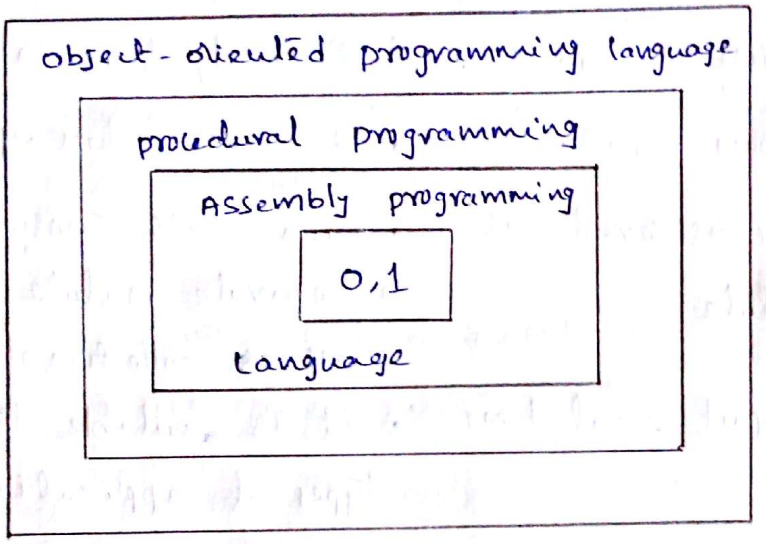
* Introduction *

* Language: It is a communication barrier, used by us to communicate with each other. In terms of programming Language is a barrier between human & a system or application (software).



→ Languages broadly classified as

- Low level programming language
- Middle level programming language
- High-level programming language

→ In low level programming language, to give instructions to a machine, we use Assembly language. It comprises of Mnemonics.

→ Next, we used middle level programming language for communication. The best known example for this is 'C'. It comes under procedure oriented programming language. It uses well structured steps and procedures to build a program. In simple terms, it is a collection of functions or procedures. Mostly, it uses english words as their identifiers.

→ Next, We started using high level programming languages for communication, the best example for this JAVA. It comes under object oriented programming language.

* Differences between procedure oriented / structure oriented and object oriented programming

procedure oriented programming language	object oriented programming language
• It mainly focus on "process".	• It mainly focus on "data".
• It uses top-down Approach	• It uses bottom-up approach.
• Each function considered as a separate module.	• Each class considered as a separate module, where class is collection of Methods.
• It doesn't support real-time applications.	• It is suitable for all types of applications.
• It is difficult to debug an application and extend any application.	• It is easy to debug and extend any application.
• It doesn't provide any security Ex:- 'C'	• It provides security Ex:- 'Java'
• Less reusability	• More reusability

→ Top-down approach is also called as step-wise approach. In point of 'C', this approach first programmer has to write a code for main function, In that, they will call sub-functions.

→ Bottom-up approach starts with low-level system, then it looks for high level system. In this, first programmer has to write code for modules, then they look for integration of modules.

* Java OOPs Concepts :-

→ OOPs stands for object oriented programming system.

→ object means a real world entity such as pen, chair, table etc.

→ object oriented programming is a methodology & paradigm to design a program using classes and objects.

→ It simplifies the software development and maintenance by providing some features & concepts.

→ The programming language where everything is represented as an object, is known as truly object-oriented programming language.

→ Smalltalk and Java are considered as the truely object-oriented programming language.

→ The following are the concepts (or) features (or) principals of object-oriented programming.

- Object
- Class
- Abstraction
- Encapsulation
- Inheritance
- polymorphism

• object :-

→ An object is a real world entity such as pen, chair, table, car, dog and etc.

→ Objects are key to understanding object-oriented technology.

In general, a real world objects have state and behavior. for example, a pen has state (color, company name, model) and behavior (writing, drawing). Ex: Motor bikes, dogs.

In software, the object's state is represented by variables and behavior is represented by methods.

Def^n:- "An object is a software bundle of variables and related Methods."

Example:- ① Object: car
State : color, make
Behavior: climb hill, SlowDown, Accelerate etc.

② Object : House
State : current location, color
Behavior : close/open main door.

• Class :-

A class is a collection of similar objects. In the real world, you often have many objects of the same kind. For example, your bicycle is just one of many bicycles in the world.

In terms of object-oriented, we say that your bicycle object is an instance of the class of objects known as bicycles.

Def^n:- " A class is a blueprint or prototype, that defines the variables and the methods common to all objects of a same kind.

Example: ① object : Byke
class : Bykes That same characterics.
② object : Dog • Each dog have same variables
class : Dogs and Methods like barking(), hungry()

• Abstraction :-

Abstraction is the concept of hiding the internal details and describing things in simple terms. For example: phone call, we don't know the internal processing.

Def^n:- "Hiding internal details and showing functionality is known as abstraction". Let us take the example of a car, we know that if accelerator pressed, speed will increase but don't know the internal process how speed will be increased.

• Encapsulation :-

Encapsulation is the technique used to implement abstraction in object-oriented programming.

Def^n:- "Binding 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.

→ In simple words, Encapsulation is a process of wrapping code and data into single unit. Let us take an example of a HR in a company. We communicate through HR not directly with the departments. HR is acting as public interface here.

• Inheritance :-

The process by which one class acquires the properties and functionalities of another class is known as inheritance.

Def^n:- "When one object acquires all the properties and behaviors of another (parent) object is known as inheritance".

→ It provides code reusability. It is used to achieve runtime polymorphism.

For example, A child inherits the properties of its parent.

Example: In object-oriented terminology, Mountain bikes, racing bikes and tandem bikes are all sub classes of the Bicycle super class.

→ Each subclass inherits the properties and functionalities of super class 'Bicycle' (eg: speed, cadence, braking ...).

- Polymorphism:-

polymorphism is the concept where an object behaves differently in different situations. There are two types of polymorphism - compile time and runtime polymorphism.

Defⁿ:- "When one task is performed by different ways is known as polymorphism".

For example, to draw something e.g. shape or rectangle etc..

Example: The same message 'Move', the man walks, fish swim and birds fly.

* What is Java :-

Java is a programming language and a platform.

