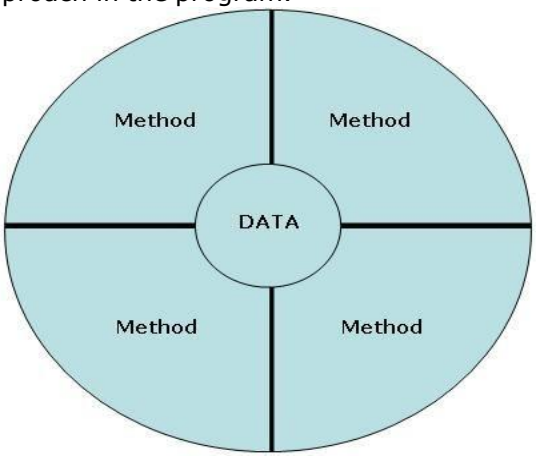
Unit-1

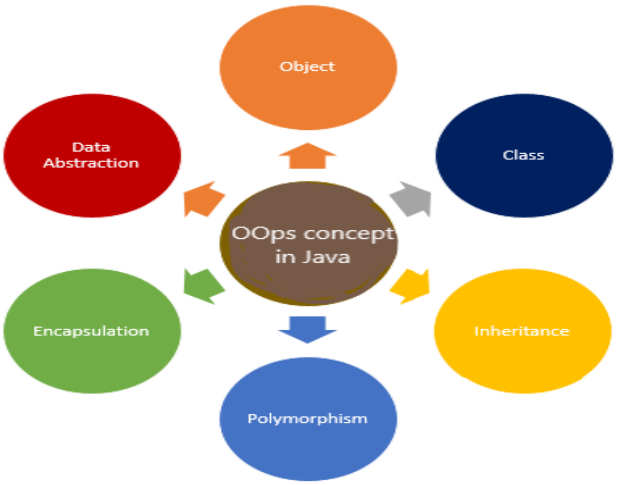
Fundamentals of object-oriented programming: -

Object oriented paradigm: -

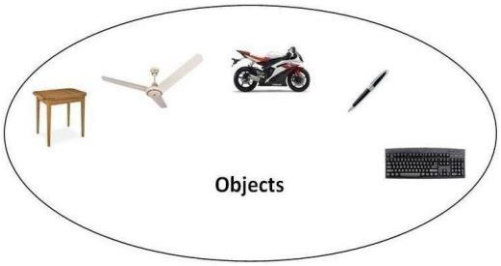
- OOP supports to inheritance, overloading of operator.
- Emphasis on data rather than procedure.
- Programs are divided into objects.
- Functions that operate on the data of an object.
- Data is hidden and cannot be accessed by external functions.
- Objects may pass the data to each other by using functions.
- It is easy to add new data and function in the class.
- It is use the bottom-top approach in the program.



Basic concept of object-oriented programming: - (OOPs Concepts)



- **Objects: -** Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.
  Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.



- **Classes: -** Collection of objects is called class. It is a logical entity. The entire set of data and code of an object can be made a user-defined data type with the help of class.
- **Encapsulation:** - Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.
  A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Capsule

- **Data abstraction:** - Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing. In java, we use abstract class and interface to achieve abstraction.
- **Inheritance: -** When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
- **Polymorphism: -** If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- **Dynamic binding: -** Binding refers to the linking of a procedure call to the code to be executed in response to the call. It is also known as late binding.
- **Message passing: -** Objects communicate with one another by sending and receiving information. A message for an object is a request for execution of procedure, and therefore will invoke the function.

Benefits (Buzzwords or futures) of OOP (JAVA): -

- Benefits of OOP are as under.
  1. **Compiled and Interpreted:** Java is both compiled and interpreted language.
  2. **Platform-Independent and portable:** Java programs can be easily moved from one computer system to another, anywhere & anytime.
  3. **Object- oriented:** All program code and data reside within objects and classes. So JAVA is a true OOP.
  4. **Robust & secure:** Java is a robust language. It provides many safeguards to ensure reliable code. Security becomes an important issue for a language that is used for programming on Internet.
  5. **Distributed:** Java is design as a distributed language for creating application on network. It has ability to share both data & programs. Java applications can open and access remote objects on Internet as easily as they can do in local system.
  6. **Simple, Small & Familiar:** Java is a small & simple language. Many features of C & C++ that are used in JAVA. For example, Java does not support Pointer, pre-processor Header Files, go-to statement, operator overloading, multiple inheritance and many others.
  7. **Multithreading:** JAVA can handle multiple tasks simultaneously. This future is known as multithreading.
  8. **High performance:** Java performance is impressive for an interpreted language, mainly due to the use of intermediate byte-cod
  9. **Dynamic & Extensible:** Java is capable of dynamic linking in new class libraries, method and objects.

Application of JAVA: -

- The programming areas of application of JAVA include:
  1. Real time systems.
  2. Simulation and modelling.
  3. Object oriented database.
  4. Hypertext and hypermedia.
  5. AI and expert system.
  6. Neural network and parallel programming.
  7. Decision support and office automation system.

History of Java: -

The history of java starts from Green Team. Java team members (also known as Green Team), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

For the green team members, it was an advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

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

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

Difference between C and JAVA: -

- Java does not include the C unique statement keywords sizeof and typedef.
- Java does not contain the data type structure and union.
- Java does not define the type modifier keywords auto, extern, register, signed and unsigned.
- Java does not support an explicit pointer type.
- Java adds new operators such as instance-of and >>>.
- Java adds labelled break & continue statement.

Difference between C++ and JAVA: -

- Java does not support operator overloading.
- Java does not have template classes as in C++.
- Java does not support pointer.
- Java has replaced the destructor function with a finalize () function.
- There are no header files in Java.
- Java does not support goto statement.

