

Unit 2 : Tokens, Expression and Control Structures

Er. Shankar pd. Dahal
pdsdahal@gmail.com

Data Type:

- Data types specify the different sizes and values that can be stored in the variable.

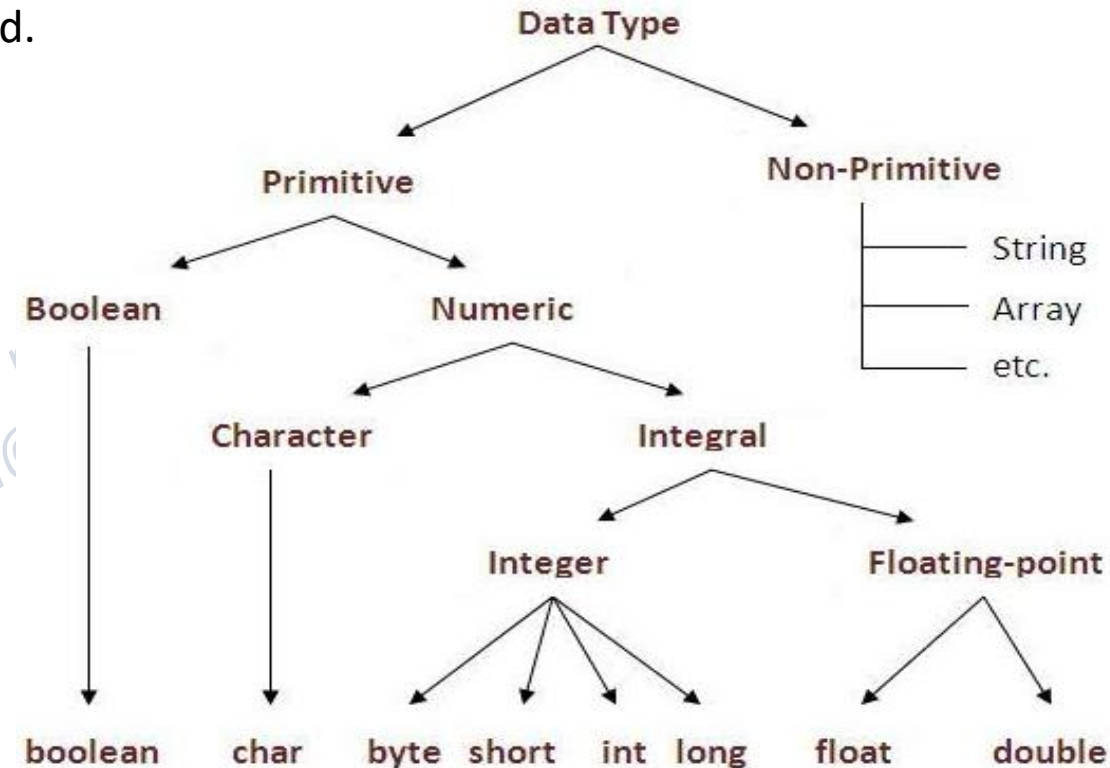
Types of data types in Java:

1. Primitive data types:

- Primitive data types are a set of basic data types from which all other data types are constructed.
- Primitive data types in Java are built-in data types that are predefined.
- A primitive type has always a value.
- A primitive type starts with a lowercase letter.

2. Non-primitive data types:

- Non-primitive data types are also called 'reference variables' or 'object references' as they reference a memory location where data is stored.
- Non-primitive types are created by the programmer and is not defined by Java (except for String).
- Non-primitive types can be null.
- Non-primitive types starts with an uppercase letter.



- A primitive data type specifies the size and type of variable values, and it has no additional methods.
- There are eight primitive data types in Java.
- These can be put in four groups:

1. **Integers** : This group includes **byte**, **short**, **int**, and **long**, which are for whole-valued signed numbers.
2. **Floating-point Numbers** : This group includes **float** and **double**, which represent numbers with fractional precision.
3. **Characters** : This group includes **char**, which represents symbols in a character set, like letters and numbers.
4. **Boolean** : This group includes **boolean**, which is a special type for representing true/false values.

Data Type	Size
byte	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
boolean	1 bit
char	2 bytes

User defined data types :

- Non-Primitive data types in Java are user-defined data types that can be created and modified by the users.
- There are five types of Non-Primitive data types in Java :

- ☐ Class
- ☐ Object
- ☐ String
- ☐ Array
- ☐ Interface.

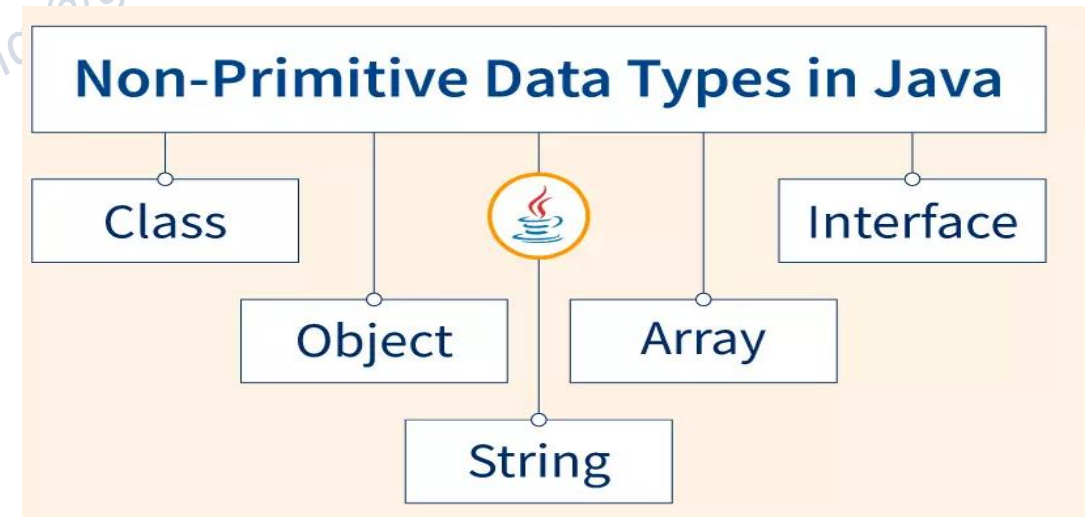
- They are used to store multiple values of either the same data type or different data types.

1. Class :

- Class is a user-defined data type that is used to create objects.
- A class contains a set of properties and methods that are common and exhibited by all the objects of the class.

