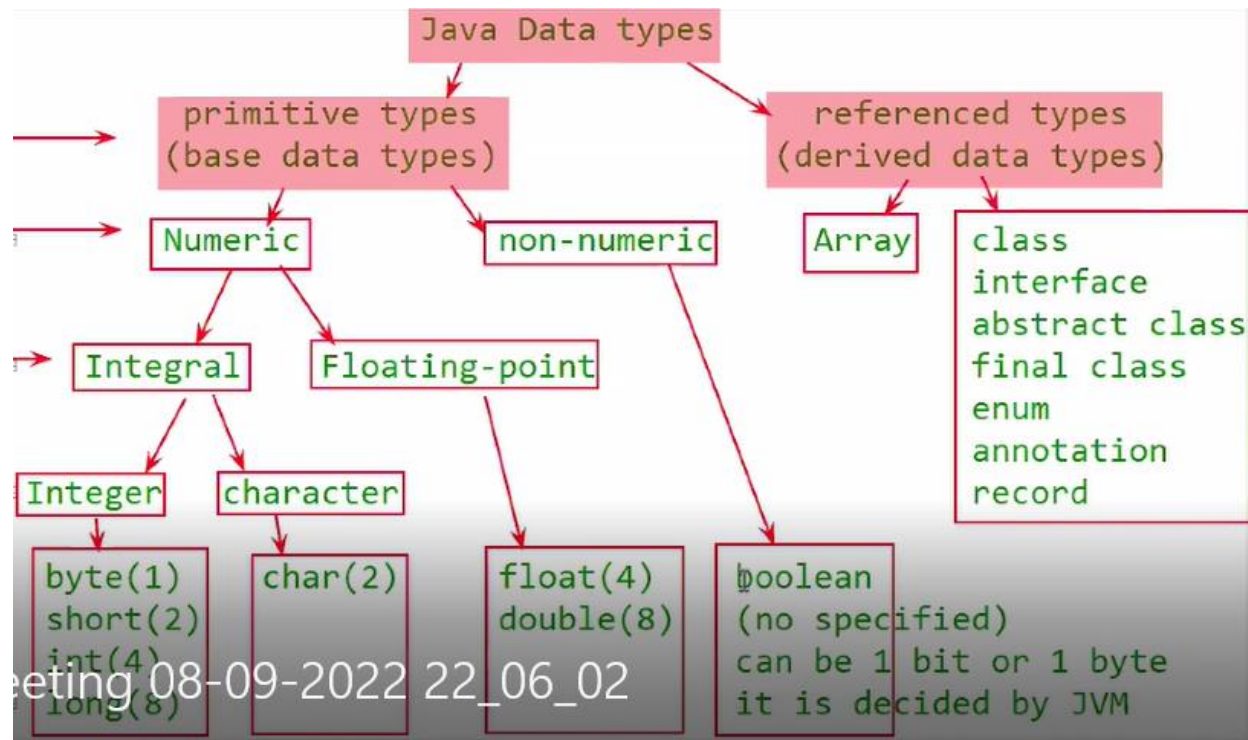
The keyword or class name which is used for creating variable or object is called data type. The data type keyword will inform compiler and JVM the following information :

- Size of memory to be created for variable .(1/2/4/8 bytes).
- Type of value that can be stored in memory .
- Range of value that can be stored in memory .

2. Java Data Types: Java supports two types of Data types.

2.1. Primitive Types(Base Data types) .

2.2. Referenced Types(Derived Data types)



2.1. Primitive Data Types:- We use primitive data types(PDT) for storing a single value. As per mathematical values, the primitive data types were divided into Numeric and non-numeric.

2.2. Referenced Data Types(RDT):-

- The array must be used for storing multiple values of same type.
- The class must be used for storing multiple values of different type.
- The enum must be used for storing multiple constant values with different names.

| Data Type | Default Value | Default size |
|-----------|---------------|--------------|
| boolean | false | 1 bit |
| char | '\u0000' | 2 byte |
| byte | 0 | 1 byte |
| short | 0 | 2 bytes |
| int | 0 | 4 bytes |
| long | 0L | 8 bytes |
| float | 0.0f | 4 bytes |
| double | 0.0d | 8 bytes |

1. Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions. The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Example: Boolean one = false

2. Byte Data Type

Its minimum value is -128 and maximum value is 127. Its default value is 0. The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.

Example: byte a = 10, byte b = -20

3. Short Data Type

Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0. The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example: short s = 10000, short r = -5000

4. Int Data Type

Its value-range lies between - 2,147,483,648 to 2,147,483,647 . Its minimum value is - 2,147,483,648 and maximum value is 2,147,483,647. Its default value is 0.

