## JAVA

--------------------------------------------------------------------------------------------------------------------------------------------------------------------**Variable Name: it’s the hold the value with its data type OR it’s a address of memory location.**

**1. Int x …………(data type + variable ) //declaration.**

**2. int x = 100; ……..(data type + variable + declaration + initialization)**

**Data Type in Java:**

**1. Primitive Data Type : byte, int, long ,float ,double, char, Boolean**

**2. Derived Data Type : Array , Array List , String, Hash Map ,class , interface .**

**Operators in Java:**

**1. Arithmetic Operators : + (plus) , - (minus) , \* (multiplication) , / (divided), %(modules)**

**2. Relational Operators : == (equal) , < (smaller than), >(Greater than) , >= (Greater than equal to), <= (Smaller than equal to) , != (not equal to)**

**3. Logical Operators : && (and), ||(Or) , ! (Not)**

**4. increment /decrement operators : ++ , --**

**5. Assignment operators : =**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **X && Y** | **X||Y** | **!X** |
| **True** | **True** | **True** | **True** | **False** |
| **True** | **False** | **False** | **True** | **False** |
| **False** | **True** | **False** | **True** | **True** |
| **False** | **False** | **True** | **False** | **True** |

**Java Control Statement:**

**Conditional Statements: if else , if else if , switch statement**

**switch (expression)**

**{**

**case condition 1: statements; break;**

**case condition 2: statements; break;**

**case condition 3: statements; break;**

**default: statement;**

**}**

**If (Condition)**

**{**

**//code**

**}**

**else if (Condition)**

**{**

**//code**

**}**

**If (Condition)**

**{**

**//code**

**}**

**else**

**{**

**//code**

**}**

**Loop / Iterative Statements: set of statements can be repeated multiple times based on a condition.**

**While, do…while, for loop**

**for (start ; process; increment/decrement)**

**{**

**// code**

**}**

**do**

**{**

**//code**

**//Increment or decrement logic**

**}**

**While (condition);**

**While (condition)**

**{**

**//code**

**//Increment or decrement logic**

**}**

**Jump Statements: break (will stop the execution), continue (will skip the execution)**

**Access Modifier :**

**1. PRIVATE :- Accessible only within class.**

**2. PROTECTED :- Accessible only within package but we use this outside the package using inheritance**

**3. PUBLIC:- Accessible all over means with in class ,package and outside the package**

**4. DEFAULT:- Accessible only with in package.**

**Non-access Modifier :**

**1. Static:-**

* **Key-word used for with Method and variable, when this used with method and variable this variable and methods compute at compile time , it’s totally independed from the other methods and variables which is present in class.**
* **This key word not applicable for local variables**

**2. Final:- Key-Word used with**

* **Variable :- Variable name cannot be modified.**
* **Method:- Method name cannot be override.**
* **Class :- Class Name Cannot be inherited**

**3. Synchronized:-**

* **The synchronized keyword used to indicate that a method can be accessed by only one thread at a time. The synchronized modifier can be applied with any of the four access level modifiers.**

**4. Abstract:-**

* **An abstract class can never be instantiated (we cannot create object of it). If a class is declared as abstract then the sole purpose is for the class to be extended.**
* **A class cannot be both abstract and final (since a final class cannot be extended). If a class contains abstract methods then the class should be declared abstract. Otherwise, a compile error will be thrown.**
* **An abstract class may contain both abstract methods as well normal methods.**

**Method:**

**It is a block of code which is execute only when programmer tigers it (run only when it’s called).**

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| **Occurs within One Class** | **Occurs With Two Different Class** |
| **Method Name Are Same but Method Parameter or parameter order or Parameter Data-type are different** | **Method Name Are Same but Method Parameter or parameter order or Parameter Data-type are different** |
| **Method return type May be same or different** | **Method return type are same** |
| **It is Example of compile time polymorphism** | **It is Example of run time polymorphism** |

**Constructors:**