JAVA EBOOK

Java

by @curious_.programmer

Python, Java, C, C++ Notes Uploaded on Telegram

Join For More Ebooks!

@curious_.programmer

About this PDF:

Clean Code is another classic for Java programmers. Anyone can learn to code, but it takes practice to write clean code. When code is poorly written, you can lose countless hours and significant resources. This PDF teaches you to read code and challenges you to think about what's right about that code. It is divided into three parts - Basics, Types, and Control flows. And AWT , Swing.

This PDF includes the materials we covered complete java for beginners.

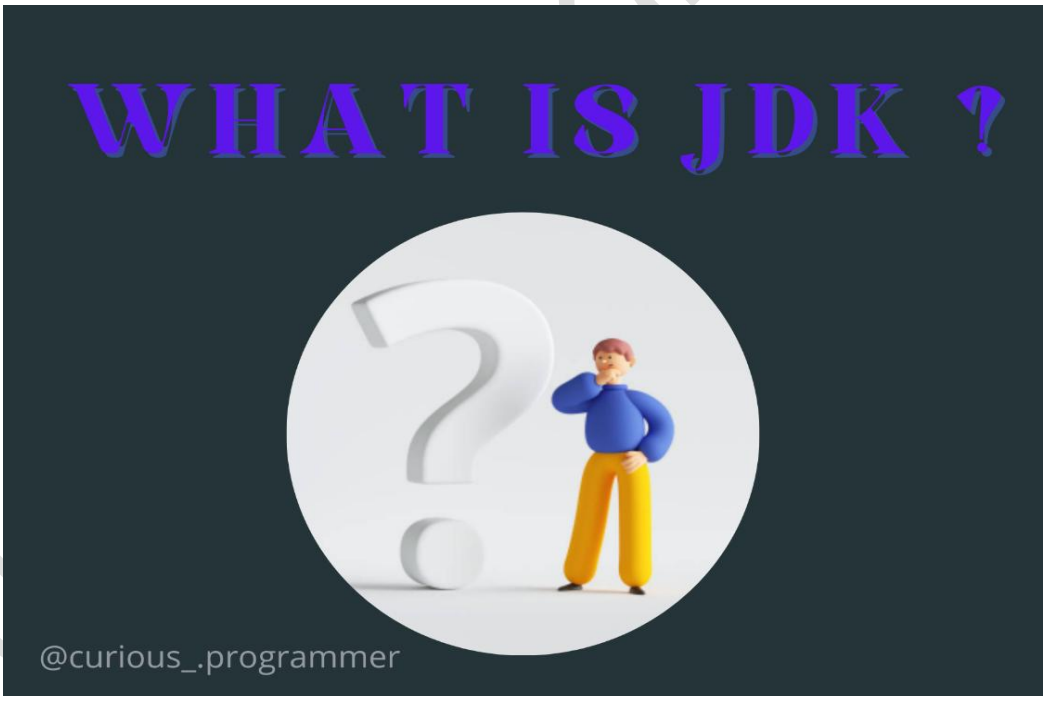
- If you're completely new to Java, this is the first PDF by curious_.programmer worth investing in.

- It covers all the fundamentals of Java.

Basics

Java is a powerful general-purpose programming language. It is used to develop desktop and mobile applications, big data processing, embedded systems, and so on. According to Oracle, the company that owns Java, Java runs on 3 billion devices worldwide, which makes Java one of the most popular programming languages.

Java Development Kit



We use Java Development Kit (JDK) to build Java applications. JDK contains a compiler, the Java Runtime Environment (JRE) and a library of classes that we useto build applications.

Java Editions:

We have four editions of Java, each used for building a different type of application:

Java Standard Edition (SE): the core Java platform. It contains all of the libraries that every Java developer must learn.

Java Enterprise Edition (EE): used for building very large scale, distributed systems. It's built on top of Java SE and provides additional libraries for building fault-tolerant, distributed, multi-tier software.