→ Java is a high level, robust, secured and object-oriented programming language.

→ Java is a high Level modern programming language, And it was introduced by "Sun MicroSystems' in 1995. It was developed by a "team under James Gosling".

→ platform is nothing but any hardware or software environment in which a program runs. Since Java has its own runtime environment i.e JRE(Java Runtime Environment)

* Where Java is used?

According to Sun, 3 billion devices run Java. There are many type of applications that can be created using Java programming. Some of them are as follows:

- Standalone Application:

It is also known as desktop application (or) window based application. An application that we need to install on every machine such as media player, antivirus and etc.

→ AWT and swing are used in Java for creating this Appln.

- Web Application:

An application that runs on the server side and creates dynamic page, is called Web application. Eg: irctc.co.in

→ Servelt, JSP, Struts, Jsf technologies are used for creating Web applications in Java.

- Enterprise Application:

An application that is distributed in nature, such as banking applications etc. "EJB" is used for this application

• Mobile Application :

An application that is created for mobiles.

→ currently Android and Java ME are used for creating Mobile applications.

* History of Java :-

→ Java team members (James Gosling, Mike Sheridan and patrick Naughton), initiated the Java language project in June 1991 for digital devices such as set-top boxes, televisions etc.

→ The small team of sun engineers called as "Green Team"

→ Firstly, it was called "Greentalk" by James Gosling and File extension was ".gt".

→ After that, it was called "Oak". Why Oak? Oak is a symbol of strength and choosen as a national tree of many countries like U.S.A, France, Germany and etc.

→ In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.

→ The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "silk", "Jolt", "DNA" etc.

→ According to James Gosling "Java was one of the top choices along with Silk". since Java was so unique, most of team members preferred Java.

→ Java is an island of Indonesia where first coffee was produced (called Java coffee).

→ Notice that Java is just a name.

→ originally developed by James Gosling at sun Microsystems and released in 1995.

* Features of Java (or) Java buzzwords :-

There are many features of Java. They are also known as Java buzzwords.
The features of Java given below

- Simple
- object-oriented
- platform independent
- Secured
- Robust
- Architecture neutral
- portable
- Dynamic
- Interpreted
- High performance
- Multithreaded
- Distributed

• Simple:

According to Sun, Java language is Simple because

→ Syntax is based on C and C++.
→ removed many confusing and rarely-used features
e.g, Explicit pointers, operator overloading etc.
→ No need to remove unreferenced objects because there is automatic garbage collection in Java.
→ It eliminates the complexities of C and C++, therefore Java has been made simple.

• Object-Oriented:

→ object-oriented means we organize our software as a combination of different types of objects that contains both data and behaviour.
→ object-oriented programming (OOPs) is a Methodology that simplify software development and maintenance by providing some concepts & rules.
→ The basic concepts of OOPs are:

* Object
* class
* Inheritance
* polymorphism
* Abstraction
* Encapsulation

- **platform independent:**

→ A platform is the hardware or software environment in which a program runs.

→ There are two types of platforms: software-based and hardware-based. Java provides software-based platform.

→ Java code can be run on multiple platforms e.g. Windows, Linux, Mac os and etc.

→ Java code is compiled by the compiler and converted into byte code. This byte code is a platform independent code.

→ It is achieved by JVM (Java virtual Machine). The philosophy of Java is "Write Once, Run Anywhere (WORA).

- **Secured:**

→ Java is secured because Java does not use explicit pointers and All Java programs runs inside the virtual machine Sandbox.

→ Java uses the public key encryption system for providing security.

- **Robust:**

→ Robust simply means strong. Java is robust programming language because

* Java uses strong Memory Management.
* There are lack of pointers that avoids Security problem.
* There is automatic garbage collection in Java.
* There is exception handling and type checking Mechanism in Java.

→ All these points makes Java robust.

- **Architecture - neutral:**

→ There is no implementation dependent features e.g. size of primitive types is fixed.

→ In 'C' programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in 'Java', it occupies 4 bytes of Memory for both 32 and 64 bit architectures.

- **Portable:**

→ Java is portable because we may carry the Java bytecode to any platform.

→ Java compiler is written in ANSI C with clean portability boundary.

→ The Java programs can run on any hard-ware environment.

- **Dynamic:**

→ Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment.

→ Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

- **Interpreted:**

→ Java byte code is translated on the fly to native machine instructions and is not stored anywhere.

→ Java byte code can be interpreted on any system that provides a Java Virtual Machine (JVM).

- **High performance:**

→ With the use of Just-In-Time compilers, Java enables high performance.

→ Java is faster than traditional interpretation.

→ Just-In-Time (JIT) compiler translates Java byte code directly into native machine code for very high speed performance.

- **Multithreaded:**

→ Java was designed to meet the real-world requirements. To accomplish this, Java supports multi-threaded programming, which allows you to write programs that do many things simultaneously.

→ A thread is like a separate program, executing concurrently.

→ The main advantage of multi-threading is that it doesn't occupy memory for each thread, it shares a common memory area.

- **Distributed:**

→ Java is designed for the distributed environment of the internet. We can create distributed applications in Java.

→ We may access files by calling the methods from any machine on the internet.

→ Java's remote method invocation (RMI) make distributed programs possible.

* Simple Java program :-

→ In this section, we can discuss how to execute a Java program and what are the requirements to execute a Java program.

→ For executing any Java program, you need to

- Install the JDK (Java Development Kit).
- Set path of the jdk/bin directory.
- Create the Java program.
- compile and run the Java program.

* Install the JDK:

JDK - Java Development Kit

→ It is a software development environment for Java applications and applets.

→ JDK includes

- Java compiler (Javac)
  It translates the source code to byte code.
- Java debugging tool (Jdb)
  It is used to run Java program
- Java Runtime Environment (JRE)
  It provides an environment to run any Java program at any platform.
- Java Archiving tool (jar)
  It is used to distribute the Java Appln through network with .jar extension.

→ JDK is available for free at www.oracle.com under Java SDKs and tools → Java SE.

→ Download the latest JDK and install it.

→ on Windows, the JDK will be installed by default directory i.e " C:\program files\Java\jdk1.8.xx ".