Benefits of OOP :

- OOP offers several benefits to both the program designer and the user.
- The principal advantages are:

1. Through Inheritance, we can eliminate redundant code and extend the use of existing classes.
2. The principle of data hiding helps the programmer to build secure programs.
3. It is possible to have multiple objects to communicate without any interference.
4. It is easy to partition the work in a project based on objects.
5. Object-oriented system can be easily upgraded from small to large systems.

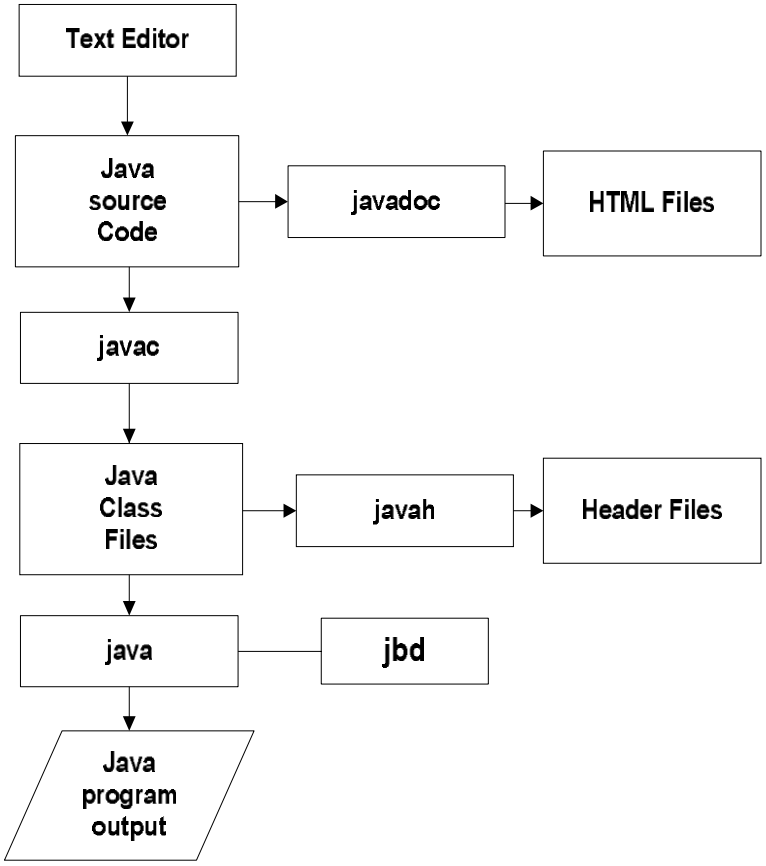
JAVA development kit (JDK): -

- The java development kit comes with a collection of tools that are used for developing and running programs.

Tools	Description
appletviewer (for viewing java applet)	Enables us to run Java applet (without actually using a java-compatible browser)
Java (java interpreter)	Java interpreter, which runs applet and application by reading and interpreting byte code file.
Javac (java compiler)	The Java compiler, which translate Java source code to byte code files that the interpreter understands.
Javadoc (for creating HTML document)	Create HTML – format documentation from java source code files.
Javah (for C header files)	Produce header files for use with native methods.
Javap (java disassembler)	Java disassembler, which enables us to convert byte-code files into a program description.
Jdb (java debugger)	Java debugger, which helps us to fine errors in our program.

- The way these tools are applied to build and run application programs is illustrate in figure.
- Create a JAVA program; we need to create a source code file using a text editor.
- The source code is then compiled using the Java compiler **javac** and executed using the JAVA interpreter **java**.
- The java debugger **jbd** is used to find errors, if any, in the source code.
- A compiled Java program can be converted in to source code with the help of Java disassembles

javap.



Application program interface (API): -

- The java standard library includes hundreds of classes & methods grouped into several functional packages.
- **Language support package: -** A collection of classes & methods required for implementing basic features of Java.
- **Utilities packages: -** A collection of classes to provide utility function such as date & time function.
- **Input/output packages: -** A collection of classes required for input / output manipulation.
- **Network packages: -** A collection of classes for communicating with other computer via Internet.
- **AWT packages: -** The abstract window toolkit package contains classes that implement platform independent graphical user interface.
- **Applet package: -** This is includes a set of classes that allow us to create Java Applet.

JAVA run time environment (JRE): -

- The JRE facilities the execution of program development in Java.
- **Java Virtual Machine (JVM): -** It is a program that interprets the intermediate Java Byte code and generates the desire output. It is because of byte code and JVM concept that programs written in java are high portable.
- **Runtime class Libraries: -**These are a set of core class libraries that are required for the execution of Java program.
- **User interface toolkits: -** AWT and Swing are example of Toolkits that support varied input methods for the users to interact with the application program.
- **Development technologies: -**
  1. Java plug-in: Enables the execution of a Java applet on the browser.
  2. Java Web Start: - Enables remote- deployment of an application. With the help of Web Start, users can launch an application directly from the Web browsers without going through the installation procedure.

Basic structure of JAVA program: -

- A java Program may contain many classes of which only one class define a main Method.
- Classes contain data members and methods that operate on the data members of the class.

- Methods may contain data type declarations and executable statements.

Documentation section	[**Suggested]**
Package statement	[**Optional**]
Import statements	[**Optional**]
Interface statements	[**Optional**]
Class definitions	[**Optional**]

Main Method class
{
Main Method Definition
}
	[**Essential**]

- **Documentation Section: -**
  1. The documentation section comprises a set of Comment lines giving the name of the program, the author and other details, which the programmer would like to refer to at later stage.
  2. Comments must explain why and what of classes & how of algorithms. This would greatly help maintaining the program.
- **Package statement: -**
  1. The first statement allowed in java file is Package statement.
  2. This statement declares a Package name and informs the compiler that the classes defined here belong to this package.