2. Object :

- An object is a variable of a class by which we can access the member functions and variables of that class.



An object consists of :

State : It shows the properties of an object and is represented by the attributes of the object.

Identity : It enables one object to interact with other objects and also provides a unique name to the object.

Behaviour : It showcases the response of an object to other objects and is represented by the methods of the object.

3. String :

- Strings in Java are designed in such a way that they can hold a sequence of characters in a single variable.

4. Array :

- Arrays are non-primitive data types in Java that are used to store elements of the same data type in a contiguous manner.
- They are not pre-defined and users have to declare and initialize the arrays by themselves.

5. Interface:

- The interface is similar to a class and can include both functional methods and the variables but the only difference is that the methods declared inside the interface are by default abstract i.e. they don't have a body and just the method declaration is provided.
- The interface is also known as a **fully abstract class**.

Java Constant :

- **Constant** is an entity in programming that is immutable. In other words, the value that cannot be changed.
- Java doesn't have built-in support for constants.
- To define the constants in Java by using the non-access modifiers static and final.

Static Modifier:

- The purpose to use the static modifier is to manage the memory.
- It also allows the variable to be available without loading any instance of the class in which it is defined.

Final Modifier:

- The final modifier represents that the value of the variable cannot be changed.
- It also makes the primitive data type immutable or unchangeable.

Syntax:

static final datatype identifier_name=value;

Identifiers in Java :

- Identifiers in Java are symbolic names used for identification.
- They can be a class name, variable name, method name, package name, constant name, and more.
- There are some reserved words that can not be used as an identifier.
- For every identifier there are some conventions that should be used before declaring them.

General Rules:

- A valid identifier must have characters [A-Z] or [a-z] or numbers [0-9], and underscore(_) or a dollar sign (\$). for example, **@texas** is not a valid identifier because it contains a special character which is @.
- There should not be any space in an identifier. For example, **java texas** is an invalid identifier.
- An identifier should not contain a number at the starting. For example, **123texas** is an invalid identifier.
- An identifier should be of length 4-15 letters only. However, there is no limit on its length. But, it is good to follow the standard conventions.
- We can't use the Java reserved keywords as an identifier such as int, float, double, char, etc. For example, **int double** is an invalid identifier in Java.
- An identifier should not be any query language keywords such as SELECT, FROM, COUNT, DELETE, etc.

Naming Convention :

- ❖ Java naming convention is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method, etc.

Advantage of Naming Conventions in Java

- ❖ By using standard Java naming conventions, you make your code easier to read for yourself and other programmers.
- ❖ Readability of Java program is very important. It indicates that less time is spent to figure out what the code does.

Type	Naming Rules	Examples
Class	<ul style="list-style-type: none">➤ It should start with the uppercase letter.➤ It should be a noun such as Color, Button, System, Thread, etc.➤ Use appropriate words, instead of acronyms.	public class Employee { //code snippet }
Interface	<ul style="list-style-type: none">➤ It should start with the uppercase letter.➤ It should be an adjective such as Runnable, Remote, ActionListener.➤ Use appropriate words, instead of acronyms.	interface Printable { //code snippet }
Method	<ul style="list-style-type: none">➤ It should start with lowercase letter.➤ It should be a verb such as main(), print(), println().➤ If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().	void draw() { //code snippet }

Variable	<ul style="list-style-type: none"> ➤ It should start with a lowercase letter such as id, name. ➤ It should not start with the special characters like & (ampersand), \$ (dollar), _ (underscore). ➤ If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName. ➤ Avoid using one-character variables such as x, y, z. 	int id ;
Package	<ul style="list-style-type: none"> ➤ It should be a lowercase letter such as java, lang. ➤ If the name contains multiple words, it should be separated by dots (.) such as java.util, java.lang. 	package com.texas ;
Constant	<ul style="list-style-type: none"> ➤ It should be in uppercase letters such as RED, YELLOW. ➤ If the name contains multiple words, it should be separated by an underscore(_) such as MAX_PRIORITY. ➤ It may contain digits but not as the first letter. 	static final int MIN_AGE = 18;

Java follows camel-case syntax for naming the class, interface, method, and variable.

If the name is combined with two words, the second word will start with uppercase letter always such as actionPerformed(), firstName.

Literal :

- Literal in Java is a synthetic representation of boolean, numeric, character, or string data.
- It is a means of expressing particular values in the program, such as an integer variable named count is assigned an integer value in the following statement.

```
int count = 0;
```

A literal '0' represents the value zero.

- Thus, a constant value assigned to the variable can be referred to as literal.

Literals in Java can be classified into six types, as below:

1. Integral Literals
2. Floating-point Literals
3. Char Literals
4. String Literals
5. Boolean Literals
6. Null Literals

Er. Shankar pd. Dahal
pdsdahal@gmail.com

Variables :

- A variable is a container which holds the value while the Java program is executed.
- A variable is assigned with a data type.
- It is a name of the memory location. It is a combination of "vary + able" which means its value can be changed.

There are three types of variables in Java:

1. local variable
2. instance variable
3. static variable

Er. Shankar pd. Dahal
pdsdahal@gmail.com

Variables Types :

1. Local :

- Local variables are declared in methods, constructors, or blocks.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are created when the method, constructor or block is entered, and the variable will be destroyed once it exits the method, constructor, or block.
- Access modifiers cannot be used for local variables.
- There is no default value for local variables, so local variables should be declared, and an initial value should be assigned before the first use.

2. Instance :

- Instance variables are declared in a class, but outside a method, constructor or any block.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.
- Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.

3. Static:

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- Static variables are created when the program starts and destroyed when the program stops.
- Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name *ClassName.VariableName*.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.

Er. Shankar pd. Dahal
pdsdahal@gmail.com

Type Casting :

- **Type Casting** is a method or process that converts a data type into another data type in both ways manually and automatically.
- The automatic conversion is done by the compiler and manual conversion performed by the programmer.

Types :

1. Widening Type Casting :

- Converting a lower data type into a higher one is called **widening** type casting.
- It is also known as **implicit conversion** or **casting down**.
- It is done automatically. It is safe because there is no chance to lose data.

2. Narrowing Type Casting :

- Converting a higher data type into a lower one is called **narrowing** type casting.
- It is also known as **explicit conversion** or **casting up**.
- It is done manually by the programmer. If we do not perform casting, then the compiler reports a compile-time error.