* Set path of the Jdk/bin directory:

The path is required to be set for using tools such as Javac, Java etc.

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

- Goto My computer 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.

→ The path of bin folder is look like

"C:\Program Files\Java\jdk1.x.x\bin".

→ To verify that JDK is properly installed, open the command prompt type "Javac" and press "Enter".

* Create the Java program:

→ To create Java program open any editor such as notepad.

→ Type the source code

Ex:-

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

→ Save this file as "Sample.Java".

* Compile and run the Java program:

→ To compile 'Javac' command is used

for ex: Javac Sample.Java
→ We get "Sample.class" file

→ To execute 'Java' command is used

for ex: Java Sample
→ We get o/p 'Hello Java'.

In the above Java program, let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- class is a 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.
- void is the return type of the method, it means it doesn't return any value.
- main is a method, it represents startup of the program. The main method is executed by the JVM.
- String[] args is used for command line arguments.
- System.out.println() is used print statement.
  - System: It is a class, which belongs to Java.lang package.
  - out: It is an output stream object, which is a member of System class.
  - println(): It is a method supported by the output stream object "out". It is used to display any kind of output on the screen.

→ At compile time, Java file is compiled by Java compiler and converts the Java code into bytecode (class file).

→ At runtime, class file is converted into machine understandible Instructions.

* Java comments :-

→ The comments are statements that are not executed by the compiler and interpreter.

→ The comments can be used to provide information, explanation and hide program code.

There are three types of comments in Java.

1. Single Line Comment
2. Multi Line Comment
3. Documenation Comment

1. Single Line comment:

This is used to comment only one line.

Syntax: // This is single line comment

2. Multi Line comment:

This is used to comment multiple lines of code.

Syntax:
```
/*
This
is
multi line
comment
*/
```

3. Document comment:

This is used to create documentation API. To create documentation API, you need to use "javadoc tool".

Syntax:
```
/**
This
is
document
comment
*/
```

→ The javadoc tool creates HTML files for your program with explanation.

* Data types in Java :-

→ Data types represents the different values to be stored in the variable.

→ Java defines eight data types, those are

- byte
- short
- int
- long

} Integer group

- float
- double

} Floating-point group

- char } character group
- Boolean } Boolean group

Datatype	Default Size	Range
byte	1 byte	-128 to 127
short	2 byte	-32,768 to 32,767
int	4 byte	-2147483648 to 2147483647
long	8 byte	-9,223,372,036,854,775,808 to 9,2--807
float	4 byte	$-1.7 * e^{38}$ to $1.7 * e^{38}$
double	8 byte	$-3.4 * e^{308}$ to $3.4 * e^{308}$
char	2 byte	0 to 65,536
Boolean	1 bit	0 or 1.

* variables :-

Variable is a name of memory location, in that we can able to store the value for the particular program.

There are three types of variables

→ Local variable

→ Instance variable

→ Static variable

Ex: `int x = 10;` // where x is variable name.

→ Local variable:

A variable which is declared inside the method is called local variable.

→ Instance variable:

A variable which is declared inside the class but outside the method, is called instance variable. It is not declared as static.

→ Static variable:

A variable that is declared as static is called static variable. It cannot be local.

Example:-

```
class  Vardemo
       {
         int x = 50; // instance variable
         static int m=10; // static variable
         void mains()
          {
            int n=9; // local variable
          }
       }
```

* Constant:-

→ There are several values in the real world which will never change, those are called as constants.

eg:- PI (π) values is 3.142 and a day will always have 24 hrs.

→ A constant in java is used to map or assign an exact and unchanging value to a variable.

→ Java does not directly support constants, However, a static final variable is effectively a constant.

Example: public static final int MAX_VALUE =25;

* The scope and Lifetime of variables :-

→ Each variable in modern programming language has:
- a name
- an address
- a type

→ In addition to above properties, each variable also has:
- a scope
- a Lifetime

• scope:

→ The scope of a variable is the locations/places/range in a program where the variable is accessible/visible.

→ We can declare variables within any block.

→ Block is begun with an opening curly brace and ended by a closing curly brace.

→ one block equal to one new scope in Java.

→ A scope determines what variables are visiable to other parts of your program and also determines the lifetime of those objects.

• Lifetime:

→ The Lifetime of a variable is the location (i.e place) where the variable exists.

→ The Lifetime refers to the amount of time a variable exists.

→ variables are created when their scope is entered, and destroyed when their scope is left. This means that a variable declared with in a method will not hold their values outside the method.

→ variables are created and destroyed while the program is running.

* operators :-

An operator is a symbol that is used to perform operations. There are many types of operators in Java such as

- Arithmetic operators

  +, −, *, /, %, ++, --

- Relational operators

  ==, !=, >, <, >=, <=

- Logical operators

  &&, ||, !

- Bitwise operators

  & - Bitwise AND
  | - Bitwise OR
  ^ - Bitwise exclusive OR

Ex:

a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

  << - Left shift. Ex: a = 0001000, b = 2
  >> - Right shift. a << b 0100000
  a >> b 0000010

- Assignment operators

  =, +=, -=, *=, /=, %=

- conditional operator

  exp ? value1 : value2

* operator precedence & hierarchy :-

Operators	precedence
postfix	expr++, expr--
prefix	++expr, --expr
Multiplicative	*, /, %
additive	+, -
shift	<<, >>
relational	<, >, <=, >=
equality	==, !=
bitwise AND	&

bitwise exclusive OR	^
bitwise OR	|
logical AND	&&
logical OR	||
conditional	? :
Assignment	=, +=, -=, *=, /=, %=

* Expression :-

An expression is a construct made up of variables, operators and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

Examples:

```
int marks =25;
int ExtMarks= 75;
int Total =0;
Total = marks + ExtMarks;
```

* Type conversion and casting :-

• Type conversion :-

It converts the one data type into another. If both are compatiable, then Java compiler will perform the type conversion automatically.

for example converting a int into float,
converting a float into double.

Ex:-

```
int i=100;
Long l= i;
float f= l;
```

o/p:

```
i value 100
l value 100
f value 100.0
```

→ Type conversion is done by Java compiler, but remember we can store a large data type into the other.