     ```
     package student;
     ```

  3. The package statement optional means our classes do not have to be a part of package.

- **Import Statements: -**
  1. This is similar to the #include statement in C.

     ```
     import student.test;
     ```

  2. This statement instructs the interpreter to load the test classes contained on the package student.
- **Interface statements: -**
  1. An interface is like a class but includes a group of method declarations.
  2. This is also an optional section and is used only when we wish to implement the multiple inheritance features in the program.
- **Class Definition: -**
  1. A java program may contain multiple class definition.
  2. Classes are the primary and essential elements of a Java Program.
  3. The number of classes used depends on the complexity of the program.
- **Main Method Class: -**
  1. Since every java Stand-alone program required a Main method as its starting point, this class is the essential part of a Java program.
  2. A simple java program may contain only this part.
  3. The Main method creates objects of various & establishes communication between them.
  4. **Public: -** Public is an access specifier that declares the main method as unprotected & therefore making it accessible to all other classes.
  5. **Static: -** Static is a keyword which can be called & executed without creating the object, so when we want to call main() method without using an object, we should declare main() method as Static.
  6. **Void: -** A method can return some result. If we want the method to return the result in form of an integer then we should write **int** before the method name.
     - If a method is not meant to return any value, then we should write void before the method's name. **Void** means no value.

7. **Main: -** This is the starting point for interpreter to begin the execution of the program.
8. **String args[]: -** An args[] is the array name & it is of String type. This means that it can be store a group of strings. This array can also store a group of number but in the form string only.

System.out.println ("Hello")

- System is the class name & out is a static variable in system class.
- Out is called a field in system class. (Out is an object of system class).
- When we call this field, a Printstream class object will be created internally, so we can call the print() method.

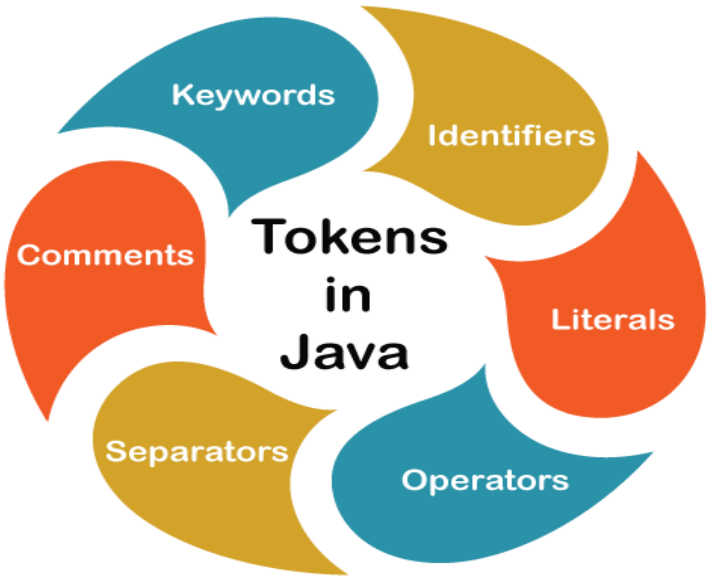
Sample program: -

- Following JAVA program find the area of circle.

```
class area
{
        public static void main(String args[])
        {
                double pi, r, a;
                pi = 3.1416;
                r = 5;
                a = pi * r * r;
                System.out.println ("This is the area of Circle→" +a);
        }
}
```

JAVA tokens: -

- Java program is basically a collection of class.
- A class define by set of declaration statements and methods containing executable statements.
- Most statements contain expression, which describe the action carried out on data.
- Smallest individual unit in a program are known as tokens.
- The compiler recognizes them for building up expression and statement.
- Java program has collection of Tokens.
    - Reserved Keywords
    - Identifiers
    - Literals
    - Operators
    - Separation
    - Comments



Keywords: -

- Keywords are an essential part of a language definition.
- Java language has reserved 50 words as keywords.
- These keywords combine with operators & separators according to syntax, form definition of the java language.
- Every JAVA word is classified as a keyword.
- All keyword have fixed meanings.
- These meaning cannot be changed.
- All keywords are to be written in lower-case letters.

abstract	do	extends	for	import	long
byte	private	short	switch	throws	volatile
class	boolean	catch	continue	double	final
void	instanceof	native	protected	static	while
finally	if	int	new	public	this
try	Break	char	default	enum	float
implements	interface	package	return	super	throw

Identifiers: -

- Identifier refers to the mane of variables, functions and array name, structure name.
- These names are defined by user, so they are known as user-define names.
- Identifier consists of letters and digits.
- Both uppercase and lowercase are permitted.
- **Rules of Identifiers: -**
  1. First character must be an alphabet or underscore.
  2. Must consist of only letters, digits and underscore.
  3. The JAVA variable of any size.
  4. We cannot use keyword as identifiers.
  5. White spaces are not allowed in identifiers.

Literals: -

- Literals in java a sequence of character (digits, letters and other characters) that represent constant values to be stored in variable.
- Constants refer as fixed value in JAVA that does not change during the execution of a program.
- Five major types of literals
  1. Integer literals
  2. Floating point Literals
  3. Character literals
  4. String literals
  5. Boolean literals
- Integer Constants refer to a sequence of digit like 1056.
- Real (Float) constants refer a fractional (point) part like 17.76. Such numbers are called real or floating-point constants.
- A single character constant contains only a single character. They are enclosed within single quote mark. Ex. – 'g', K', 'l' etc.…
- A string constant is sequence of character enclosed within double quote mark. Ex. – "Well Done", "Hello Word", etc.…

Separator: -

- Separators are symbols used to indicate where groups of code are divided and arranged.
- () Parentheses: - Used to enclose parameter in method definition and invocation also used for defining precedence in expression.
- {} Braces: - Used to contain the values of automatically initialize array and to define a block of code classes, methods & local scopes.
- [] Brackets: - Used to declare an array type.
- ; Semicolon: - Used to separate the statements.
- , Comma: - Used to separate variables, in for loop etc.…
- **.** Period: - Used to separate package names, also used to separate variable or method from a reference variable.