Data Type	Size
byte	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
boolean	1 bit
char	2 bytes

Widening :

byte -> short -> char -> int -> long -> float -> double

Narrowing :

double -> float -> long -> int -> char -> short -> byte

Er. Shankar Pdsdahal
pdsdahal@gmail.com

Modifiers Types :

❖ **Access modifiers** in Java helps to restrict the scope of a class, constructor, variable, method, or data member.

1. Java Access Modifiers

2. Non-Access Modifiers

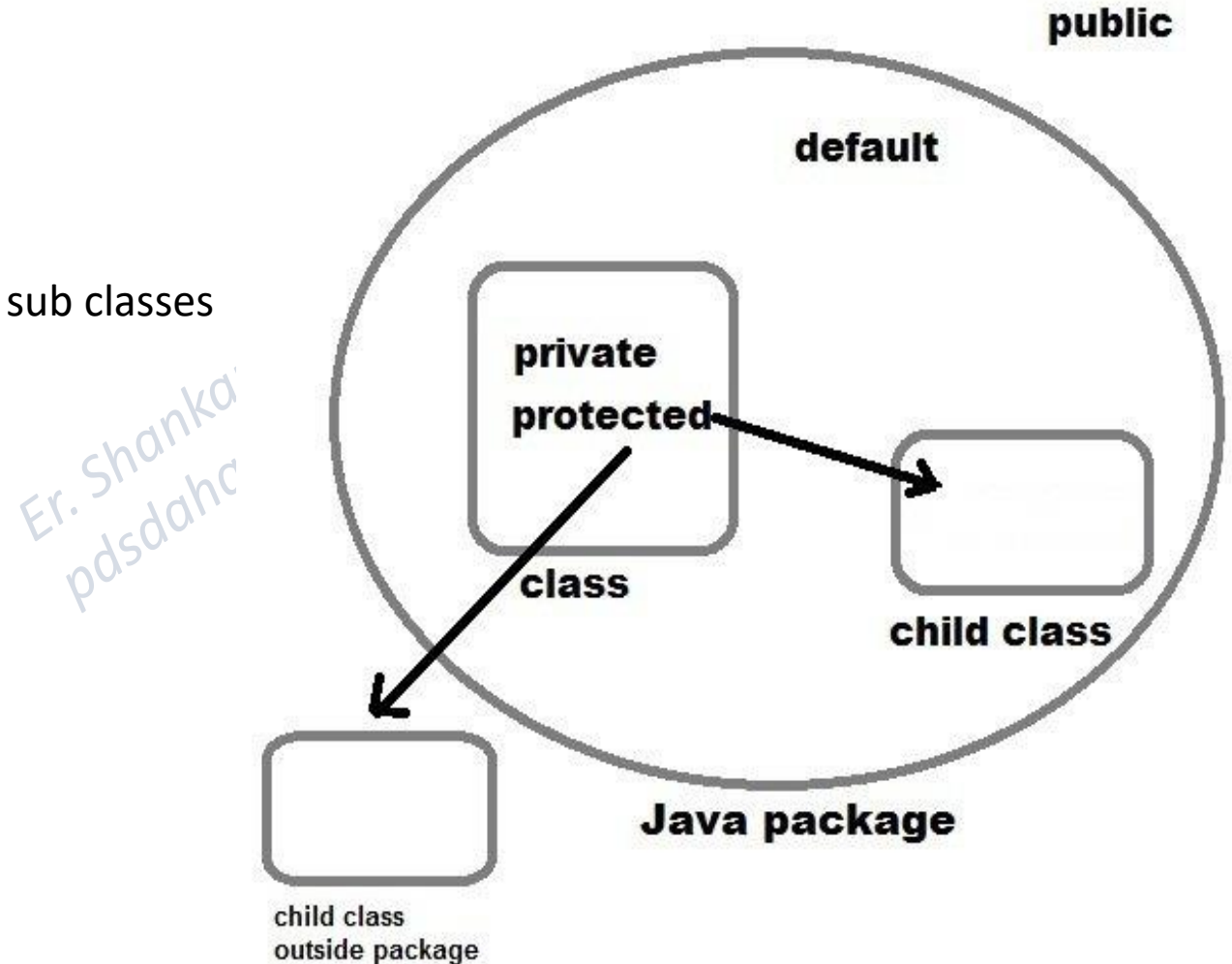
1. Java Access Modifiers - ***controls the access level***

Default: Default has scope only inside the same package

Public: Public has scope that is visible everywhere

Protected: Protected has scope within the package and all sub classes

Private: Private has scope only within the classes



2. Non-Access Modifiers - ***do not control access level, but provides other functionality***

a. **final** :

- ❖ Attributes and methods cannot be overridden/modified.
- ❖ The class cannot be inherited by other classes.

b. **static** :

- ❖ Attributes and methods belongs to the class, rather than an object.
- ❖ A static method means that it can be accessed without creating an object of the class.

c. **abstract** :

- ❖ An abstract method belongs to an abstract class, and it does not have a body. The body is provided by the subclass.
- ❖ An abstract class can never be instantiated. If a class is declared as abstract, then the sole purpose is for the class to be extended.
- ❖ If a class contains abstract methods, then the class should be declared abstract.

Er. Shankar p.d.
pdsdahal@gmail.com

Operators in Java and its Types :

❖ Operators are used to perform operations on variables and values.

Types :

1. Arithmetic Operators
2. Assignment Operators
3. Logical Operators
4. Relational Operators
5. Unary Operators
6. Bitwise Operators
7. Ternary Operators
8. Shift Operators

1. Arithmetic Operators :

❖ Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Operator	Result
+	Addition
−	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition assignment
−=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment
—	Decrement

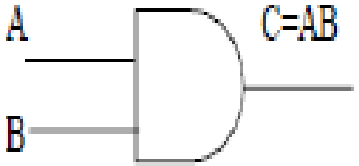
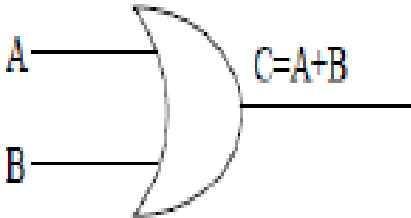
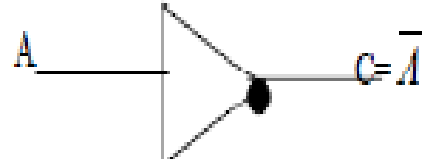
2. Assignment Operators:

❖ An *Assignment Operator* is an *operator* used to *assign* a new value to a variable.