• Type casting :-

When a user can convert the one higher data type into lower data type then it is called as the type casting.

If both are incompatiable types, we must type casting.

Example :-

```
double d = 10.04;
long L = (long)d;
int i = (int)L;
```

o/p:
d value 10.04
L value 10
i value 10.

Note:- Type conversion is done by compiler and Type casting is done by user.

Example :-

```
class TypeConversion
{
  public static void main(String[] args)
  {
    int x = 1024;
    float y;
    y = x;
    System.out.println("y value is "+y);
  }
}
```

o/p:- y value is 1024.0

Example :-

```
class TypeCasting
{
  public static void main(String[] args)
  {
    double d = 10.04;
    long L = (long)d;
    int i = (int)L;
    System.out.println("d value is "+d);
    System.out.println("L value is "+L);
    System.out.println("i value is "+i);
  }
}
```

o/p:-
d value is 10.04
L value is 10
i value is 10

* Enumerated Types:-

An enum type is a special datatype that contains fixed set of constants.

In Java programming, you define an enum type by using the 'enum' keyword.

Example:

```
public enum Day {
        Sunday, Monday, Tuesday, Wednesday, Thursday,
        Friday, Saturday
        }
```

you should use enum types any time you need to repre - sent a fixed set of constants.

program:-

```
public class EnumExample {
  enum Day {
  Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
   Saturday
   }
  public static void main (String[] args)
   {
     Day yesterday = Day.Thursday;
     Day today = Day.Friday;
     Day tomorrow = Day.Saturday;
   System.out.println ("Today is " + today);
   System.out.println ("Tomorrow will be" + tomorrow);
   System.out.println ("yesterday was" + yesterday);
   }
}
```

o/p: Today is Friday
Tomorrow will be Saturday
yesterday was Thursday.

* Conditional statements:-

These are used to check the condition and execute the set of statements based on condition.

The Java supports following conditional statements

→ If-else statements

→ switch statement

→ If-else statement:-

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
- If-else statement
- If-else-if ladder.

- **If statement:**

Syntax:-

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

- **If-else statement:**

Syntax:-

```
if (condition)
{
// code if condition is true
}
else
{
// code if cond'n is false
}
```

- **If-else-if ladder:**

Syntax:

```
if (condition1)
{
// code if condition1 is true
}
else if (condition2)
{
// code if condition2 is true
}
...
else
{
// code if all conditions are false
}
```

Example: Demonstrate if-else statements.

```
public class IfElseDemo
{
  public static void main (String [] args)
  {
    int marks = 76;
    char grade;
    if (marks >= 90)
    {
      grade = 'A';
    }
    else if (marks >= 80)
    {
      grade = 'B';
    }
    else if (marks >= 70)
    {
      grade = 'C';
    }
    else if (marks >= 60)
    {
      grade = 'D';
    }
    else if (marks >= 50)
    {
      grade = 'E';
    }
    else
    {
      grade = 'F';
    }
    System.out.println ("Grade is " +grade);
  }
}
```

output:- Javac IfElseDemo.Java
Java IfElseDemo
Grade is C.

→ Switch-case statement :-

The switch-case statement tests the value of given variable against a list of case values and when a match is found, a block of statements associated with that case is executed.

Syntax :-

```
switch (expression)
{
    case value1 : statements;
                  break;
    case value2 : statements;
                  break;
                  :
    default : statements;
}
```

Example :- Demonstrate switch-case statement.

```
public class SwitchDemo
{
 public static void main (String[] args)
 {
  int day = 2;
  switch (day)
  {
  case 1 : System.out.println ("sunday"); break;
  case 2 : System.out.println ("Monday"); break;
  case 3 : System.out.println ("Tuesday"); break;
  case 4 : System.out.println ("Wednesday"); break;
  case 5 : System.out.println ("Thursday"); break;
  case 6 : System.out.println ("Friday"); break;
  case 7 : System.out.println ("saturday"); break;
  default : System.out.println ("Invalid choice");
  }
}
}
```

O/P : Monday

* Loop statements :-

The Java supports following looping statements.

Those are
- while loop
- Do-while loop
- for loop

- 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
 }
```

- Do-while loop:

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.

syntax:

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

- for loop:

If the number of iteration is fixed, it is recommended to use for loop.

There are three types of for loop in Java.

- Simple for loop
- For-each loop
- Labeled for loop

- Simple For loop:

Syntax:

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

- Foreach loop:

Syntax:-

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

- Labeled For loop:

Syntax:

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

Examples:- While

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

o/p:- 1
2
3
4
.
.
.
10

Example: Do-While

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

o/p: 1 2 3 ... 10

Example: For

```
public class ForExample
{
public static void main(String[] args)
{
 for(int i=1; i<=10; i++)
  {
   System.out.println(i);
  }
}
}
```

o/p: 1 2 3 ... 10

* Break and continue statements :-

→ The statements break and continue alter the normal control flow of compound statements.

→ The break statement immediately jumps to the end of the appropriate compound statement (loop).

→ The continue statements immediately jumps to the next iteration (if any) of the appropriate loop.

• **break:-**

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

Syntax:- break;

Example:

```
public class BreakDemo
{
public static void main(String[] args)
{
 for(int i=1; i<=10; i++)
 {
   if (i==5)
   {
     break; // Terminate loop if i is 5
   }
  System.out.println(i);
 }
 System.out.println("Loop is over.");
}
}
```

output: C:\>Javac BreakDemo.Java

C:\>Java BreakDemo

1
2
3
4
Loop is over.

→ In simple words, the break keyword is used to breaks (stopping) a loop execution.

• Continue :-

When a continue statement is encountered inside the body of a loop, remaining statements are skipped and loop proceeds with the next iteration.

Syntax: continue;

Example:

```
public class ContinueDemo
{
  public static void main(String[] args)
  {
   for(int i=1; i<=10; i++)
    {
      if(i%2==0)
      {
        continue; // skip next statement if i is even
      }
     System.out.println(i);
    }
   }
}
```

output: C:\> Javac ContinueDemo.java

C:\> Java ContinueDemo

1
3
5
7
9

→ In simple words, the continue keyword is used to skip the particular recursion only in a loop execution.

* Simple Java Standalone programs :-

1. Fibonacci Series in Java