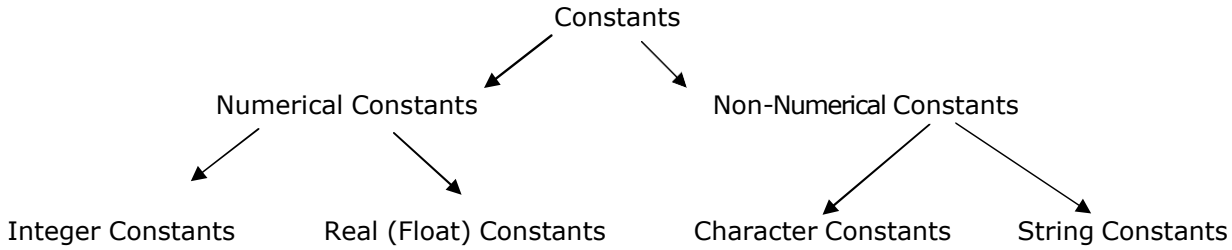
Variables: -

- A variable is a data name that may be used to store a data value.
- Variables may take different values at different time during execution.
- A variable name can be chosen by the programmer.
- Ex.- Average, height, Total, Counter_1, class_strenght etc.….
- **Rules for Variable: -**
  1. They must begin with letter.
  2. The JAVA variable of any size.
  3. Uppercase and lowercase are significant.
  4. It should not be a keyword.
  5. White space is not allowed.
- Some valid example of variable.

  John, Value, T_raise, Delhi, x1, ph_value, mark, sum2, sum_5 etc.….

Constants: -

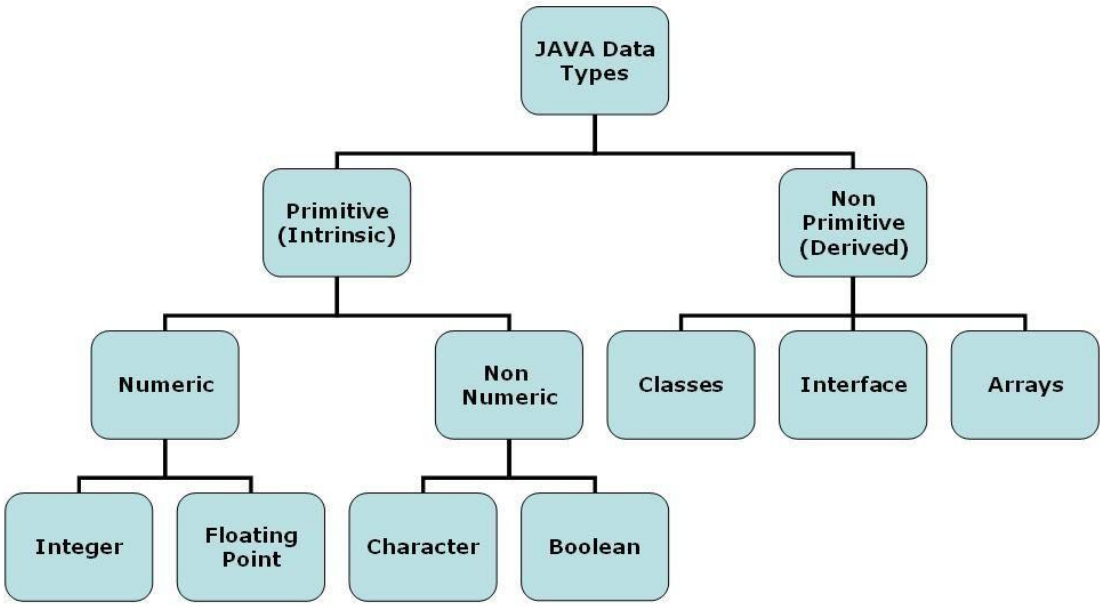
- Constants refer as fixed value in JAVA that does not change during the execution of a program.
- JAVA supports several types of constants.



- Integer Constants refer to a sequence of digit like 1056.
- Real (Float) constants refer a fractional (point) part like 17.76. Such numbers are called real or floating-point constants.
- A single character constant contains only a single character. They are enclosed within single quote mark. Ex. – 'g', K', 'l' etc.…
- A string constant is sequence of character enclosed within double quote mark. Ex. – "Well Done", "Hello Word", etc.…

Data Types: -

- JAVA language is very rich in data types.
- The varieties of data types are available in JAVA.
- JAVA supports two types of data types,
  1. Primitive (Intrinsic) data type
  2. Non primitive (derived) data type.
- All JAVA compilers support five types of primitive data types.
- Integer (int), character (char), floating point (float), double floating point (double), null (void) and Boolean.
- Integers are whole numbers with the range of values and define by keyword int.
- Floating point numbers are defining in JAVA by the keyword float, double and long double.
- A single character can be defined as a char data type.
- The void type has no values. This is usually used to specify the type of function.
- JAVA data types are classified as under.



- Size and range of primitive data types are mentioned as under.

Data type	Size	Minimum	Maximum
byte	1 bytes	-128	127
short	2 bytes	-32,768	32767
int	4 bytes	-2,147,483,648	2,147,483,647
long	8 bytes	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
float	4 bytes	3.4 e—038	1.7 e+38
double	8 bytes	3.4 e—038	1.7 e+308

- **Character Type: -** In order to store character constant in memory, JAVA provides a character data type called **char**. The character type assumes a size of 2 bytes but, it can hold only a single character.

