**CORE-JAVA IMP NOTES**

**0.Important Keys Words:**

**0.1 “Static” Key in Java**

🡪The “Static” Keyword in java will Specify the Location of programming.

🡪 ”Static Programming” Components will get memory in the Class.

🡪 ”Non-Static Programming” Components will get memory in the Object.

**0.2 “Parameters in Java”**

🡪 The Parameters are the variables which are used to transfer data from one method to another method.

**0.3 “Return type in Java”**

🡪 Return-type will specify the method will return the value or not after the execution.

**0.4 “Access Modifiers in Java”**

🡪 It will specify scope (Visibility) of programming components with in the project.

🡪 Some important access modifiers in java:

1.Public

2.Private

3.Protected

4.Default

1. Public

* “public” programming components can be accepted with in project.

1. Private

* “Private” programming components are accepted/accessed only in class [Avaiable in only class]

1. Protected

* “protected” programming components are accessed by the child class with in package [Inheritance Process]

1. Default

* The programming components within the class which are declared without any access modifiers are considered as “default”
* “Default” programming components are accessed with in package

**1.Java Programming Components:**

**1.1 VARIABLES:**

🡺 Variables are the “Data Holders” in programs.

**1.Primitive Data types:**

🡺 The “Single-Valued data types” are known as Primitive Data types

***🡺 This Primitive Data types are categorized into four types:***

a) Integer {Byte, Short, Integer, long}

b) Float {Float, Double}

c) Character {Char}

d) Boolean {Boolean}

**2.Non-Primitive Data types:**

🡺 The “Grouped valued data formats” are known as Non-Primitive Data types.

***🡺 This Primitive Data types are categorized into four types:***

a) Class

b) Interface

c) Array

d) Enum

***🡺 Based on data types the variables in java are categorized into two types:***

1.Primitive Data Type Variables

2.Non-Primitive Data Type Variables

**1.Primitive Data Type Variables:**

🡺 The variables which are declared with primitive data types like {Byte, Short, Int, Long, Float, Double, Char, Boolean}

🡺 This Primitive Data type variables will “values”

**2.Non-Primitive Data Type Variables:**

🡺 The Variables which are declared with Non-Primitive Data types like {Class, Interface, Array, Enum}

🡺 This Non-Primitive variables hold “Object-References”

***Based on Static Keyword Variables are categorized into two Types:***

1. Static Variables

2. Non-Static Variables

a) Local Variables

b) Instance Variables

**1. Static Variables:**

🡺The Variables which are declared with “Static” Keyword are known as Static Variables or Class Variables

🡺 These Static Variables will get memory within Class while “Class-loading” and can be a accessed with Class Name.

**2. Non-Static Variables:**

🡺 The Variables which are declared without “Static” Keyword are Non-Static Variables.

🡺 These Non-Static Variables are categorized

***a) Local Variables:***

🡺 The Non-Static Variables which are declared inside methods are known as “Local or Methods Variable”.

🡺 These Local Variables will get the Memory within methods while method execution and can be accessed inside the method Only.

***b) Instance Variables:***

🡺 The Non-Static Variables which are declared outside the methods are known as “Instance or Object Variable”.

🡺 These Variables will get the memory with in Object while Object-Creation and can be accessed with Object-Name.

**1.2 METHODS [FUNCTIONS]:**

🡺 Methods are the “Actions” which are executed to generate results.

***🡺 Based on the “Static” Keyword the methods are categorized into two types:***

1.Static Methods

2.Instance Methods

~~[ 3.Local Methods ]~~

**1.Static Methods:**

🡺 The Methods which are declared with “Static Keyword” are known as Static or Class Methods.

🡺 Static methods will get the memory within the “Memory” and can be accessed with Class\_Name.

🡺 Structure of Static Methods:

***Public Static return\_type Method\_Name( Parameters\_list)***

***{***

***//Method Boby***

***}***

🡺 **Coding-Rules:** “Static Methods” can access “Static Variables” directly but cannot access “Instance Variables” directly.

**2.Instance Methods**

🡺 The Methods which are not declared with “Static” Keyword are known as “Instance or Object Methods”

🡺 This Instance Methods will get memory within object while objection creation and can be accessed with object\_Name.

🡺 Structure of Instance Methods:

***Public return\_type Method\_Name(Parameters\_List)***

***{***

***//Instance Methods***

***}***

🡺 **Coding-Rules:** “Instance Methods” can access “Static Variables” and “Instance Variables” directly.