Example: int a = 100000, int b = -200000

5. Long Data Type

Its value-range lies between -9,223,372,036,854,775,808(-2^{63}) to 9,223,372,036,854,775,807($2^{63} - 1$)(inclusive). Its minimum value is -9,223,372,036,854,775,808 and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Example: long a = 100000L, long b = -200000L

6. Float Data Type

The float data type is a single-precision 32-bit floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

Example: float f1 = 234.5f

7. Double Data Type

The double data type is a double-precision 64-bit floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

Example: double d1 = 12.3

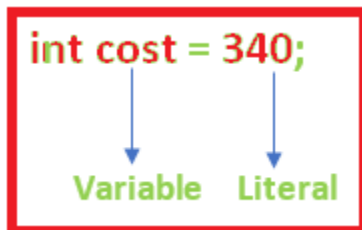
8. Char Data Type

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive). The char data type is used to store characters.

Example: char letter A = 'A'

4.Literal:-

The literal is single character or sequence of characters(letters,digits & other characters) that represent value to be stored in variable name. The following figure represents a literal.



There are 6 major types of literals.

- 1.Integer literal
- 2.Floating_point literal.
- 3.Character literals.
- 4.String literals.
- 5.boolean literals.
- 6.Backslash character literals.

OPERATORS IN JAVA**1.Operator:-**

Definition: The symbol or word that performs

- a. Assignment operation
- b. Validation and
- c. Calculation operations

and returns a result. It is called operator.

- Validation means checking/knowing the given value is correct or wrong. The operators that perform validations will return boolean type result true/false. So these operators are also called as boolean operators.
- The operators that perform calculations will return number type result.
- The operator that stores result in destination variable will return the same copying values as result. This operator is also called as assignment operator.

1.1.Types of Operators:- Based on number of operands accepted by an operator, they are divided into three types.

- a. Unary operators.(11)
- b. Binary Operators.(27)
- c. Ternary Operators.(2)

- a) **Unary Operator:-** An operator that can take only one operand is called unary operator. We can place unary operator either before or after given operand.

Ex:-

+a, -b, !true, !false, ++a, a++, b--, --b, new Student();

- b) **Binary Operators:-** An operator that takes two operands is called binary operator. We can place binary operator only in between two operands.

Example:

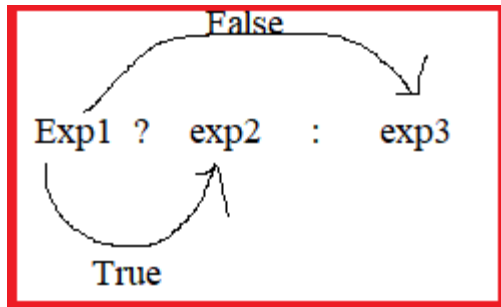
A+b, a-b, a*b, a&b, a<b, a!=b, a==b, ...etc.

- c) An operator that takes 3 operands is called ternary operator.

Example:

(?:) is ternary operator. It has two operators ? and :. We should place ternary operator also in between 3 operands.

Example:



The first expression should generate boolean result.

- ➔ If result is true, exp2 is only executed.
- ➔ If result is false, exp3 is only executed.

1.2. Unary Operators:-(+, -, ! ~, ++, --):

a.unary +:- It is used to represent the positive operand. In this use the operator is optional.

Syntax:- + operand

b.unary - :- It is used to convert positive value into negative and negative value into +value.

Syntax:- -operand

c.logical complement operator:- It is used to reverse the value of Boolean value.

Syntax:- ! operand

Example:

```

import java.io.*;
class sample
{
    public static void main(String[] args) {
        System.out.println(+10);
        System.out.println(-10);
        System.out.println(!true);
        System.out.println(~200000);
        System.out.println(~10);
    }
}
  
```

```

C:\Windows\system32\cmd.exe
10
-10
false
-200001
-11
Press any key to continue . . . _

```

1.3.Binary Operators:-

1. Arithmetic operator:

Arithmetic operators are used to perform arithmetic calculations. The following are the Arithmetic operators

| operator | Meaning |
|----------|----------------|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |

2. Relational operators:

Relational operators are used to compare two values and to give either true or false. These are used to form simple conditions. All the relational operators are applied on any type of data. The following are the relational operators.

| Operator | Meaning |
|----------|-----------------------------|
| < | Less than |
| <= | Is less than or equal to |
| > | Is greater than |
| >= | Is greater than or equal to |
| == | Is equal to |
| != | Is not equal to |