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C – A
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator.	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator.	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
=	bitwise inclusive OR and assignment operator.	C = 2 is same as C = C 2

3. Logical Operators:

❖ These operators are used to perform “logical AND” , “logical OR” and “logical NOT” operations, i.e., the function similar to AND gate and OR gate in digital electronics.

GATE	SYMBOL	INPUTS		OUTPUT
		A	B	C
AND IC 7408		0	0	0
		0	1	0
		1	0	0
		1	1	1
OR IC 7432		0	0	0
		0	1	1
		1	0	1
		1	1	1
NOT IC 7404		1	-	0
		0	-	1

Operator	Description	Example
&& (and)	True if both the operands is true	a<10 && a<20
(or)	True if either of the operands is true	a<10 a<20
! (not)	True if an operand is false (complements the operand)	!(x<10 && a<20)

Er. Shani
pdsdahal@y.

4. Relational Operators:

❖ These operators are used to check for relations like equality, greater than, less than. They return Boolean results after the comparison.

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(A == B) is not true
!=	If the values of two operands are not equal, then condition becomes true.	(A != B) is true
>	If the value of the left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true
<	If the value of the left operand is less than the value of right operand, then condition becomes true.	(a < b) is true
>=	If the value of the left operand is greater than or equal to the value of the right operand, then condition becomes true.	(a >= b) is not true
<=	If the value of the left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true

Assume A = 10 and B = 20.

kar pd. Dahal
dahal@gmail.com

5. Unary Operators :

❖ Unary operators require only one operand to perform various operations such as increment/ decrement of the value by one.

Syntax	Operator type	Operation
expr++	Postfix	Increases the value by 1 after the value is assigned
expr--	Postfix	Decreases the value by 1 after the value is assigned
++expr	Prefix	Increases the value by 1 before the value is assigned
--expr	Prefix	Decreases the value by 1 before the value is assigned
+expr	Prefix	Displays expression in positive
-expr	Prefix	Displays expression in negative
!expr	Prefix	Inverts the value of expression

6. Bitwise Operators:

❖ These operators perform **bit by bit** operation. The output is either 1 or 0 based on the comparison.

Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
=	Bitwise OR assignment
^=	Bitwise exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left assignment

A	B	A B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

The bitwise logical operators are &, |, ^, and ~.

Er. Shankar pd. Dahal
pdsdahal@gmail.com

7. Ternary Operators:

❖ Ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name ternary.

Syntax :

(Condition) ? (Statement1) : (Statement2);

8. Shift Operators:

These operators are used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively. They can be used when we have to multiply or divide a number by two.

Format:

number **shift_op** number_of_places_to_shift;

1. <<, Left shift operator: shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as of multiplying the number with some power of two.

2. >>, Signed Right shift operator: shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of the initial number. Similar effect as of dividing the number with some power of two.

3. >>>, Unsigned Right shift operator: shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0.

Packages In Java

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.

Packages are used for:

- ❖ Preventing naming conflicts.

For example, there can be two classes with name Employee in two packages,
college.staff.cse.Employee and
college.staff.ee.Employee

- ❖ Making searching/locating and usage of classes, interfaces, enumerations and annotations easier.

- ❖ Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.

- ❖ Packages can be considered as data encapsulation (or data-hiding).

Java Control Statements :

- ❖ Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements.

Three types of control flow statements.

1. Decision Making statements (Conditional statements)

- ❖ if statements
- ❖ switch statement

2. Loop statements

- ❖ do while loop
- ❖ while loop
- ❖ for loop
- ❖ for-each loop

3. Jump statements

- ❖ break statement
- ❖ continue statement
- ❖ return statement

Er. Shankar pd. Dahal
pdsdahal@gmail.com

1. Decision-Making statements / Conditional statements

❖ Decision-making statements decide which statement to execute and when.

I. If Statement:

- ❖ In Java, the "if" statement is used to evaluate a condition.
- ❖ The condition of the If statement gives a Boolean value, either true or false.

Types :

- Simple if statement
- if-else statement
- if-else-if ladder
- Nested if-statement

a. Simple if Statement :

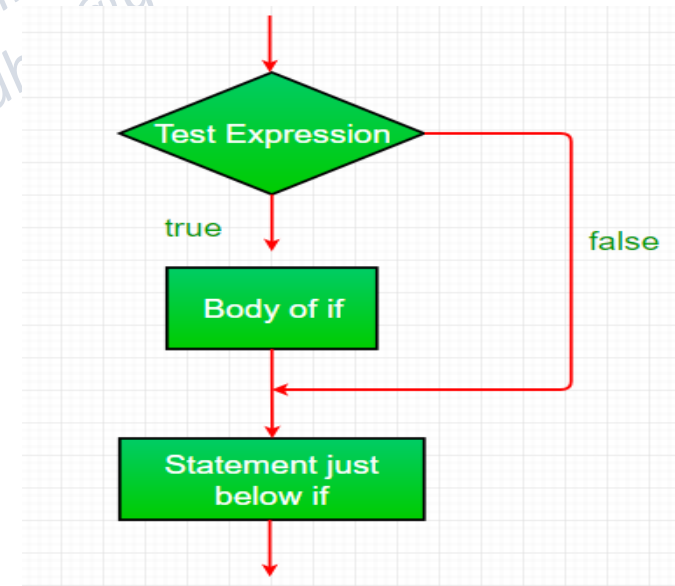
- ❖ It is used to decide whether a certain statement or block of statements will be executed or not i.e., if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if(condition) {
```