```
class Fibonacci
{
public static void main(String args[])
{
 int n1=0, n2=1, n3, i, count=10;
 System.out.print(n1+" "+n2); // printing 0 and 1
for(i=2; i<count; i++)
{
 n3=n1+n2;
 System.out.print(" "+n3);
 n1=n2;
 n2=n3;
}
}
}
```

output: 0 1 1 2 3 5 8 13 21 34

2. prime Number

```
class primeNum {
    public static void main(String args[])
    {
      int num=17; //int num=Integer.parseInt(args[0]);
      int flag=0;
      for(int i=2; i<num; i++)
      {
        if(num%i==0)
        {
          System.out.println(num + " is not a prime");
          flag=1;
          break;
        }
      }
      if(flag==0)
        System.out.println(num + " is a prime number");
    }
}
```

output: 17 is a prime number

3. palindrome Number:

```
class palindrome
 {
  public static void main(String args[])
  {
   int r, sum=0, temp;
   int n=454; // n= Integer.parseInt(args[0]);
   temp=n;
   while(n>0)
   {
    r=n%10;
    sum=(sum*10)+r;
    n=n/10;
   }
   if(temp==sum)
   {
    System.out.println(" palindrome Number");
   }
   else
   {
    System.out.println(" Not palindrome");
   }
  }
 }
```

output:-
palindrome Number

4. Factorial

```
class Factorial
 {public static void main(String args[])
  {
   int i,fact=1;
   int num=5; // n= Integer.parseInt(args[0]);
   for(i=1;i<=num;i++)
    {
     fact=fact*i;
    }
   System.out.println("Factorial of "+num+" is: "+fact);
  }
 }
```

output:- Factorial of 5 is 120

5. Armstrong Number

```
class ArmstrongNum
{
public static void main (String args[])
{
 int sum=0, r, temp;
 int n=153; // n=Integer.parseInt(args[0]);
 temp = n;
 while (n>0)
 {
  r = n%10;
  sum = sum + (r*r*r);
   n=n/10;
 }
 if (temp == sum)
 {
   System.out.println ("Armstrong number");
 }
 else
  {
   System.out.println ("Not a Armstrong Number");
  }
}
}
```

output: Armstrong Number

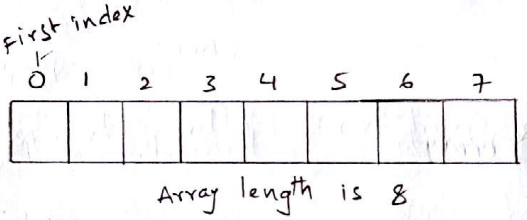
* Arrays:-

→ Till now, we have discussed how to declare variables of a particular datatype, which can store a single value. There are the situations where we might wish to store a group of similar type of values in a variable.

→ It can be achieved by a special kind of data structure known as arrays.

* An array is a collection of similar data elements and it is a data structure where we store similar elements. We can store only fixed set of elements in an array.

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



- Advantages of array:

→ code optimization: It makes the code optimized, we can retrive & sort the data easily.

→ Random access: We can get any data located at any index position.

- Disadvantage of array:

→ Size Limit: We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

There are two types of arrays:

- Single dimensional array
- Multi dimensinal array

- **Single dimensional array:**

It is an array with only one dimension (or) index. It can be visualized as a single row or a column.

Declaration:

```
Datatype variable [] = new datatype [size];
(or)
Datatype variable[];
variable = new datatype [size];
```

// Declaration & Initialization

```
(or) datatype variable [] =
     {element1, element2,... ,n}
```

Ex :-

```
int marks [] = new int [5];
```

(or)

```
int marks [] = {45, 60, 70, 80, 9};
```

```
int marks [];
marks = new int [5];
```

Example :-

```
class SingleDarray
{
 public static void main (String args [])
 {
  int marks [] = new int [6]; // Declaration
  marks [0] = 60; // Initialization
  marks [1] = 58;
  marks [2] = 70;
  marks [3] = 80;
  marks [4] = 78;
  marks [5] = 89;
 // printing values in array
  for (int i=0; i < marks.length; i++)
  {
    System.out.println (marks [i]);
  }
}
}
```

(or)

```
int marks [] =
    {60, 58, 70, 80, 78, 89};
```

output:

```
60
58
70
80
78
89
```

- **Multi dimensional array:**

It is an array with two or more dimensions or indexes. It can be visualized as a matrix of rows and columns.

Lets take a two-dimensional array, In this case, data is stored in row and column based index (also known as Matrix form).

Syntax for array declaration (two)

```
Datatype variable [][] = new datatype [rows] [col];
```

(or)

```
Datatype variable[][];
variable = new datatype [rows] [col];
```

(or)

```
datatype variable[][] =
{ { e11, e12 -- } { e21, e22 -- } -- }
```

Ex:-

```
int marks[][] = new int [3][6];
```

(or)

```
int marks[][];
marks = new int [3][6];
```

(or)

```
int marks[][] =
{ {48, 65 -- } {78, 62 -- } { } -- }
```

Example:-

```
class TwoDarray
{
public static void main(String args[])
{
  int marks[][] = new int[3][6];
  marks[0][0] = 48;
  marks[0][1] = 52;
  marks[0][2] = 63;
    :
    :
  marks[2][4] = 75;
  marks[2][5] = 82;

// printing 2-D array
for (int i=0; i < marks.length; i++) {
for (int j=0; j < marks[i].length; j++) {
    System.out.println (Marks[i][j] + " ");
  }
  System.out.println ();
} } }
```

```
// Declaring & Initializing 2D array.
int marks[][] =
(or) {{48,52,63,65,66,71},{67,78,87,65,64,
  76},{78,67,65,77,75,82}};
```

output:-

```
48 52 63 65 66 71
67 78 87 65 64 76
78 67 65 77 75 82
```

* Console input and output (or) console class :-

→ In this, We learn about java.io.Console class. This class provides convenient methods for reading input and writing output to standard streams (Keyboard and display) in command-line (console) programs. Note:- console class comes under "java.io" package

→ The Console class provides following methods,

those are

- printf() - Writes a formatted string to console's output stream.
- readLine() - Reads a single line of text from console's input stream.
- readPassword() - Reads a password from console input stream with echoing disabled.

Example :-