3. Logical operators:

These operators are used to combine two or more conditions and give the result either true or false. These conditions are called compound conditions.

| Operator | Meaning |
|----------|-------------|
| && | Logical AND |
| | Logical OR |
| ! | Logical NOT |

➤ Logical AND (&&) Operator:

This operator gives true if all the conditions are true otherwise it gives false. The truth table for && operator is as follows

| Condition 1 | Condition 2 | Condition 1 &&Condition 2 |
|-------------|-------------|---------------------------|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

➤ **Logical OR (||) Operator:**

This operator gives false if all conditions are false otherwise it gives true. The truth table for || operator is as follows:

| Condition 1 | Condition 2 | Condition 1 condition 2 |
|-------------|-------------|---------------------------|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

➤ **Logical not (!) operator:**

This operator negates (opposite) the result of a condition. It means if the condition is true then it gives false. If the condition is false then it returns true.

The truth table for (!) Operator is as follows :

| conditon | !(condition) |
|----------|--------------|
| True | False |
| False | True |

Eg: Let a=15, b=20 then

| Expression | Result |
|--------------|--------|
| a==15 b==20 | True |
| a>15&&b>=20 | False |
| a<=15&&b<=20 | True |
| !(a==15) | False |

4. Assignment Operators:-The operator that will return the same copying values as result and store result in destination variable. This operator is called as assignment operator.

Syntax:

dest-var = literal/expression/non-void method call /var-name

a)If right side part of assignement operator is literal/const expression/non-void method call then compiler directly uses the values generates the final result and verifies that whether result is in dest-var range or not and result type is compitable or not.

If yes, store the result in dest-var.

If no, display CE : possible loss of precession.

5. Increment and Decrement operator:

The ++ is increment operator.

The -- is decrement operator.

- a) Pre-Increment and Pre-decrement operator:-If increment operator is before the variable then it is said to be pre-increment operator.

Example:- ++a , ++c , ++b.

If decrement operator is after the variable then it is said to be pre-decrement operator.

Example:-

--b , --d, --x.

Note:- First increment /decrement value and then read value ,use this in expression.(expression is executed with new value).

- b) Post-increment and post-decrement operator:- if increment operator is after the variable then it is said to be post-increment operator.

Example:- a++ , c++ , b++

If decrement operator is after the variable then it is said to be post-decrement operator.

Example:-

b-- , d --- x—

Note:- First read value , substitute in expression and then either increase or decrease.(expression is executed with old value)

Example:-1

```
Int x=10;
```

```
Int y=20;
```

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

```
++x;
```

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

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

```
Y++;
```

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

Output:

10

11

20

21

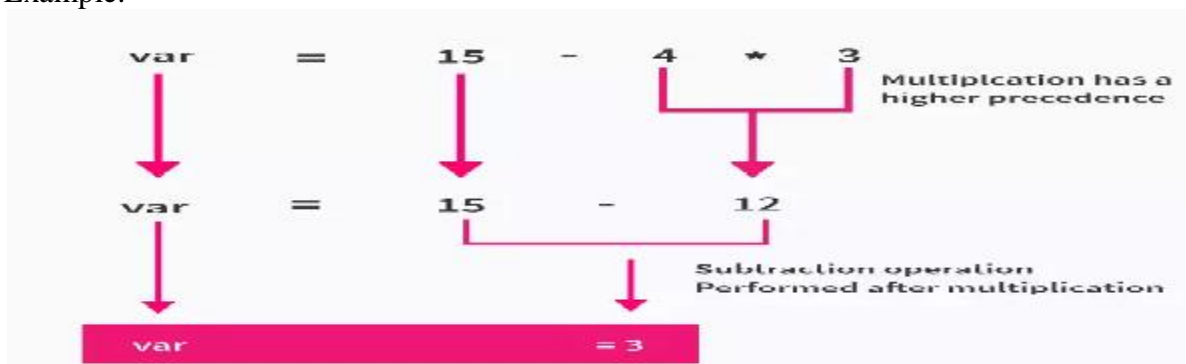
2. Priority of Operators:

The order of operators execution in expression is called operator precedence.

If expression consists of several operators then which operator to be executed first and which operator to be executed next this order is decided by operator precedence.