- **Boolean type:** - Boolean type is used when we want to test particular condition during the execution of the program. There are only two values that a Boolean type can take: **true** or **false**.

Scope of variable: -

- An area of the program where the variable is accessible is called Scope.
- JAVA supports three types of variables.
  1. Instance variables.
  2. Class variables.
  3. Local variable.
- **Instance variable: -**
  1. Instance variable declare in class.
  2. Instance variable are created when the object are instantiated and therefore they are associated with objects.
  3. They take different value for each object.
- **Class variable: -**
  1. Class variable are global to a class and belong to entire set of objects that class created.
  2. Only one memory location is created for each class variable.
- **Local variable: -**
  1. Local variable is declare and used in methods.
  2. They are called so because they are not available for use outside the method definition.
  3. Local variable is also declared in program blocks defined between an {open & close braces}.
  4. This variable is visible only from the beginning of its program block to end of the program block.

Type casting: -

- There is a need to store a value of one type in to variable of another type.
- Syntax:

```
type variable1= (type) variable2;
```

- The process of converting one data type to another is called casting.
- Example:

```
int m = 50;
byte n = (byte) m;
long count= (long) m;
```

- Casting is often necessary when a method returns a type different than the one we require.
- Casting in to smaller type may result in a loss of data.
- Like, he float & double can be cast to any other type expect Boolean.
- Casts that Results in NO LOSS OF Information

From	To
byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	Long, float, double
long	Float, double
float	Double

- **Standard Default Value: -**

Type of Variable	Default value
Byte	zero: (byte)0
Short	zero: (short)0
Int	zero: 0
Long	zero : 0l
Float	0.0f
double	0.0d
Char	null char
boolean	False
reference	Null

- **Automatic Conversion: -**
  1. It is possible to assign a value of one type to a variable of different type without a cast.
  2. Java does the conversion of the assigned value automatically.
  3. This is known as automatic type conversion.
  4. Automatic type conversion is possible only if the destination type has enough precession to store the source.

```
byte b = 75;
int a = b;
```

  5. The process of assigning a smaller type to large one is known as winding or Promotion.
  6. Larger to smaller one is known as Narrowing.

Operators: -

- JAVA supports several kinds of operators.
- An operator is a symbol that tell to computer that to perform mathematical or logical operation.
- An operator is used in program to perform the operation on variable or data values.
- JAVA operator can be classified as under.
  1. Arithmetic operator
  2. Relational operator
  3. Logical operator
  4. Assignment operator
  5. Increment and Decrement operator
  6. Conditional operator
  7. Special operator
  8. Bitwise operator

Arithmetic operator: -

- JAVA provides all the basic arithmetic operators.
- The operators +, -, *, /, % all work the same way as they do in other language.
- These can operate on any basic data type.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division

- For example, here **a** and **b** are variable and known as a operand.
- a + b: variable a is added to b
- a - b: variable a is subtract from b
- a * b: variable a is multiply by b
- a / b: variable a is divided by b and integer division truncate the fractional part.
- a % b: variable a is divided by b and produce the reminder of an integer division.
- **Note: - Modulo division (%) operator cannot be used with floating point data.**

Relational operator: -

- JAVA provides all the basic relational operators.
- These operators are used to compare two more values with each other to get relation.
- These kinds of comparisons can be done with the help of relational operators.
- The value of expression is either one or zero.
- The value is one if the relational expression is true.
- The value is zero if the relational expression is false.
- JAVA supports six relational operators.
- These operators and meaning are shown as under.

Relational operator	Meaning
<	Is less then
<=	Is less then or equal to
>	Is greater then
>=	Is greater then or equal to
==	Is equal to
!=	Is not equal to

- For example, here **a** and **b** are integer type of variables.a<b a>b
  a<=b b>=a
  b==a a! =b

- Among the six relational operators, each one is complement of another operator.
  > is complement of <=
  < is complement of >=
  == is complement of !=
- We can simplify an expression the not and the less then operators using complement as shown as:
  !(x<y) x>=y
  !(x>y) x<=y
  !(x!=y) x==y
  !(x<=y) x>y
  !(x<y) x>=y

Logical operator: -

- JAVA has following logical operators.

Logical operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

- The logical operator && and || are used when we want to test more then one condition and take decision. An example is:

  a>b && x == 10
- This kind of expression, which is combination of two or more relational expression, is known as logical expression or compound expression.
- Like the simple relational expression, logical expressions also provide a value one for true expression and zero for false expression.
- Some examples of logical expression.
  1. if (age > 55 && salary < 1000)
  2. if (number < 0 || number >=1000)

Assignment operator: -

- Assignment operators are used to assign the result of an expression.
- Also, JAVA has short hand assignment operators.
- Syntax is:

  variable op = expression, where op is an operator.
- The assignment statements

  V op = exp;

  Is equal to

  V = v op (exp);

  Ex: -

  X + = y+1;

  Is equal to

  X = x+(y+1);
- Some of common shorthand assignment operators are shown as under.

Statement with simple assignment operator	Statement with shorthand operator
a=a+1	a+=1
a=a-1	a-=1
a=a*(n+1)	a*=(n+1)
a=a/(n+1)	a/=(n+1)
a=a%b	a%=b

- The use of shorthand assignment operator has three advantages.
  1. What appears on the left-hand side need not be repeated and therefore it becomes easier to write.
  2. The statement is easier to read.
  3. The statement is more efficient.

Increment and Decrement operator: -

- JAVA has two very useful operators. These two operators not found in other language.
- These operators are Increment (++) and Decrement (--) operators.
- The operator ++ adds 1 to operand.
- The operator – subtracts 1 from operand.