```
// Statements to execute if  
// condition is true
```

```
}
```



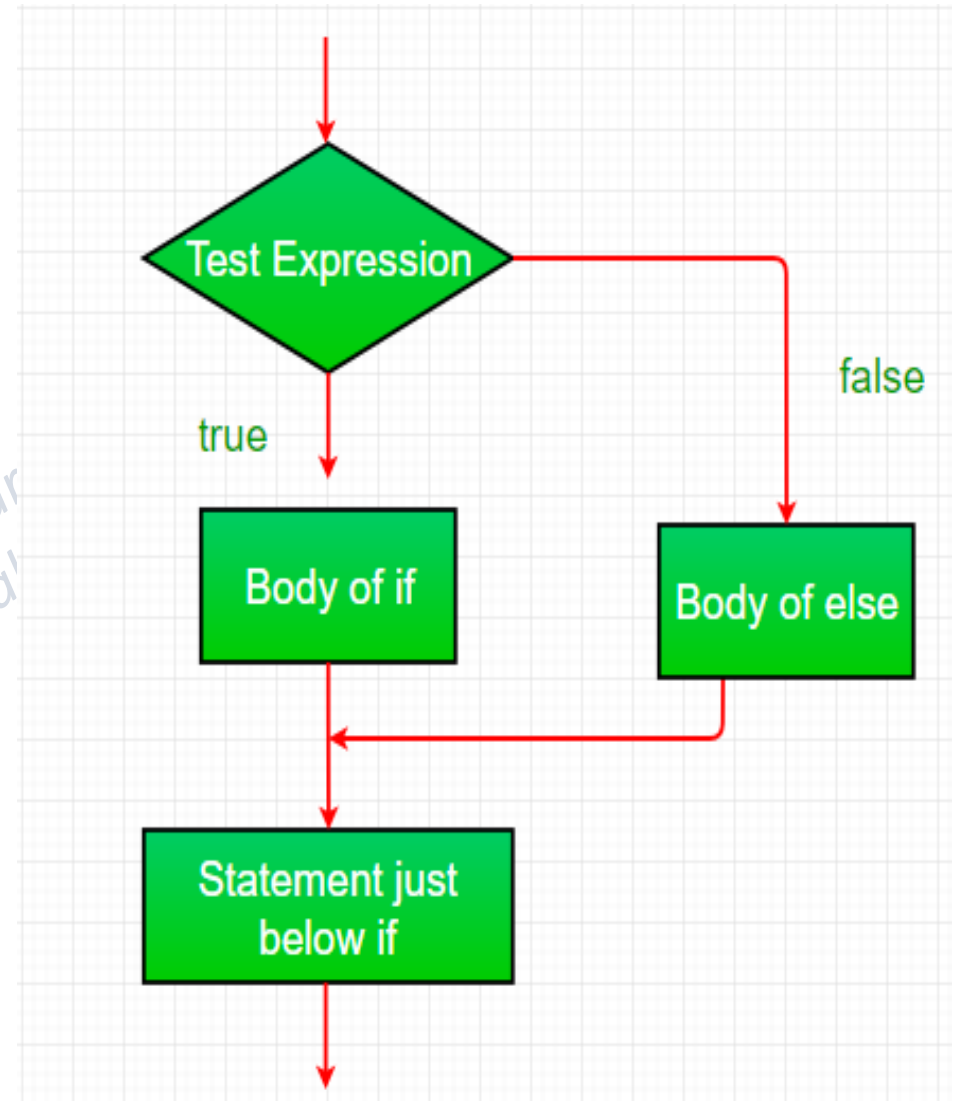
b. if-else statement :

- ❖ The if-else statement is an extension to the if-statement, which uses another block of code, i.e., else block. The else block is executed if the condition of the if-block is evaluated as false.

Syntax:

```
if(condition) {  
    statement 1; //executes when condition is true  
}  
else{  
    statement 2; //executes when condition is false  
}
```

Er. Shankar
pdsdaha'

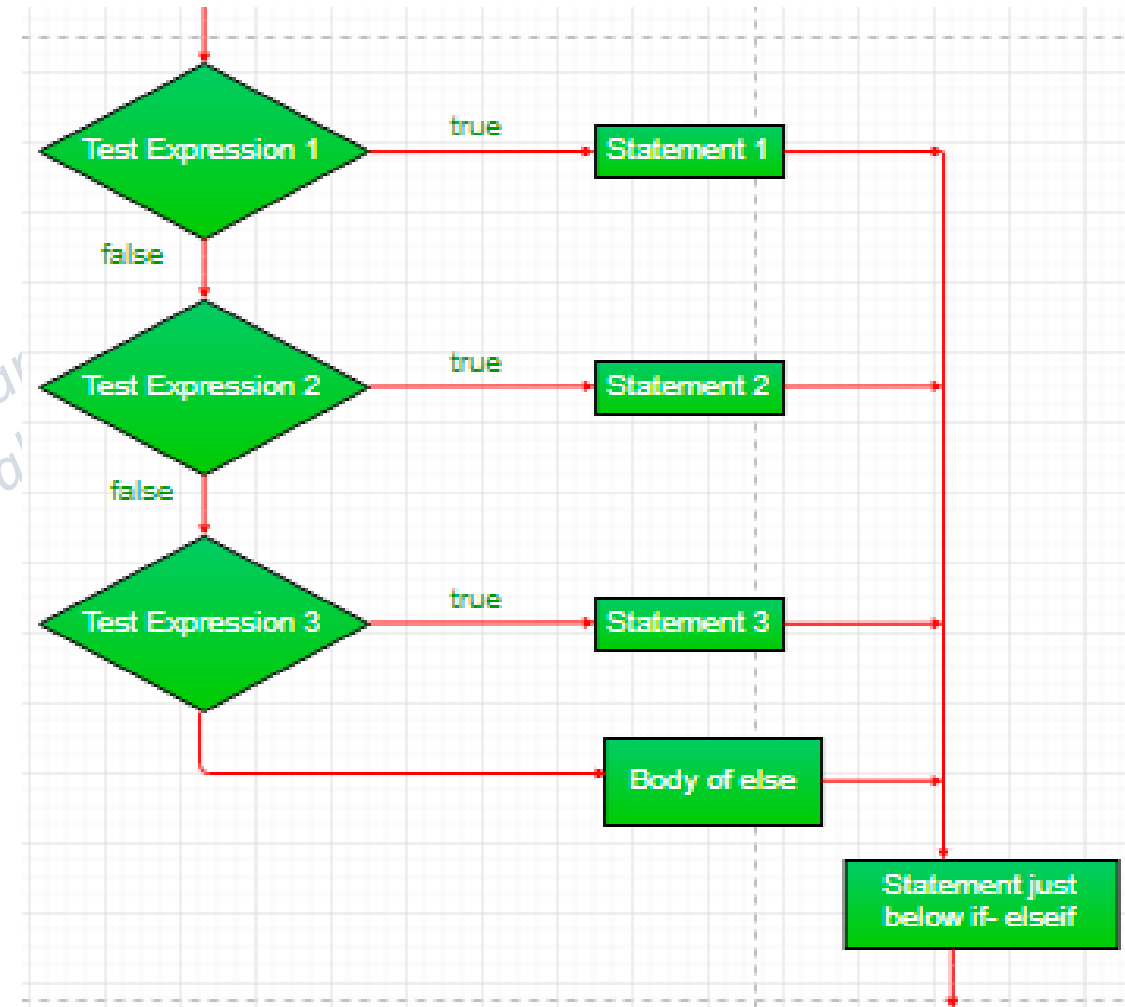


c. if-else-if ladder

- ❖ The if-else-if statement contains the if-statement followed by multiple else-if statements.
- ❖ The if statements are executed from the top down.
- ❖ As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed.
- ❖ If none of the conditions is true, then the final else statement will be executed.

Syntax :