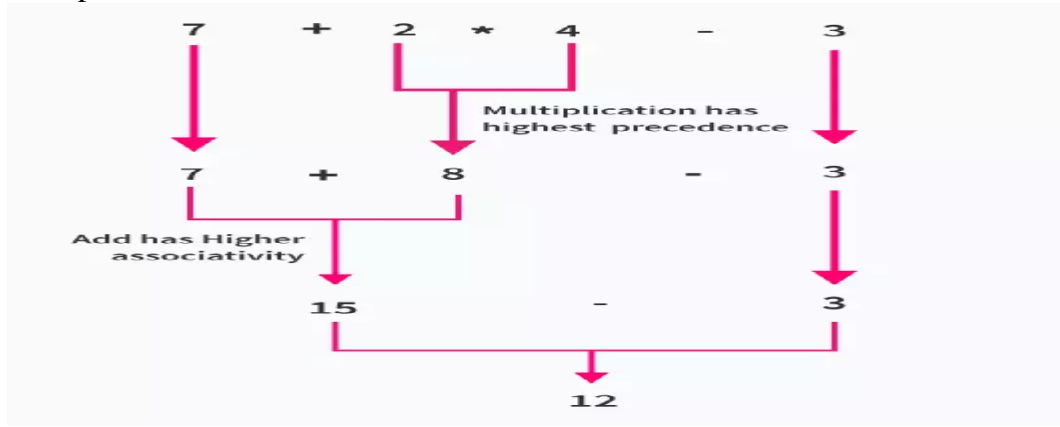
| Operators | Precedence |
|------------------------|--|
| postfix, prefix, unary | <code>expr++ expr-- ++expr --expr +expr -expr ~ ! new</code> |
| multiplicative | <code>* / %</code> |
| additive | <code>+ -</code> |
| shift | <code><< >> >>></code> |
| relational | <code>< > <= >= instanceof</code> |
| equality | <code>== !=</code> |
| bitwise AND | <code>&</code> |
| bitwise exclusive OR | <code>^</code> |
| bitwise inclusive OR | <code> </code> |
| logical AND | <code>&&</code> |
| logical OR | <code> </code> |
| ternary | <code>? :</code> |
| Lambda Expression | <code>-></code> |
| assignment | <code>= += -= *= /= %= &= ^= = <<= >>= >>>=</code> |

Example:-



When two operators have same precedence in expression, operator associativity comes into play. The associativity of operators is used to determine the direction in which same precedence operators are executed.

Example:-



CONTROL STATEMENTS IN JAVA

1.Control Statements in Java:

A control statement works as a determiner for deciding the next task of the other statements whether to execute or not. An 'If' statement decides whether to execute a statement or which statement has to execute first between the two. In Java, the control statements are divided into three categories which are **selection statements**, **iteration statements**, and **jump statements**. A program can execute from top to bottom but if we use a control statement. We can set order for executing a program based on values and logic.

- [Decision Making in Java](#)
 - [Simple if Statement](#)
 - [if...else Statement](#)
 - [Nested if statement](#)
 - [if...else if...else statement](#)
 - [Switch statement](#)
- [Looping Statements in Java](#)
 - [While](#)
 - [Do...while](#)
 - [For](#)
- [Branching Statements in Java](#)
 - [Break](#)
 - [Continue](#)

1. Decision Making Statements in Java

Decision making statements are statements which decides what to execute and when. They are similar to decision making in real time. Control flow statements control the flow of a program's execution. Here flow of execution will be based on state of a program. We have 4 decision making statements available in Java.

- ***Simple if Statement***

Simple if statement is the basic of decision-making statements in Java. It decides if certain amount of code should be executed based on the condition.

Syntax:

```
if (condition) {  
Statement 1; //if condition becomes true then this will be executed  
}  
Statement 2; //this will be executed irrespective of condition becomes true or false
```

Example:

```
class ifTest  
{  
    public static void main(String args[])  
    {  
        int x = 5;  
        if (x > 10)  
            System.out.println("Inside If");  
        System.out.println("After if statement");  
    }  
}
```

Output:

After if statement

- ***if...else Statement***

In if...else statement, if condition is true then statements in if block will be executed but if it comes out as false then else block will be executed.

Syntax:

```
if(condition) {  
    Statement 1; //if condition becomes true then this will be executed  
}  
else{  
    Statement 2;  
}
```

Example:

```
class ifelseTest  
{  
    public static void main(String args[])  
    {  
        int x = 9;  
        if (x > 10)  
            System.out.println("i is greater than 10");  
        else  
            System.out.println("i is less than 10");  
        System.out.println("After if else statement");  
    }  
}
```

Output:

i is less than 10
After if else statement

- ***Nested if statement***

Nested if statement is if inside an if block. It is same as normal if...else statement but they are written inside another if...else statement.