- For example:

  M++; ++M;

  M--; --M;
- We use the increment and decrement operator in for loop, while loop and do while loop.
- There are two types of increment and decrement operators. Prefix and postfix operators.
- A prefix operator first adds 1 to the operand and then result is assign to the variable.
- A postfix operator first assigns the value to the variable and then increments the operand.

Conditional (Ternary) operator: -

- Conditional operator is also known as a ternary operator.
- A conditional operator pair "? "is available in JAVA.
- A conditional operator is used to create conditional expression.
- Syntax is:

  exp1 ? exp2 : exp3;
- In this syntax exp1 is evaluated first. If it is nonzero (true), then the expression exp2 is evaluated and become the value of expression.
- If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.
- Note that only one of the expressions either exp2 or exp3 is evaluated.
- For example:

```
int a=10;
int b=15;
int x;
x = (a<b) ? a : b ;
```

  In this example, x will be assigning the value of b.

Precedence of arithmetic operators: -

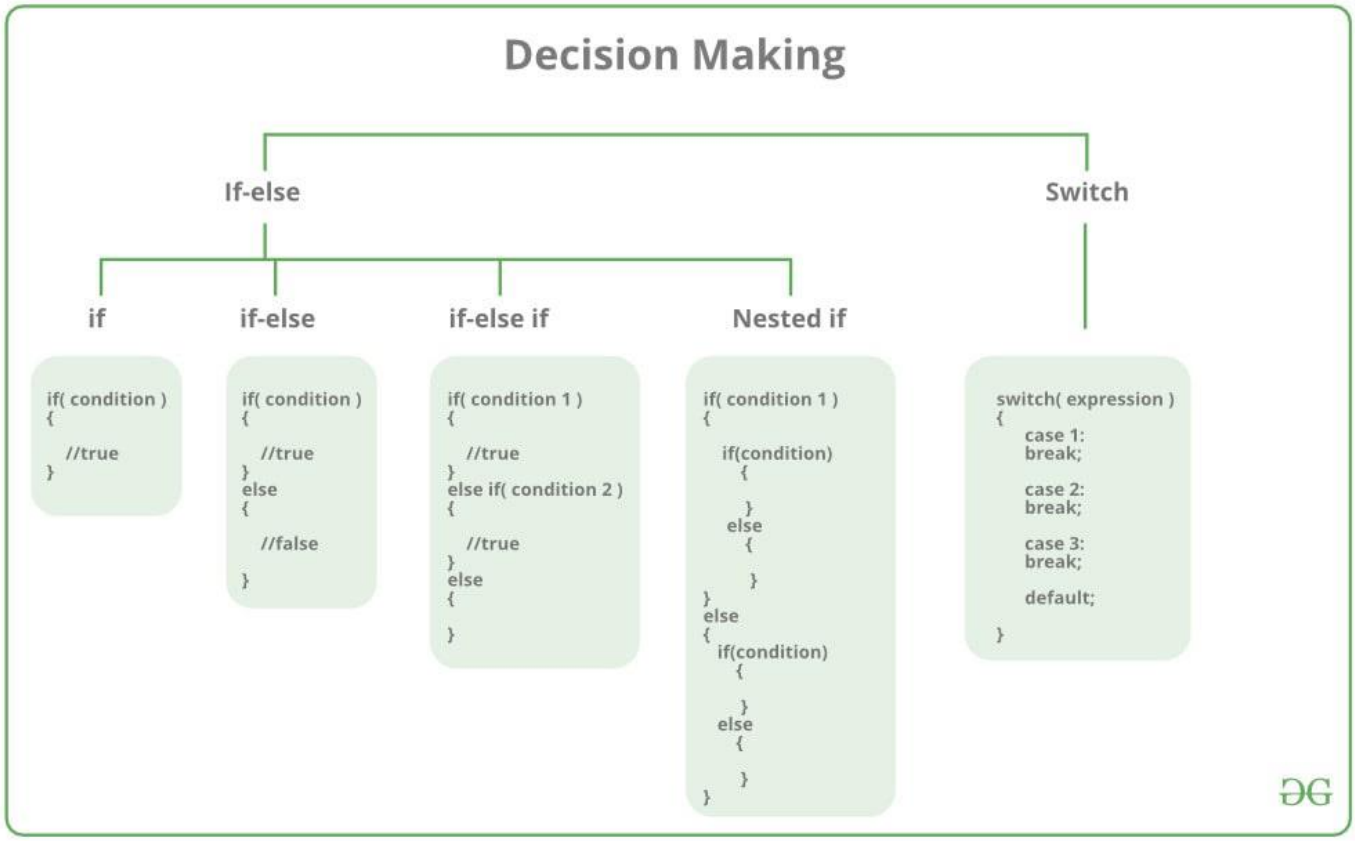
- An arithmetic expression without parentheses will be evaluated from left to right using the rules of precedence of operator.
- There are two levels of arithmetic operators in JAVA.
- High priority: * / %
- Low priority: + -
- Consider the following statement that has been used.

X = a-b/3+c*2-1
When a = 9, b = 12, and c = 3 then the statement become

X=9-12/3+3*2-1

First pass:
Step-1: x=9-4+3*2-1
Step-2: x=9-4+6-1

Second Pass:
Step-3: x=5+6-1
Step-4: x=11-1
Step-5: x=10



Simple IF statement: -

- The if statement is powerful decision-making statement.
- The if statement is used to control the flow of execution of statement.
- It is two-way decision statements and is used to conjunction with expression.
- General form of if statement is:

```
if (test condition)
{
        block-statement;
}
statement-x;
```

- The 'block-statement' may be a single statement or a group of statements.
- If the test condition is true, the block statement will be executed.
- Otherwise the block-statement will be skipped and execution will be jumped to the statement-x.
- Consider the following example.

```
……..

if (category == sports)
{
        marks = marks + bonus marks;
}
System.out.println ("marks =" + marks);
……..
```