```
import java.io.*; //package
class ConsoleioDemo
{
public static void main(String args[])
{
Console c= System.console();
c.printf("Enter your name: "); //console output
String name = c.readLine(); // console input
c.printf("Enter your company name : ");
String cname = c.readLine();
c.printf("congrats %s \n", name);
c.printf("you are the Employee of company : %s", cname);
}
}
```

output :-

```
Javac ConsoleioDemo.Java
Java ConsoleioDemo
Enter your name : Madhu
Enter your company name : TKRCET
congrats Madhu
you are the Employee of company : TKRCET
```

* Scanner class :-

→ The Scanner class comes under java.util package. And it is used to getting input from user.

→ System is a class in the java.lang package. This class has three predefined variables : in, out and err.

- in refers to standard input stream (Keyboard)
- out refers to standard output stream (Monitor)
- err refers to standard error output stream (Monitor).

→ The scanner class uses System.in object & variable to get input from Keyboard (Standard input stream).

The scanner class have following Methods

- nextLine() - It returns the input as a string.
- nextInt() - It returns the input as an integer.
- nextFloat() - It returns the input as a Float.
- nextLong() - It returns the input as a long.
- nextShort() - It returns the input as a short.
- nextDouble() - It returns the input as a double.

Example :-

```
import java.util.*;
class ScannerDemo {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter your name: ");
String name = sc.nextLine();
System.out.println("Enter your age : ");
int age = sc.nextInt();
System.out.println("Hai " +name);
System.out.println("your age is " +age);
}
}
```

output :-

```
Javac ScannerDemo.java
Java ScannerDemo
Enter your name
Madhu
Enter your age
30
Hai, your age is 30
```

* Classes and objects :-

Java is an object-oriented programming language. classes and objects are basic building blocks of OOP.

* classes :-

A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.

(or) A class is a group of objects that has common properties.

→ A class in java can contain: data member, method, constructor, block, class and interface.

Syntax to declare a class

```
class <class_name>
{
 //variables declaration
 // Methods declaration
}
```

→ A class can be declared using the keyword 'class' followed by the name of the class that you want to define.

→ The class body contains two different sections: variable declaration and Methods declaration.

* Objects :-

An object is a software bundle of variables and related Methods. An object is an instance of a class. i.e by using object, we can able access variables and methods in that class.

Syntax to declare an object

```
class_name object_name = new class_name();
```

(or)

In general, Syntax is `type objectname;`

where type is name of class. because a class is an user-defined data type.

→ The keyword 'new' is used to allocate memory at runtime.

• Instance variable:

A variable that is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when object is created. That is why, it is known as instance variable.

Example of object and class:

```
class Student
{
  int id = 521;                    } //data Members (also instance variable)
  String name = "Madhu";           }
  public static void main (String args[]) -> //method.
  {
    Student s1 = new Student(); //creating an object of Student
    System.out.println ("your R.NO is:" +s1.id);
    System.out.println ("your name is:" +s1.name);
    int age = 30;                  } //local variables
    String branch = "CSE";         }
    System.out.println ("your age is :" +age);
    System.out.println ("your branch is :" +branch);
  }
}
```

o/p: javac Student.java
java Student
your R.NO is : 521
your name is : Madhu
your age is : 30
your branch is : CSE

In this example, we have created a Student class that have two data members id and name. We are creating the object of the Student class by new keyword and printing the objects value.

* Methods:-

Def^n: A java method is a collection of statements that are grouped together to perform an operation.

→ None of the methods can be declared outside the class.

→ All methods have a name that starts with a lowercase character.

→ In Java, Methods are used to,

- Make the code reusable.
- Simplify the code.
- Top-down programming.

→ Java supports two types of methods, those are

• Instance methods:- These are used to access/manipulate the instance variables and also access class variables.

• Class methods:- These are used to access class variables but cannot access the instance variables unless and until they use an object for that purpose.

Syntax for Method Declaration

```
[modifiers] return_type method_name (parameter-list)
        {
          statements list // Method body
        }
```

In the above syntax,

* modifiers (optional) defines the scope (public, protected, default or private).
* return_type - It can be either void (if no value is returned) or if a value is returned.
* Method_name - The method name must be a valid Java identifier. (Method name starts with lower case letter).

* Parameter List : you can pass one or more values to a method by listing the values in parentheses following method name.

* Method body : The method body defines what the methods does with the statements.

Example :-

The following example to demonstrate how to define a method and how to call it-

```
import java.io.*;
public class MaxNumber
{
public static void main (String args[])
{
   Console c= System.console();
   c.printf("Enter first number \n");
   int a = Integer.parseInt(c.readLine());
   c.printf("Enter second number \n");
   int b = Integer.parseInt(c.readLine());
   int maxc = MaxFunction(a,b);
   System.out.println("Maximum Number is = " + maxc);
 }
public static int maxFunction(int n1, int n2)
{
    int max;
    if (n1 > n2)
       max = n1;
    else
       max = n2;
    return max;
 }
}
```

O/p: Javac MaxNumber.Java
java MaxNumber
Enter first Number
56
Enter second Number
51
Maximum Number is = 56.

Example: A method with void return type.

```
import java.io.*;
Public class Grade
{
 public static void main (String args[])
 {
  Console c = System.consde();
  c.printf("Enter your marks in blw 0 to 100 \n");
  int marks = Integer.parseInt(c.readLine());
  grade(marks);
 }
 public static void grade (int tmarks)
 {
   if (tmarks >= 90)
     System.out.println(" Grade A");
   else if (tmarks >= 70)
     System.out.println(" Grade B");
   else if (tmarks >= 50)
     System.out.println(" Grade C");
   else
     System.out.println(" Grade D");
  }
}
```

o/p: javac Grade.java
java Grade
Enter your marks in blw 0 to 100
92
Grade A.

Note:- If a method is declared using static keyword, then no need to create an object to access it. otherwise, we can create object and then access methods by using that object.

Example: The following example shows method calling with an object creation.