Syntax:

```
if(condition1) {  
    Statement 1; //executed when condition1 is true  
    if(condition2) {  
        Statement 2; //executed when condition2 is true  
    }  
    else {  
        Statement 3; //executed when condition2 is false  
    }  
}
```

Example:

```
class nestedifTest
{
    public static void main(String args[])
    {
        int x = 25;

        if (x > 10)
        {
            if (x%2==0)
                System.out.println("i is greater than 10 and even number");
            else
                System.out.println("i is greater than 10 and odd number");
        }
        else
        {
            System.out.println("i is less than 10");
        }
        System.out.println("After nested if statement");
    }
}
```

Output:

i is greater than 10 and odd number
After nested if statement

- **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.

Let us understand it through one example.

```
class switchDemo{
    public static void main(String args[]){
        int i=2;
        switch(i){
            case 0:
                System.out.println("i is 0");
                break;
            case 1:
                System.out.println("i is 1");
                break;
            case 2:
                System.out.println("i is 2");
                break;
            case 3:
                System.out.println("i is 3");
                break;
        }
    }
}
```

```
    case 4:
    System.out.println("i is 4");
    break;
    default:
    System.out.println("i is not in the list");
    break;
}
}
}
```

2.Looping Statements in Java

Looping statements are the statements which executes a block of code repeatedly until some condition met to the criteria. Loops can be considered as repeating if statements. There are 3 types of loops available in Java.

- ***While Loop:-***

While loops are simplest kind of loop. It checks and evaluates the condition and if it is true then executes the body of loop. This is repeated until the condition becomes false. Condition in while loop must be given as a Boolean expression. If int or string is used instead, compile will give the error.

Syntax:

```
while (condition)
{
    statement 1;
}
```

Example:

```
class whileLoopTest
{
    public static void main(String args[])
    {
        int j = 1;
        while (j <= 10)
        {
            System.out.println(j);
            j = j+2;
        }
    }
}
```

Output:

```
1
3
5
7
9
```

- ***Do...while***

Do...while works same as while loop. It has only one difference that in do...while, condition is checked after the execution of the loop body. That is why this loop is considered as exit control loop. In do...while loop, body of loop will be executed at least once before checking the condition

Syntax:

```
do{
statement1;
}while(condition);
```

Example: here

```
class dowhileLoopTest
{
    public static void main(String args[])
    {
        int j = 10;

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

Output:

10

- ***For loop:-***

It is the most common and widely used loop in Java. It is the easiest way to construct a loop structure in code as initialization of a variable; a condition and increment/decrement are declared only in a single line of code. It is easy to debug structure in Java.

Syntax:

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

Example:

```
class forLoopTest
{
    public static void main(String args[])
    {
        for (int j = 1; j <= 5; j++)
            System.out.println(j);
    }
}
```

Output:

1
2
3
4
5

3.Branching Statements in Java

Branching statements jump from one statement to another and transfer the execution flow. There are 3 branching statements in Java.

- ***Break:-***

Break statement is used to terminate the execution and bypass the remaining code in loop. It is mostly used in loop to stop the execution and comes out of loop. When there are nested loops then break will terminate the innermost loop.

Example:

```
class breakTest
{
    public static void main(String args[])
    {
        for (int j = 0; j < 5; j++)
        {
            // come out of loop when i is 4.
            if (j == 4)
                break;
            System.out.println(j);
        }
        System.out.println("After loop");
    }
}
```

Output:

0
1
2
3
4
After loop

- ***Continue***

Continue statement works same as break but the difference is it only comes out of loop for that iteration and continue to execute the code for next iterations. So it only bypasses the current iteration.

Example:

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

```
for (int j = 0; j < 10; j++)
{
    // If the number is odd then bypass and continue with next value
    if (j%2 != 0)
        continue;    // only even numbers will be printed
    System.out.print(j + " ");
}
}
```

Output:
0 2 4 6 8

INPUT AND OUTPUT STREAMS IN JAVA**1.Accepting the Input from keyboard:**

1.1HardCoded Application: The application in which values are directly specified to perform validation and calculation is technically called hardcoded application.

Example:

```
class A{
    psv main(Stirng rgs[]){
        int a=10;
        int b=20;
    }
}
```

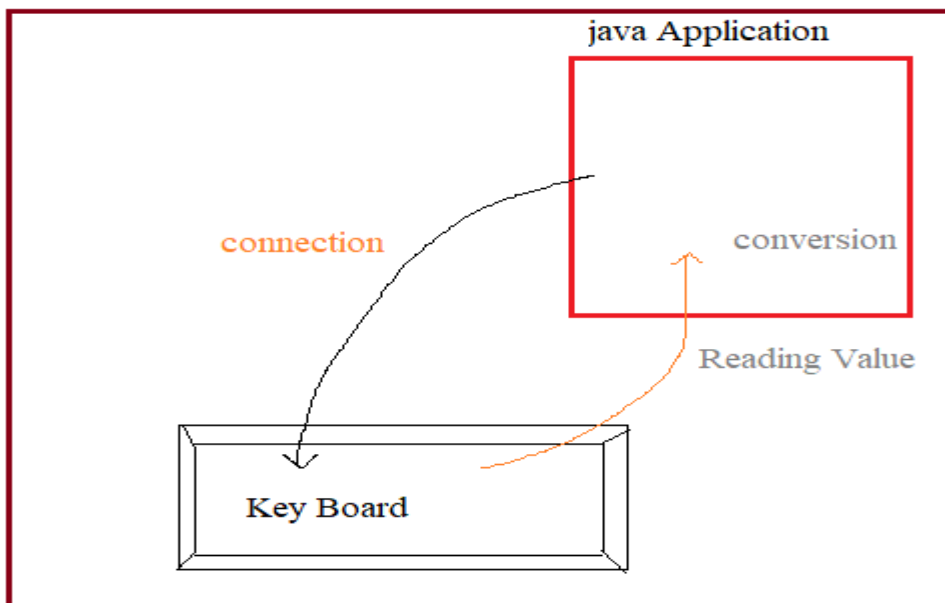
1.2.Dynamic Application:The application which reads input from keyboard at the time of execution is called “dynamic application”.

1.3.Problems with HardCoded application:

- a. The application can always be executed with same values specified in program.
- b. if want to change a values, we must modify values in source code then we must recompile the source code.

Real Time Example:

Let Us assume, ATM application is static Application. It does ‘t takes with draw amount from User at run time. Problem is that Bank customer always draw same amount from his account. He can’t take less amount and he can’t take greather amount than hard code amount which is specified in ATM application.

Different Activities we must do to read values from keyboard:

Step1: Java program has to be connected with keyboard.For Connecting , We must use Scanner class.

Step2: Java program has to read value from keyboard.

Step3:Taken value from keyboard is always String DT. Convert value into its original DT.

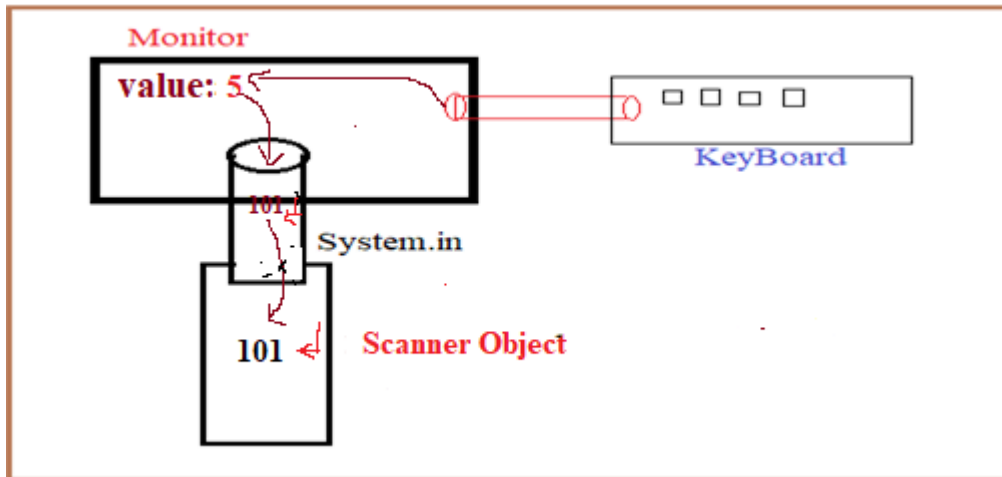
For converting , We must use proper method of Scanner class.

2. Reading Input with java.util.Scanner Class:

It is class. It is available in 'java.util' package.

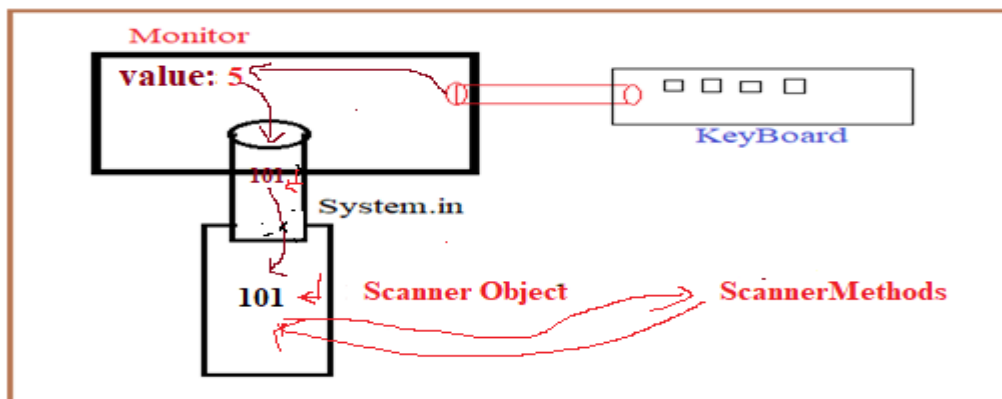
Syntax:

```
Scanner scr=new Scanner(System.in)
```



This class has following Methods. These methods read data as string from scanner object and these methods converts it into corresponding datatype data.

1. `nextByte()`
2. `nextShort()`
3. `nextInt()`
4. `nextLong()`
5. `nextFloat()`
6. `nextDouble()`
7. `nextBoolean()`
8. `nextLine()`
9. `next()`

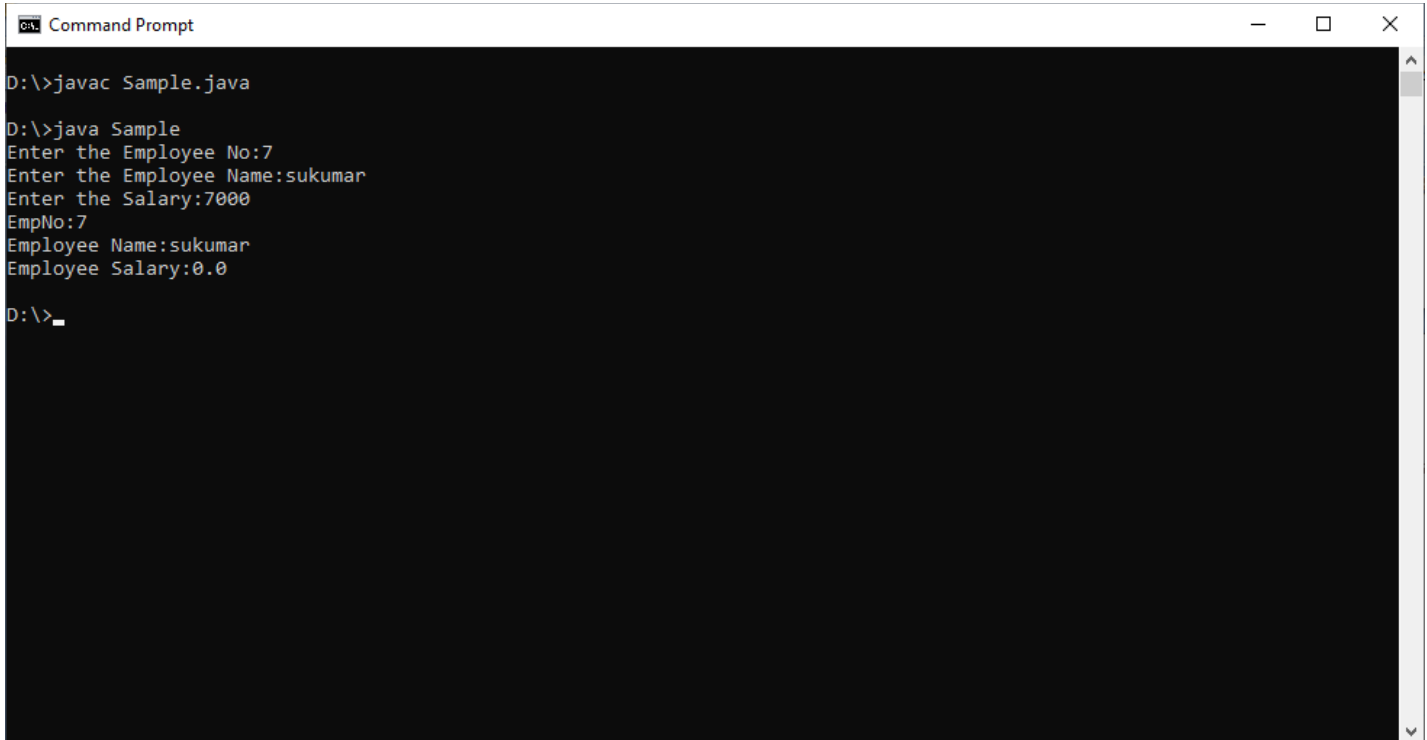


Diff between nextLine() and other methods:-The nextLine() method read all data including 'enter' from scanner object. The remaining methods read data upto before the 'space' or 'enter' character from scanner object.

Example:-