The if.... else statement: -

- The if…. else statement is an extension of simple is statement.
- The general form of if…. else statement is as under.

```
if (test condition)
{
        true block statement;
}
else
{
        false block statement;
}
statement-x;
```

- If the test condition is true, then the true block statement is executed.
- If the test condition is false, then the false block statement is executed.
- In either case, either true block or false block will be executed, not both.
- In the both cases, the control is transferred subsequently to statement-x.
- Let us consider an example of counting the number of boys and girls in class.
- We use the code 1 for boys and code 2 for girls.

```
……..
if (code == 1)
{
  boy = boy + 1;
}
else
{

 girl = girl + 1;

}
……..
```

Nesting of if.... else statement: -

- When a series of decision are involved, we may have to use more then one if…. else statement innested from as shown below.

```
if (test condition-1)
{
        if (test condition-2)
        {
                statement-1;
        }
        else
        {
                statement-2;
        }
}
else
{
        statement-3;
}
statement-x;
```

- If the condition-1 is true then test condition-2 is executed.
- If condition-2 is true then statement-1 is executed.
- If condition-2 is false then statement-2 is executed.

- But if the condition-1 is false then control is transfer to the statement-3. In this case statement-1 and statement-2 is not executed.
- In both case control is transfer to the statement-x.
- For example:

```
…….
if(s = 'f')
{
        if(balance >= 5000)
        {
                bonus = 0.05 * balance;
        }
        else
        {
                bonus = 0.02 * balance;
}       }
else
{
         bonus = 0.02 * balance;

 }
 balance = balance + bonus;
 ……
```

The else if ladder: -

- There is another way of putting ifs together when multi path decisions are involved.
- A multi path decision is a chain of ifs in which the statement associated with each else is an if.
- General form of else if ladder is:

```
if (condition-1)
        statement-1;
else if (condition-2)
        statement-2;
        else if (condition-3)
                statement-3;
                else if (condition-n)
                        statement-n
                        else
                        default-statement;
statement-x;
```

- This construct is known as the else if ladder.
- The conditions are evaluated from the top (of ladder) downwards.
- As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x.
- When all the conditions become false, then the final else containing the default-statement will be executed.
- Let us consider the following example.

```
……
if (code == 1)
        color = "red";
else if (code == 2)
        color = "green";
        else if (code =3)
                color = "white";
             else
                color = "yellow";
……..
```

The switch statement: -

- The switch statement is another decision making and branching statement.
- We have seen that when one of many alternatives is to be selected, we can use an if statement to control the selection.

When the numbers of alternatives are increased at that time, we have to use switch statement.

- JAVA has built-in multi way decision statement known as a switch.
- The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statement associated with that case is executed.
- The general form of the switch statement is shown below:

```
switch (expression)
{
        case value-1:
                statement-1;
                break;

        case value-2:
                statement-2;
                break;

        case value-3:
                statement-3;
                break;

        ……
        ……..
        default:
                default statement;
                break;
}
statement-x;
```

- The switch expression must be either integer or character type.
- Case labels must be integer or character.
- Case labels must be unique. No two labels can have the same values.
- Case labels must end with semicolon.
- The break statement transfers the control out of the switch statement.
- The break statement is optional. That is, two or more case labels may belong to the same statement.
- The default label is optional. If present, it will be executed when the expression does not find a matching case label.
- There can be at most one default labels.
- The default may be placed anywhere but usually placed at the end.
- It is permitted to nest switch statements.

The ? : operator: -

- Conditional operator is also known as a ternary operator.
- A conditional operator pair " ? : " is available in C.
- A conditional operator is used to create conditional expression.
- Syntax is:

  exp1 ? exp2 : exp3;
- In this syntax exp1 is evaluated first. If it is nonzero (true), then the expression exp2 is evaluated and become the value of expression.
- If exp1 is false, exp3 is evaluated and its value becomes the value of the expression.
- Note that only one of the expressions either exp2 or exp3 is evaluated.
- For example:

```
int a=10;
int b=15;
int x;

x = (a<b) ? a : b ;
```

- In this example, x will be assigning the value of b.

The while statement: -

- The simplest of all the looping structure in java is the while loop.
- The basic format of the while statement is:

```
Initialization;
while (test condition)
{
        body of the loop;
}
statement-x;
```

- The while is an entry-controlled loop statement.
- The test-condition is evaluated and if the condition is true, then the body of the loop is executed.
- After execution of the body, the test-condition is once again evaluated and if it is true, the body is executed once again.
- The process of repeated execution of the body continues until the test-condition finally becomes false and the control is transferred out of the loop.
- On exit, the program continues with the statements immediately after the body of the loop.
- Let us consider the following example.

```
sum =0;
n=1;
while(n<=10)
{
        sum = sum + n * n;
        n = n + 1;
}
System.out.println("Sum= " + sum);
……..
```

The do...while statement: -

- On some time, it is required to execute the body of the loop before the test condition is checked.
- This kind of situation can be handle with help of the do while statement.
- The do while statement is also known as exit control or post tested loop.
- General form of do while statement is:

```
Initialization;
do
{
        body of the loop;
}
while (test condition);
statement-x;
```

- On reaching the do statement, the program proceeds to evaluate the body of the loop first.
- At the end of loop, the test condition in the while statement is executed.
- If the condition is true then control is further transfer to body of the loop.
- But if the condition is false control is transfer to statement-x.
- The do while loop create an exit control loop and therefore body of the loop is always executed at least once.