***🡺Based on Parameters the Methods are categorized into two types:***

1.Methods without Parameters

2.Methods with Parameters

**1.Methods without Parameters:**

🡺The Methods which are declared without parameters are known as “Zero Parameter or Methods without Parameters”

**2.Methods with Parameters:**

🡺 The Methods which are declared with parameters are known as “Parameterized Methods or Methods with Parameters”

***🡺Based on Return Type the Methods are categorized into two types:***

1.Non-Return type Parameters

2. Return type Parameters

**1.Non-Return Type Parameters:**

🡺The Methods which will not return, any value after method execution are known as Non-Return Type.

🡺 The Methods which are declared with “VOID” are known as Non-Return type Methods.

**2.Return Type Parameters:**

🡺The Methods which return the value after the method execution are known as Return type Methods.

🡺 In Return Type Methods the “VOID” is replaced with “DATATYPE”. The Method is returning the Value.

**1.3 OPERATORS IN JAVA**

🡺 Operators is a special Symbol in which is used to “perform some operations”

🡺 The Following are some important Operator:

1.Arithmatic Operators

2.Relational Operators

3.Logical Operators

4.Increment/Decrement Operators

1) Arithmatic Operators:

🡺 The Operators which are used to perform some basic operators are known as Arithmatic Operators

a. Addition [+]

b. Substraction [-]

c. Multiplication [\*]

d. Division [/] (generate Quotient)

e. ModDivision [%] (generate Remainder)

2) Relational Operators:

🡺 The Operators which are used to compare Two values and generate Boolean result.

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| > | Greater than |
| >= | Greater than Or Equal |
| < | Less Than |
| <= | Less than Or Equal |
| == | Is Equal to |
| != | Not Equal To |

3) Logical Operators:

🡺The Operators which are used to compare two comparsion are known as Logical Operators

|  |  |
| --- | --- |
| Operator | Meaning |
| && | Logical and |
| || | Logical or |
| ! | Logical Not |

4) Increment/Decrement Operators:

🡺Increment Operator will Increment the Value by “One”

🡺Decrement Operator will Decrement the Value by “one”

🡺[++] 🡪 Increment

🡺[--] 🡪 Decrement

**1.4 CONTROL STRUCTURES IN JAVA**

🡺 The Structures which are used to control the execution loop in the program are known as Control Structures.

1. Selection Statements
2. Iterative Statements
3. Branching Statements
4. **Selection Statements**

🡺The Statements which are used to select some lines of program for execution are known as Selection Statements.

1. **Simple-If:**

🡺In Simple-If we have only “True” Condition which means only if-block.

🡺Syntax:

***If(Condition)***

***{***

***//Statements***

***}***

1. **If-Else:**

🡺In If-Else we have both “True and False” blocks.

🡺SYNTAX:

***If(Condition)***

***{***

***//Statements***

***} else{***

***//Statements***

***}***

1. **Nested-If:**

🡺The process of declaring “If inside the if” is Known as Nested-If

🡺SYNTAX:

If(Condition)

{

If(Condition)

{

//Statements

}

}

1. **Laddar If-Else:**

🡺The process of inter-linking more If-Else blocks is known as Laddar If-Else.

🡺SYNTAX:

If(Condition)

{

//Statements

}

Else if(Condition)

{

//Statements

}

……

else

{

//Statements

}

1. **Switch-Case:**

🡺The process of selecting one from “Multiple Options” is Known as Switch Case.

🡺SYNTAX:

Switch(value)

{

case 1:

//Statements

break;

case 2:

//Statements

break;

.

.

Default:

//Statements

}

1. **Iterative Statements**
2. **While-loop:**

🡪 In While looping structure of the condition is checked first; if the condition is true then loop-body is executed; and this process is repeated until the condition is false.

🡪 **Syntax:**

***while(Condition){***

***//Loop Body***

***}***

1. **Do-While Loop:**

**🡪** Do While looping Structure the loop body is executed; first then condition is checked. This Process is repeated until condition is False.

**🡪 Syntax:**

***do{***

***//Loop-Body***

***}***

***while(Condition)***

1. **For Loop:**

**🡪** For-Loop is more simple in representation when compared to while and do-while loops, bcz “initialization”; “condition”; and “Incre/Decre” are declared in the same line.

**🡪 Syntax:**

***For(Intialization;Condition;Inc/Dec){***

***//Loop-Body***

***}***

1. **Branching Statements**
2. **Break:**

* “break” is used to stop switch-case execution and which is also used to stop Iterative statements based on condition

1. **Continue:**

* “continue” is used to skip some lines from the iteration based on some Condition

1. **Return:**

* “return” is used to transfer the execution control from one method to another method

1. **Exit:**

* “exit” is used to stop the program execution.