```
import java.util.Scanner;
class Employee
{
    byte empno;
    String ename;
    float salary;
    public void sleep()
    {
        System.out.println("Employee can sleep at home");
    }
    public void teach()
    {
        System.out.println("Employee teaches programming languages only");
    }
    public void setData()
    {
        Scanner s1=new Scanner(System.in);
        System.out.print("Enter the Employee No:");
        empno=s1.nextByte();
        System.out.print("Enter the Employee Name:");
        s1.nextLine();
        ename=s1.nextLine();
        System.out.print("Enter the Salary:");
        s1.nextFloat();
    }
    public void getData()
    {
        System.out.println("EmpNo:"+empno);
        System.out.println("Employee Name:"+ename);
        System.out.println("Employee Salary:"+salary);
    }
}

class Sample {
public static void main(String args[])
{
    Employee e1=new Employee();
    e1.setData();
    e1.getData();
}
```



```
Command Prompt

D:\>javac Sample.java

D:\>java Sample
Enter the Employee No:7
Enter the Employee Name:sukumar
Enter the Salary:7000
EmpNo:7
Employee Name:sukumar
Employee Salary:0.0

D:\>
```

3. Displaying Output with System.out.println():

Where System is class name . It is final class.

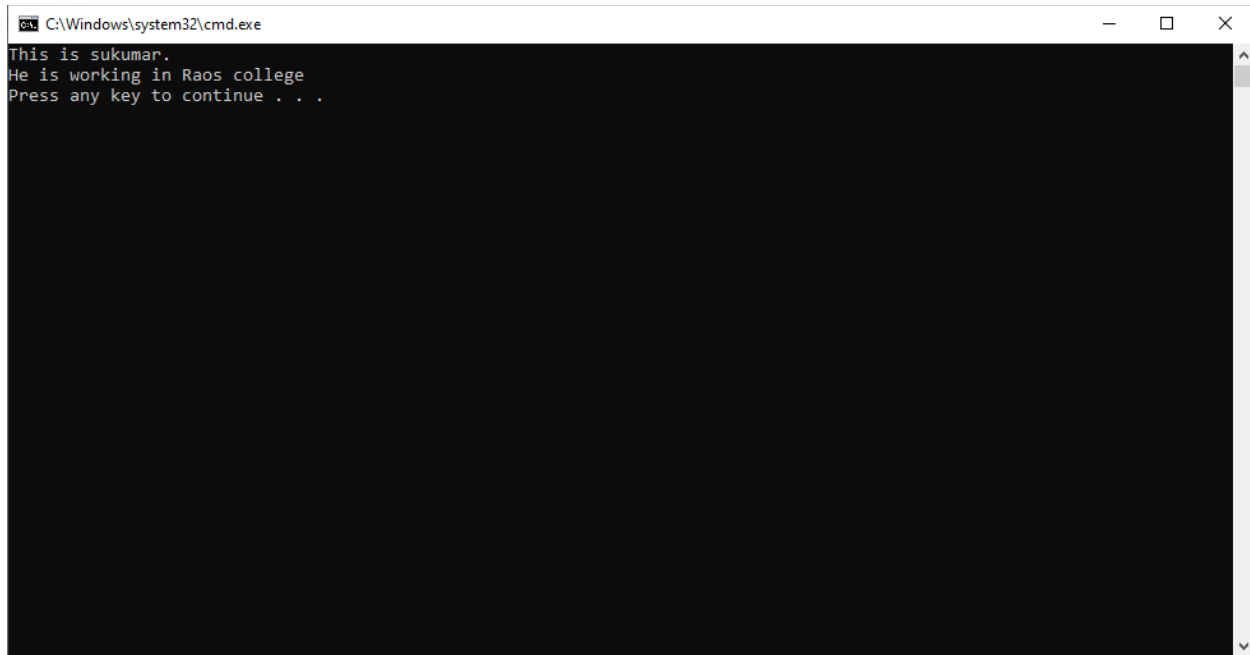
Out is static and final variable of System and is type of PrintStream class.

println():-The println() method also displays the result on the console and moves the cursor to the next line. **It can also work without arguments.**

Syntax: public void println([Argument]);

Example:-

```
class Clerk extends BankAccount
{
    public static void main(String arg[])
    {
        System.out.println("This is sukumar.");
        System.out.println("He is working in Raos college");
    }
}
```



```
C:\Windows\system32\cmd.exe
This is sukumar.
He is working in Raos college
Press any key to continue . . .
```