- A simple example of do while loop is:

```
number = 0;
do
{       System.out.println ("input a
        number");number = number + 1;
} while (number>0 && number < 100);
```

The for statement: -

- The for loop is another entry control or pre tested loop.
- The general form of for loop is:

```
for (initialization; test condition; increment or decrement)
{
        body of the loop;
}
statement-x;
```

- There are three part in a for loop.
- Initialization, test condition and increment or decrement.
- Initialization done first. Initialization portion is executed only one time.
- Then control is transfer to test condition. If the condition becomes true then control is transfer to body of the loop.
- After the execution of body of the loop control is goes to increment or decrement portion.
- After the execution of increment or decrement part control is goes to test condition.
- If condition becomes false then control is transfer to statement-x.
- Consider the following example:

```
……….
for (x=0; x<=10; x++)
{
        System.out.println ("x =" + x);
}
System.out.println ("\n");
…….
```

Nesting of for loop: -

- Nesting of for loops, that is, one for statement within another for statement.
- For example consider the following:

```
for (row = 1; row <= romax; row++)
{
        for (column =1; column <= colmax; ++column)
        {
                y = row * column;
                System.out.println (+y);
        }
        System.out.println ("\n");
}
```

The nesting may continue up to any desired level.

While	do…while
It will execute only if the test condition is true.	It will execute at least once even if the test condition is false.
It is an entry-controlled loop.	It is an exit-controlled loop.
It is generally used for implementing common looping situations.	It is basically used where the number of statements required to be executed at least once.

Jumping in loops: -

Break:

- When the break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop.
- When the loops are nested, the break would only exit from the loop containing it.
- That is, the break will exit only a single loop.

```
While (….)
{
        …..
        …… If(condition)
                break;
        …..
}
…..
```

```
do (….)
{
        …..
        …… If(condition)
                break;
        …..
} while (….);
…..
```

```
for (….)
{
        …..
        …… If(error)
                break;
        …..
}
…..
```

```
for (….)
{
        …..
        for (….)
        {
                If(condition)
                break;
        }
        …..
}
```

Continue :

- Like the break statement, Java supports another similar statement called the continue statement.
- The break which causes the loop to be terminated, the continue, as the name implies causes the loop to be continued with the next iteration after skipping any statement in between.
- The continue statement tells the compiler, "SKIP THE FOLLOWING STATEMENTS AND CONTINUE WITH THE NEXT ITERATION".
- The format of the continue statement is:

Continue;

- In a while and do loops, continue causes the control to go directly to the test condition and then to continue the iteration process.
- In the case of for loop, the increment section of the loop is executed before the test condition is checked.

```
while (test condition)
{
        …..
        if (….)
                continue;
        …..
        …..
}
```

```
do
{
        …..
        if (….)
                continue;
        …..
        …..
} while (test condition);
```

```
for (initialization; test condition; increment)
{
        …..
        if (….)
                continue;
        …..
        …..
}
```

Arrays: -

- Definition: - "Array is sequence collection of related data item that share same name and same
- data type, but store a different value."
- C supports a derived data type known as array that can be used for many applications.
- Three type of array is available in C :
  1. One dimensional array.
  2. Two dimensional arrays.
  3. Multidimensional arrays.

One dimensional array: -

- Definition: - "Array is sequence collection of related data item that share same name and same
- data type, but store a different value."
- A list of items can be given one variable name using only one subscript and such a variable is
- called a one–dimensional array.
- Like any other variable, arrays must be declared before they are used.
- The general form of array declaration is:

```
data-type variable-name [ ] = new data-type [ size ];
```

- The data-type specifies the data type of array.
- Variable is a name of variable which is declared by user.
- The new is dynamic memory allocation operator which provides the memory.
- Size indicates the maximum number of elements that can be stored inside the array.

- For example, if we want to represent a set of five number, say (35,40,20,57,19), by an array variable number, then we may declare the variable number as follows:

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

- The computer provides five storage locations as shown below:

	number[0]
	number[1]
	number[2]
	number[3]
	number[4]

- The values to the array elements can be assigned as follows:

```
number[0] = 35;
number[1] = 40;
number[2] = 20;
number[3] = 57;
number[4] = 19;
```

- This would cause the array number to store the values as shown below:

35	number[0]
40	number[1]
20	number[2]
57	number[3]
19	number[4]

- These elements may be used in programs just like any other C variable.
- For example, the following are valid statements:

```
a = number[0] + 10;
number[4] = number[0] + number [2];
number[2] = x[8] + y [10];
```

- When the compiler sees the character array sting, it terminates it with additional null character.
- The null character of the character is determined by '\0'.
- When declaring character array, we must allow one extra element space for the null terminator.

Initialisation of array: -

- Initialisation means to store the values to an array element.
- An array can be initialized at two ways.
  1. At compile time.
  2. At run time.

Compile time initialisation: -

- We can initialisation the elements of arrays in the same way as the simple variables when they are declared.
- The general form of initialisation of arrays is:

```
data-type array-name[size] = {list of values};
```

- The values in the list are separated by commas.
- For example int num[3] = {10, 23, 43};
- If the number of elements are greater then number of assigned value at that time only that elements will be initialised. The remaining elements will be initialised with zero.
  For example int num[5] = {2, 4, 5};

2	num[0]
4	num[1]
5	num[2]
0	num[3]
0	num[4]

- In some case the size of array will be omitted.
- In such case, the compiler allocates enough space for all initialized elements.
  For example int num[ ] = {10, 49, 54, 43};
- In this statement the size of array will be four.
- Character array may be initialized in same way.
  For example char name[ ] = {'d', 'o', 'c', 't', 'o', 'r'};

```
char city[ ] = "idar";
```

Run time initialisation: -

- An array can be initialised at run time.
- Run time initialisation is used with large array.
- Consider the following example.

```
int i;
int num[50];
for( i = 0; i<=49; i++)
{
scanf("%d", & num[i]);
}
```

The 50 elements are initialised with values which are provided by user from keyboard at runtime.