```
import java.io.*;
class Calc
{
public static void main(String args[])
{
 Calc obj = new Calc(); // object creation for class Calc.
 Console c = System.console();
 c.printf("Enter first number");
 int x = Integer.parseInt(c.readLine());
 c.printf("Enter second number");
 int y = Integer.parseInt(c.readLine());
 obj.sum(x,y);
 obj.sub(x,y);
}
void sum(int a, int b)
 {
  System.out.println("sum is :" +(a+b));
 }
void sub(int a, int b)
 {
  System.out.println("sub is :" +(a-b));
 }
}
```

O/P:- javac Calc.java

```
java Calc
Enter first number
20
Enter second number
5
Sum is : 25
sub is : 15
```

* Method overloading :-

→ If a class have multiple methods by same name but different parameters, it is known as Method overloading.

→ In java whenever a method is being called, first the name of the method is matched and then, the number and type of arguments passed to that methods are matched.

→ Method overloading is a feature that allows a class to have two or more methods having same name but different parameters.

There are two different ways of method overloading

- Method overloading by changing data type of arguments
- Method overloading by changing no. of arguments.

Example :-
Method overloading by changing datatype of arguments.

```
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.
 }      %% javac calculate.java
}           java Calculate
            Sum is 13
            sum is 8.4
```

Example :-

Method overloading by changing no. of argument.

```
class Calsum
{
 void sum(int a, int b)
 {
  System.out.println(a+b);
 }
 void sum(int a, int b, int c)
 {
  System.out.println(a+b+c);
 }
 public static void main(String args[])
 {
  Calsum obj = new Calsum();
  obj.sum(10,10,10); // sum() with 3 parameters
  obj.sum(20,20); // sum() with 2 parameters
 }
}
```

o/p: javac Calsum.java
java Calsum
30
40

* Constructor :-

→ constructor in java is a special type of method that is used to initialize the object. Java constructor is invoked at the time of object creation.

→ It constructs the values i.e provides data for the object that is why it is known as constructor.

→ Java constructors are the methods which are used to initialized objects.

There are basically two rules defined for the constructor

→ constructor name must be same as its class name.

→ Constructor must have no explicit return type.

There are two types of constructors

- Default constructor
- parameterized constructor

- **Default constructor :-**

A constructor that have no parameter is known as default constructor. This is used to provides default values to an object.

Example :-

```
class DefaultConstr
{
DefaultConstr() // constructor method
{
System.out.println("Default Constructor method called.");
}
public static void main(String args[])
{
DefaultConstr dc = new DefaultConstr();
}
}
```

output:- javac DefaultConstr.java
java DefaultConstr
Default Constructor method called.

- **parameterized constructor :-**

→ A parameterized constructor is a constructor, that has parameters. (or) A constructor that have parameters is known as parameterized constructor.

→ This is used to provide different values to the distinct objects.

Example :-

In the following example, we have created the constructor of student class that have two parameters.

```
class Students
{
 int id;
 String name;
 Students (int i, String n) // parameterized constructor.
 {
  id = i;
  name = n;
 }
 void display() // Method
 {
  System.out.println(id + "  " + name);
 }
 public static void main(String args[])
 {
  Students S1 = new Students (521, "Madhu");
  Students S2 = new Students (512, "Hari");
  S1.display();
  S2.display();
 }
}
```

output:-

```
javac Students.java
java Students
521  Madhu
512  Hari
```

* Constructor overloading :-

Constructor overloading is a technique in java in which a class can have any number of constructors that differ in parameter lists.

The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

Example

```
class Sum
{
  int a,b,c;
Sum(int x, int y) // with 2 parameters  (constructor)
{
  a=x;
  b=y;
}
Sum (int x, int y, int z) // constructor with 3 parameter
{
  a=x;
  b=y;
  c=z;
}
void display()
{
 System.out.println("sum is"+(a+b+c));
}
public static void main(String args[])
{
  Sum s1 = new Sum(10, 11); // With 2 parameters
  Sum s2 = new Sum(10,20,30); // with 3 parameter
  s1.display();
  s2.display();
}
}
```

output: javac Sum.java

java Sum

Sum is 21

Sum is 60

There are some differences between constructors and methods. Those are,

Constructor	Method
* Constructor is used to initia-lize values to an object.	* Method is used to expose behavior of an object.
* Constructor must not have return type.	* Method must have return type.
* Constructor is invoked implicitly	* Method is invoked explicitly
* constructor name must be same as the class name.	* Method name may or may not be same as class name.

* Java Garbage collection :-

In Java, garbage means unreferenced objects. Garbage collection is process of destroyed the unused memory automatically.

In other words, it is a way to destroy the unused objects.

To do so, we were using free() function in C language and delete() in C++. But, in Java it is performed automatically. So Java provides better Memory Management.

An object can be unreferenced in following ways

→ By nulling the reference

Example :- Employee e = new Employee();
e = null;

→ By assigning a reference to another

Example :- Employee e1 = new Employee();
Employee e2 = new Employee();
e1 = e2

In java, the garbage collector of JVM collects only those objects that are created by new keyword. so if you have created any object without new, you can use finalize Method to perform cleanup processing (destroying remaining objects).

finalize() method:

This method invoked each time before the object is garbage collection. This method is used to destroying the objects which are not created by "new" keyword.

gc() method:

The gc() method is used to invoke the garbage collector to perform cleanup processing. But this method does not guarantee that JVM will perform the garbage collection. It is only request to the JVM for garbage collection.

Example:-

```
class GarbageDemo
{
public void finalize()
  {
   System.out.println("object is garbage collected");
  }
  public static void main(String args[]
  {
   GarbageDemo g1 = new GarbageDemo();
   GarbageDemo g2 = new GarbageDemo();
    g1 = null;
    g2 = null;
    System.gc();
  }
}
```

o/p:- javac GarbageDemo.java
java Garbage
object is garbage collected
object is garbage collected

* String class :-

Generally, string is a sequence of characters. But in java, String is an object that represents a sequence of characters.

→ The java.lang.String class is used to create string object.

→ In Java, An array of characters works same as java String.

for example:

```
char[] ch={'M','a','d','h','u'};
String s = new string(ch);
```

is same as:

```
String s="Madhu";
```

→ In Java, string is an object, it can be created by using java.lang.String class.

There are two ways to create String object

* By String literal :-

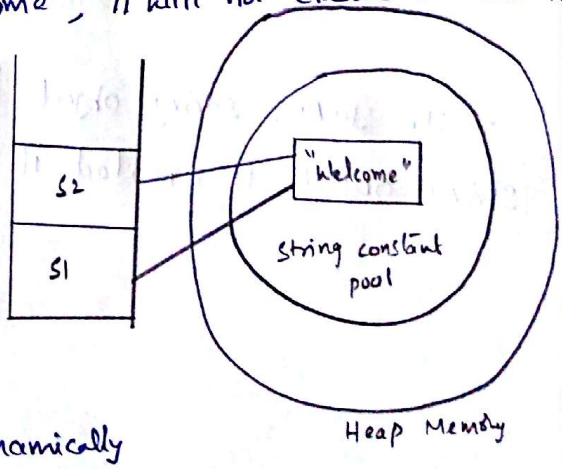
Java String literal is created by using double quotes

Ex:- `String s="Welcome";`

When we create a string literal, the JVM checks the string constant pool first, if the string already exists in the pool, a reference to the pooled instance is returned.
If string doesn't exist in the pool, a new string instance is created and placed in the pool.

for example:

```
String s1="Welcome";
String s2="Welcome"; // will not create new instance.
```



Note:- String objects are stored in a special memory area known as string constant pool. It is an constant table to store String literals.

Note:- A heap is a general term used to (to) any Memory that is allocated dynamically and randomly. (It is allocated by O.S).

* By new Keyword :-

In This, We can string object by using "new" Keyword.

for example :

String s = new String ("Welcome");

In this case, JVM will create a new string object in heap memory and the literal "Welcome" will be placed in the string constant pool.

Example :-

```
class StringExample
{
public static void main (String args[])
{
 String s1 = "Java"; // creating a string by java string literal
 char ch[] = {'s', 't', 'r', 'i', 'n', 'g', 's'};
 String s2 = new String (ch); // converting char array to string
 String s3 = new String ("example"); // Creating string by new keyword
 System.out.println (s1);
 System.out.println (s2);
 System.out.println (s3);
 }
}
```

output :- javac StringExample.java
java StringExample
Java
Strings
example.

→ In Java, string object is immutable That means once a string object is created it cannot be altered.

In Java, String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace() and etc.

* charAt():

This method returns a char value at the given index number. The index starts from 0 (zero).

Syntax:- charAt(int index)

→ It returns char value.

Example:
```
String name = "Madhu";
Char ch = name.charAt(4);
S.o.p(ch);
```
o/p:- u

* concat():

This method is used to combine two strings. And it returns combined string.

Syntax:- concat(String str2)

Example:
```
String s1 = "java string";
s1 = s1.concat(" is immutable");
S.o.p(s1);
```
o/p:- Java string is immutable

* contains():

This method searches the sequence of characters in this string. It returns true if sequence of char values are found in the given string otherwise returns false.

Syntax:- contains(char sequence)

Example:
```
String name = "Madhu Naidu Talikota";
S.o.p(name.contains("Naidu"));
S.o.p(name.contains("Reddy"));
```
o/p:- True
false

* equals():

This method is used to compares the two given strings. If both are equal it returns true otherwise it returns false.

Syntax:- equals(anotherstring)

Example:

```
String s1 = "madhu";
String s2 = "madhu";
String s3 = "MADHU";
S.O.P(s1.equals(s2));
S.O.P(s1.equals(s3));
```

O/P: true
false.

* length():

This method is used to find the length of string. It returns count of talal numbers of characters.

Syntax:- int length();

Example:

```
String s1 = "Madhu";
String s2 = "Tatikota";
S.O.P(s1.length());
S.O.P(s2.length());
```

O/P: 5
9

* substring():

This method returns a part of the string. It is used to extract some part of given string.

Syntax:- substring(int startIndex)
and
substring(int startindex, int endindex)

Example:-

```
String s1 = "Madhu";
S.O.P(s1.substring(2,4));
S.O.P(s1.substring(3));
```

O/P: dhu
hu

* startsWith():

This method checks if This string starts with given prefix. It returns true if this starts with given prefix else returns false.

Syntax:- startsWith(String prefix)

Where prefix is sequence of characters.

Example:- String s1 = "Madhu";
s.o.p(s1.startsWith("Ma"));
s.o.p(s1.startsWith("dh"));

o/p: true
false

* endsWith():

This method checks if this string ends with given prefix. It returns true if this ends with given prefix else returns false.

Syntax:- endsWith(string prefix)

Where prefix is sequence of characters.

Example:- String s1 = "Madhu";
s.o.p(s1.endsWith("hu"));
s.o.p(s1.endsWith("dh"));

o/p: true
false.

* replace():

This method is used to replace old characters with new characters in a given string.

Syntax:- replace(char oldchar, char newchar)

Example:- String s1 = "Java is an OOP";
s.o.p(s1.replace('a','i'));
s.o.p(s1.replace("is","was"));

o/p:- Jivi is in OOP
Java was an OOP

* indexOf():-

This method returns index of given character value or substring. If it is not found, it returns -1, The index counter starts from zero.

Syntax:- indexOf (char ch)
indexOf (char ch, int fromindex)
indexOf (String substring)
indexOf (String substring, int fromindex)

Example:
```
String s= "this is madhu";
S.O.P (s.indexOf ('s')); // it returns 3
S.O.P (s.indexOf ('s', 4)); // it returns 6
S.O.P (s.indexOf ("is")); // it returns 2
S.O.P (s.indexOf ("is", 4)); // it returns 5
```

* toLowerCase():-

This method is used to convert given string into lower case letter.

Syntax:- toLowerCase()

Example:
```
String name= "MAdhu";
S.O.P (name.toLowerCase());
```
O/P:- madhu

* toUpperCase():-

This method is used to convert all characters in given string into upper case letter.

Syntax:- toUpperCase()

Example:
```
String name= "madhu";
S.O.P (name.toUpperCase());
```
O/P:- MADHU.