```
if(condition 1) {  
    statement 1; //executes when condition 1 is true  
}  
else if(condition 2) {  
    statement 2; //executes when condition 2 is true  
}  
else {  
    statement n; //executes when all the conditions are false  
}
```

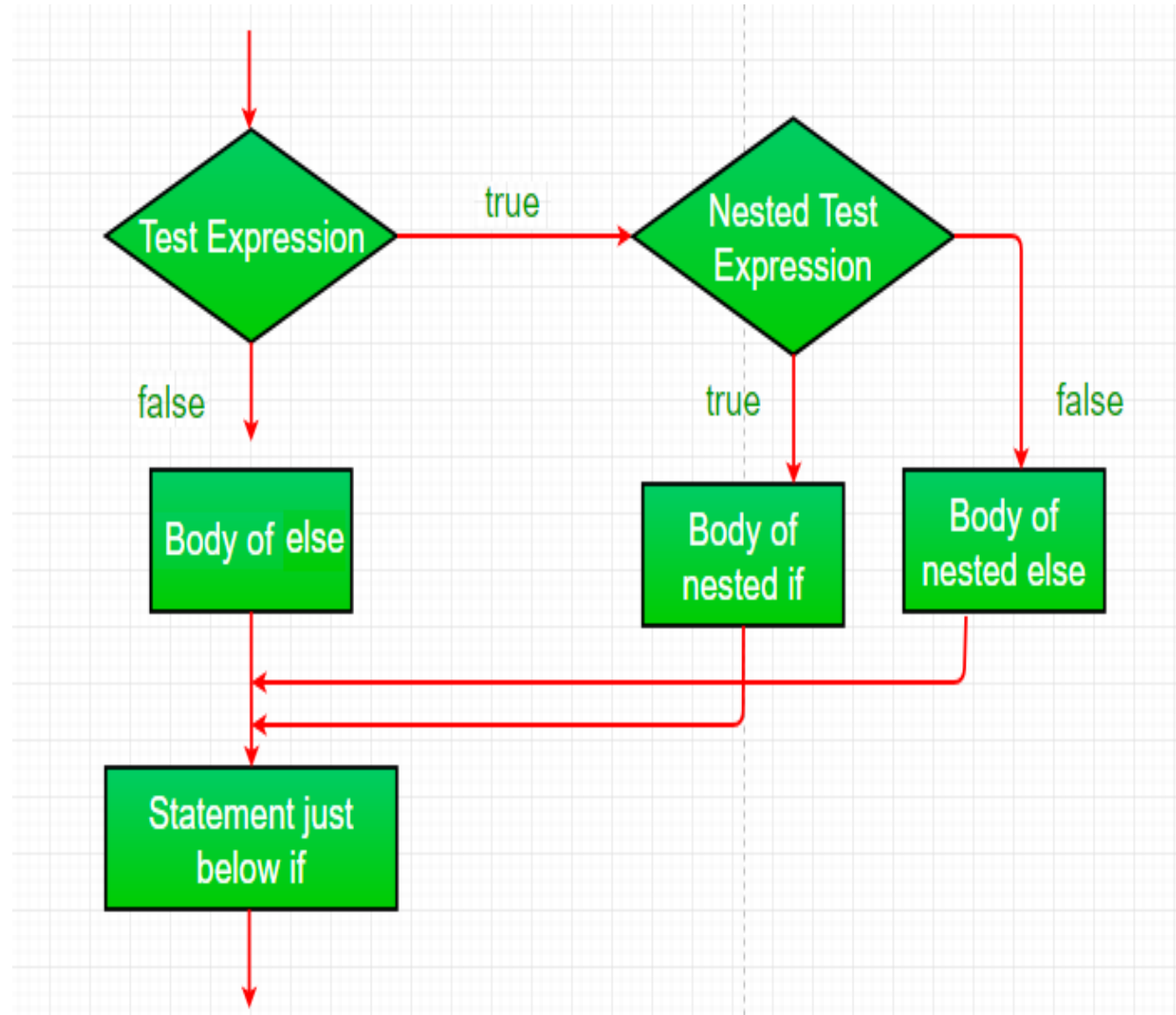


d. Nested if-statement

❖ In nested if-statements, the if statement can contain a **if** or **if-else** statement inside another if or else-if statement.

Syntax :

```
if(condition 1) {  
    statement 1; //executes when condition 1 is true  
    if(condition 2) {  
        statement 2; //executes when condition 2 is true  
    }  
    else{  
        statement 3; //executes when condition 2 is false  
    }  
}
```



II. switch statement :

- ❖ In Java, Switch Statement are similar to if-else-if statements.
- ❖ The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched.
- ❖ The switch statement is easier to use instead of if-else-if statements.
- ❖ It also enhances the readability of the program.

Details about the switch statement:

- ❖ The case variables can be int, short, byte, char, or enumeration.
- ❖ String type is also supported since version 7 of Java
- ❖ Cases cannot be duplicate
- ❖ Default statement is executed when any of the case doesn't match the value of expression. It is optional.
- ❖ Break statement terminates the switch block when the condition is satisfied. It is optional, if not used, next case is executed.
- ❖ While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value.

The syntax to use the switch statement is given below.

