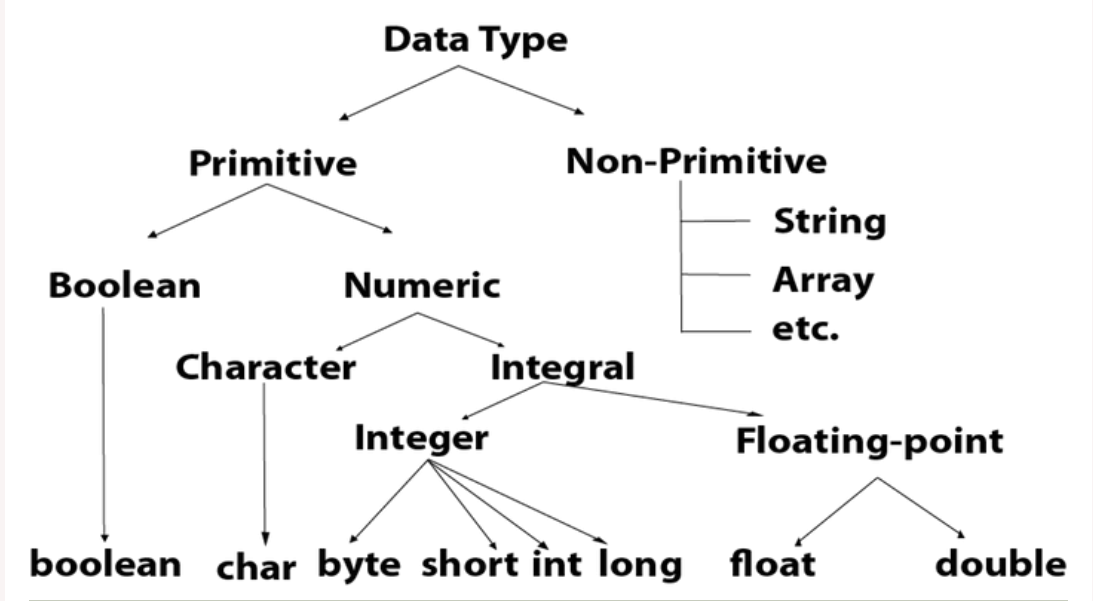
**JAVA**

**Introduction:** Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995. James Gosling is known as the father of Java. Before Java, its name was Oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

* **class** keyword is used to declare a class in Java.
* **public** keyword is an access modifier that represents visibility. It means it is visible to all.
* **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
* **void** is the return type of the method. It means it doesn't return any value.
* **main** represents the starting point of the program.
* **String[] args** or **String args[]** is used for command line argument. We will discuss it in coming section.
* **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class. We will discuss the internal working of System.out.println() statement in the coming section.

**Java Datatypes**



1. **Java Operators**

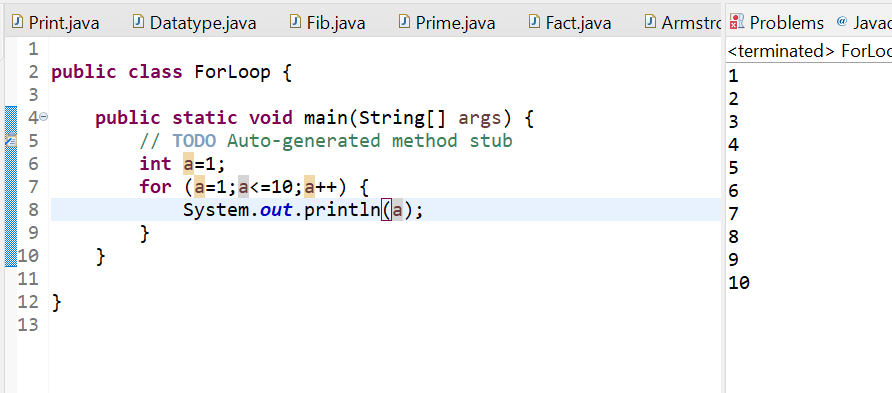
**Operator** in Java is a symbol that is used to perform operations.

|  |  |  |
| --- | --- | --- |
| **Operator Type** | **Category** | **Precedence** |
| Unary | postfix | *expr*++ *expr*-- |
| prefix | ++*expr* --*expr* +*expr* -*expr* ~ ! |
| Arithmetic | multiplicative | \* / % |
| additive | + - |
| Shift | shift | << >> >>> |
| Relational | comparison | < > <= >= instanceof |
| equality | == != |
| Bitwise | bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| Logical | logical AND | && |
| logical OR | || |
| Ternary | ternary | ? : |
| Assignment | assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

**Java Control Statements**

Java provides three types of control flow statements.

* 1. **Decision Making statements:** As the name suggests, decision-making statements decide which statement to execute and when. Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided.
     + if statements
     + switch statement
  2. **Loop statements:** sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.
     + do while loop
     + while loop
     + for loop
     + for-each loop
  3. **Jump/Transfer statements:** Jump statements are used to transfer the control of the program to the specific statements. In other words, jump statements transfer the execution control to the other part of the program.
     + break statement
     + continue statement
* **If Statement:** If the condition is true the statements inside the if block will execute.
* **If –else Statement:** If the condition is true the statements inside the if block will execute otherwise the else block will execute
* **If –else-if ladder Statement:** It is the chain of if-else statements that create a decision tree where the program may enter in the block of code where the condition is true.
* **Nested If Statement:** In nested if-statements, the if statement can contain an **if** or **if-else** statement inside another if or else-if statement.
* **Switch Case Statement:** 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.
* 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.
* **For Loop:** We use the for loop only when we exactly know the number of times, we want to execute the block of code.



* **While Loop:** We use the for loop when we do not know the exact number of times, we want to execute the block of code. 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.
* **Do while Loop:** When the number of iterations 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.

* **For each Loop:** Java provides an enhanced for loop to traverse the data structures like array or collection. In the for-each loop, we don't need to update the loop variable.

* **Break Statement:** As the name suggests, the break statement is used to break the current flow of the program and transfer the control to the next statement outside a loop or switch statement. However, it breaks only the inner loop in the case of the nested loop.
* **Continue Statement:** It skips the specific part of the loop and jumps to the next iteration of the loop immediately.

1. **Java Object Oriented Programming Concepts**

**Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.

* + **Object:** Any entity that has state and behavior is known as an object. An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.
  + **Class:** Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.
  + **Inheritance:** When one object acquires all the properties and behaviors of a parent object, it is 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. In Java, we use method overloading and method overriding to achieve polymorphism.

**Method overloading**

**Method Overriding**

* + **Abstraction:** Hiding internal details and showing functionality is known as abstraction. In Java, we use abstract class and interface to achieve abstraction.
  + **Encapsulation:** Binding (or wrapping) code and data together into a single unit are known as encapsulation. A java class is the example of encapsulation.