```
switch (expression){  
  
    case value1:  
        statement1;  
    break;  
    .  
    .  
    .  
    case valueN:  
        statementN;  
    break;  
  
    default:  
        default statement;  
}
```

Er. Shankar pd. Dahal
pdsdahal@gmail.com

2. Loop statements / Looping statement

- ❖ **Looping statement** are the statements execute one or more statement repeatedly several number of times.
- ❖ In java programming language there are three types of loops; while, for and do-while.

Why use loop ?

- ❖ When you need to execute a block of code several number of times then you need to use looping concept in Java language.

Advantage with looping statement

- ❖ Reduce length of Code
- ❖ Take less memory space.
- ❖ Burden on the developer is reducing.
- ❖ Time consuming process to execute the program is reduced.

Difference between conditional and looping statement

- ❖ Conditional statement executes only once in the program where as looping statements executes repeatedly several number of time.

Types :

- ❖ do while loop
- ❖ while loop
- ❖ for loop
- ❖ for-each loop

a. do while loop:

- ❖ The do while loop checks the condition at the end of the loop after executing the loop statements.
- ❖ When the number of iteration is not known and we have to execute the loop at least once, we can use do-while loop.
- ❖ It is also known as the exit-controlled loop since the condition is not checked in advance.

Syntax :

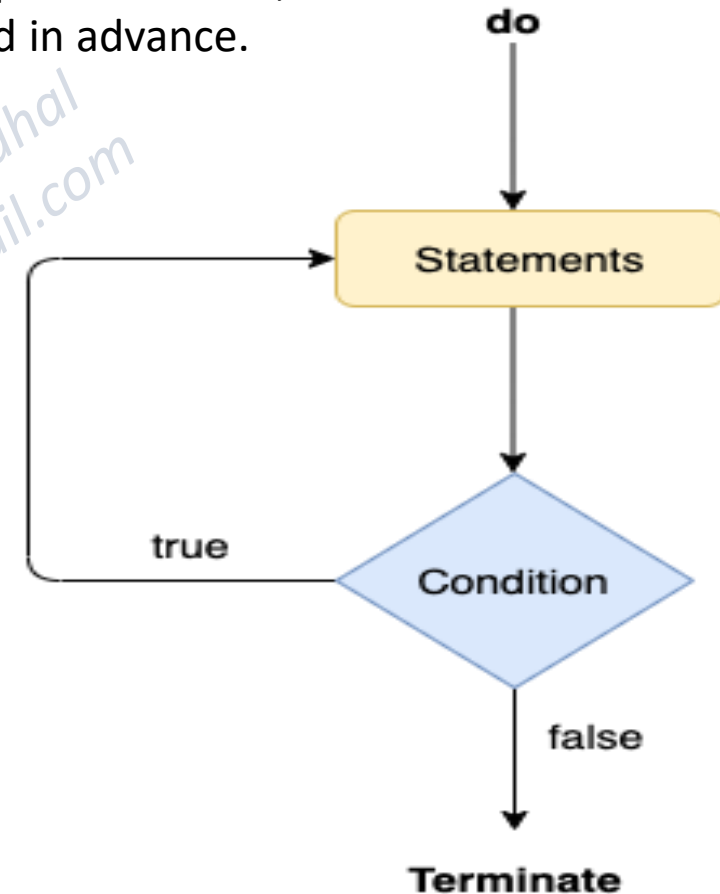
do

{

//statements

} **while** (condition);

Er. Shankar pd. Dahal
pdsdahal@gmail.com

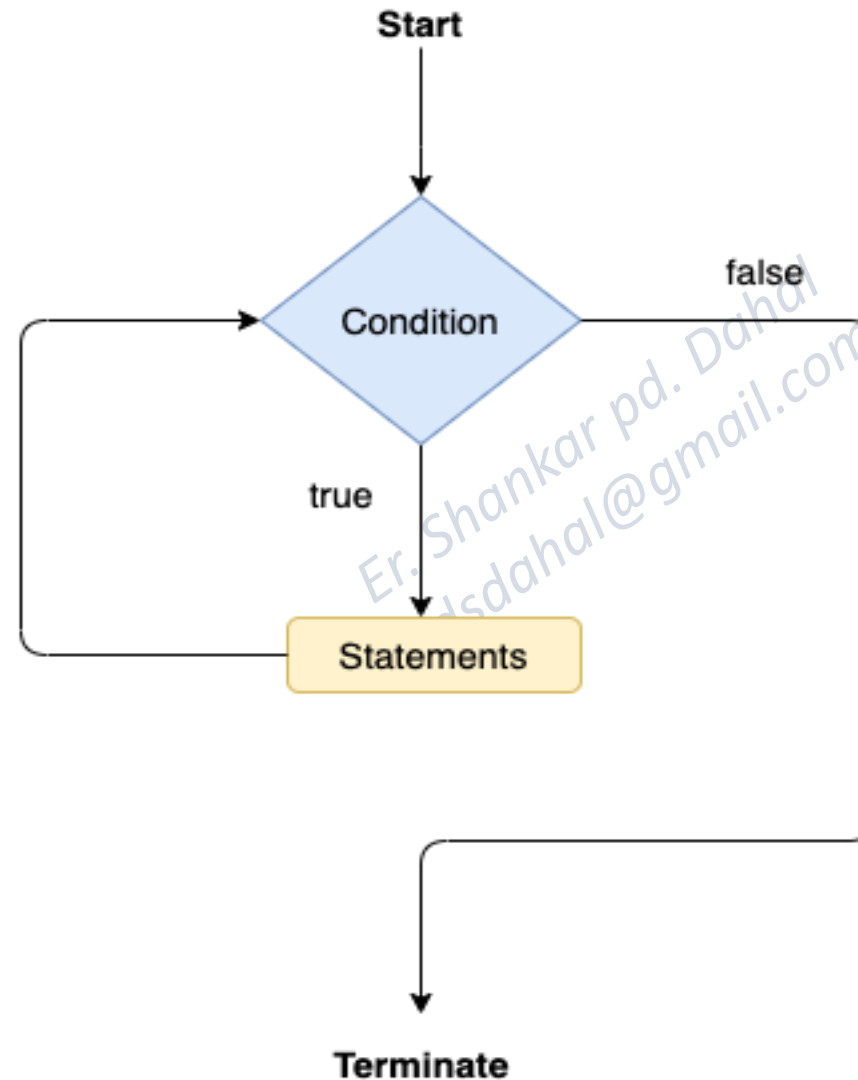


b. while loop

- ❖ The while loop is also used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop.
- ❖ It is also known as the entry-controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

Syntax :

```
while(condition){  
  //looping statements  
}
```



c. for loop :

- ❖ The for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Syntax:

```
for (statement 1; statement 2; statement 3) {
```

```
    // code block to be executed
```

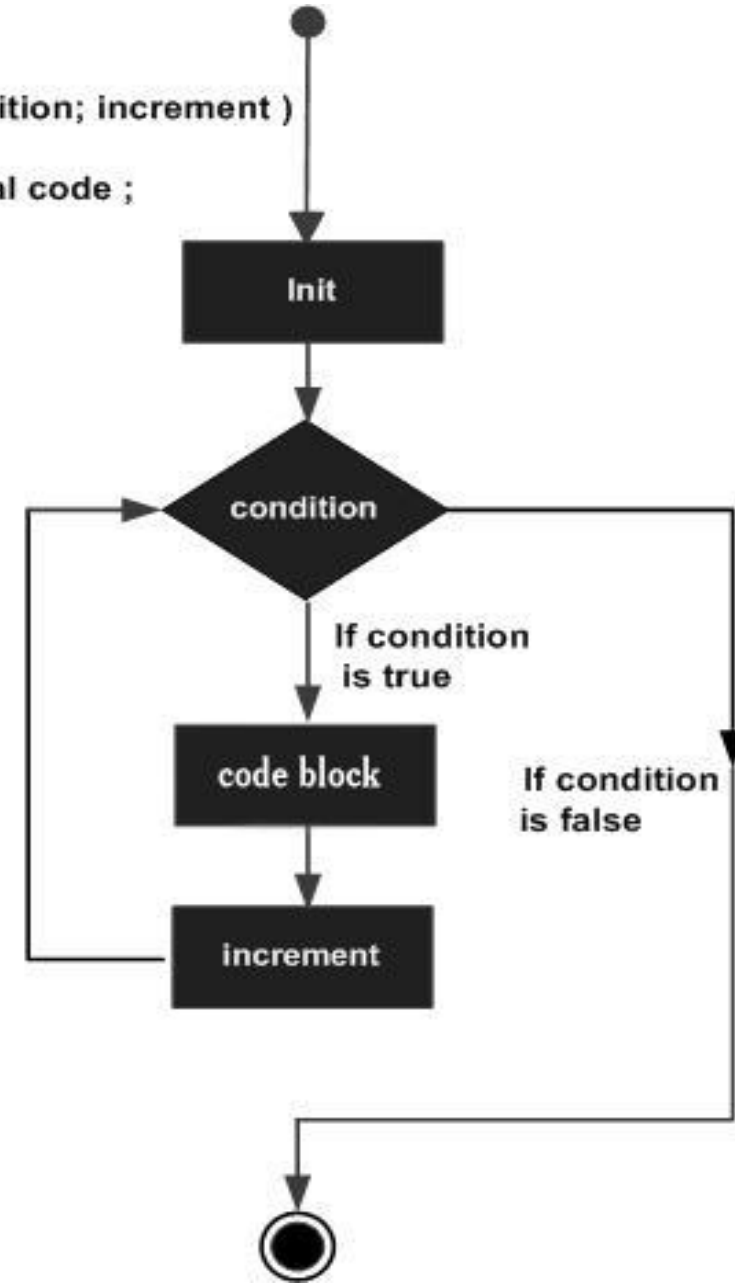
```
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

```
for( init; condition; increment )  
{  
    conditional code ;  
}
```



d. for-each loop / Enhanced For Loop

- ❖ The Java for-each loop or enhanced for loop is introduced since Java5.
- ❖ It provides an alternative approach to traverse the array or collection in Java.
- ❖ It is mainly used to traverse the array or collection elements.
- ❖ It is known as the for-each loop because it traverses each element one by one.
- ❖ In Java, the **for-each** loop is used to iterate through elements of **arrays** and collections (like **ArrayList**).

Advantages

- ❖ It makes the code more readable.
- ❖ It eliminates the possibility of programming errors.

Drawback

- ❖ it cannot traverse the elements in reverse order.
- ❖ you do not have the option to skip any element because it does not work on an index basis.

for-each Loop Sytnax:

```
for(dataType item : array) {  
...  
}
```

Here,

array - an array or a collection

item - each item of array/collection is assigned to this variable

dataType - the data type of the array/collection

3. Jump statements

❖ It is used to transfer execution control from one line to another line.

a. break statement:

- ❖ The break statement in java is used to terminate a switch or looping statement.
- ❖ That means the break statement is used to come out of a switch statement and a looping statement like while, do-while, for, and for-each.
- ❖ It breaks only the inner loop in the case of the nested loop.

We can use break statement in the following cases:

- ❖ Inside the switch case to come out of the switch block.
- ❖ Within the loops to break the loop execution based on some condition.
- ❖ ***The break cannot be used outside the loops and switch statement.***

Syntax:

break;

```
for ( expression )  
{  
    statement1;  
    ....  
    if (condition) true  
        break;  
    .....  
    statement2;  
}
```

out of the loop

```
while (test condition)  
{  
    statement1;  
    ....  
    if (condition) true  
        break;  
    .....  
    statement2;  
}
```

out of the loop

b. continue statement:

- ❖ This statement is used only within looping statements.
- ❖ When the continue statement is encountered then execution control skips the rest of the statements in the looping block and directly jumps to the beginning of the loop.
- ❖ It continues inner loop only if you use the continue statement inside the inner loop.
- ❖ The remaining statements in the loop are skipped. The execution starts from the top of loop again.
- ❖ ***When we use continue statement with while and do-while statements, the execution control directly jumps to the condition.***
- ❖ ***When we use continue statement with for statement the execution control directly jumps to the modification portion (increment/decrement/any modification) of the for loop.***

```
while ( condition)
{
    ....
    continue;
    ....
}
do
{
    ....
    continue;
    ....
} while ( condition) ;
```

```
for (initilization; condition; modification)
{
    ....
    continue;
    ....
}
```

Difference :

Break Statement	Continue Statement
The Break statement is used to exit from the loop constructs.	The continue statement is not used to exit from the loop constructs.
The break statement is usually used with the switch statement, and it can also use it within the while loop, do-while loop, or the for-loop.	The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.
When a break statement is encountered then the control is exited from the loop construct immediately.	When the continue statement is encountered then the control automatically passed from the beginning of the loop statement.
Syntax: break;	Syntax: continue;

c. return statement:

- ❖ The return statement exits from the current method, and control flow returns to where the method was invoked.
- ❖ The return statement has two forms:
 - a) one that returns a value.
 - b) one that doesn't.
- ❖ To return a value, simply put the value (or an expression that calculates the value) after the return keyword.
- ❖ The data type of the returned value must match the type of the method's declared return value. When a method is declared void, use the form of return that doesn't return a value.

Er. Shankar pd. Dahal
pdsdahal@gmail.com