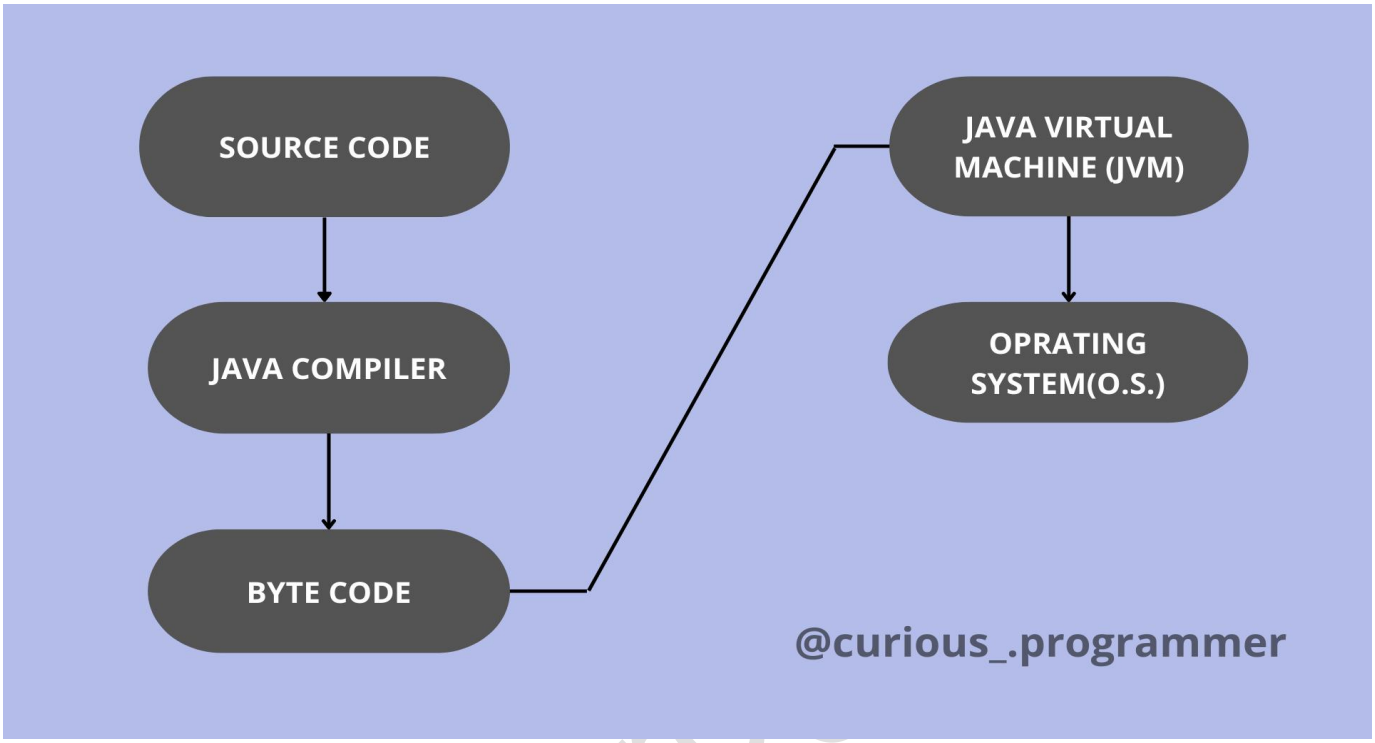
Java Micro Edition (ME): a subset of Java SE, designed for Mobile devices.

Java Card: used in smart cards.

How Java Code Gets Executed

The Java compiler takes Java code and compiles it down to Java Byte code which isa cross-platform format. When we run Java applications, Java Virtual Machine (JVM) gets loaded in the memory. It takes our bytecode as the input and translatesit to the native code for the underlying operating system. There are various implementations of Java Virtual Machine for almost all operating systems.

Architecture of Java Applications



The smallest building blocks in Java programs are **methods** (also called functions in other programming languages). We combine related methods in **classes**, and related classes in **packages**. This modularity in Java allows us to break down large programs into smaller building blocks that are easier to understand and re-use.

5 Interesting Facts about Java

1. Java was developed by James Gosling in 1995 at Sun Microsystems (later acquired by Oracle).
2. It was initially called Oak. Later it was renamed to Green and was finally renamed to Java inspired by Java coffee.
3. Java has close to 9 million developers worldwide.

4. About 3 billion mobile phones run Java, as well as 125 million TV sets and every Blu-Ray player.

5. According to indeed.com, the average salary of a Java developer is just over
    a. $100,000 per year in the US.

• Types

Variables

As the name suggests, data types specify the type of data that can be stored inside variables in Java. Java is a statically-typed language. This means that all variables must be declared before they can be used.
A variable is a location in memory (storage area) to hold data. To indicate the storage area, each variable should be given a unique name (identifier).

We use variables to temporarily store data in computer's memory.

In Java, the typeof a variable should be specified at the time of declaration.

In Java, we have two categories of types:

- **Primitives:** for storing simple values like Numbers, Strings and Booleans.
- **Reference Types:** for storing complex objects like email messages.

Primitive Types

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.

Type	Bytes	Range
byte	1	[-128, 127]
short	2	[-32K, 32K]
int	4	[-2B, 2B]
long	8	
float	4	
double	8	
char	2	A, B, C, ...
boolean	1	true / false

Declaring Variables

- In Java, we terminate statements with a semicolon.
- We enclose characters with single quotes and strings (series of characters) withdouble quotes.
- The default integer type in Java is int. To represent a long value,

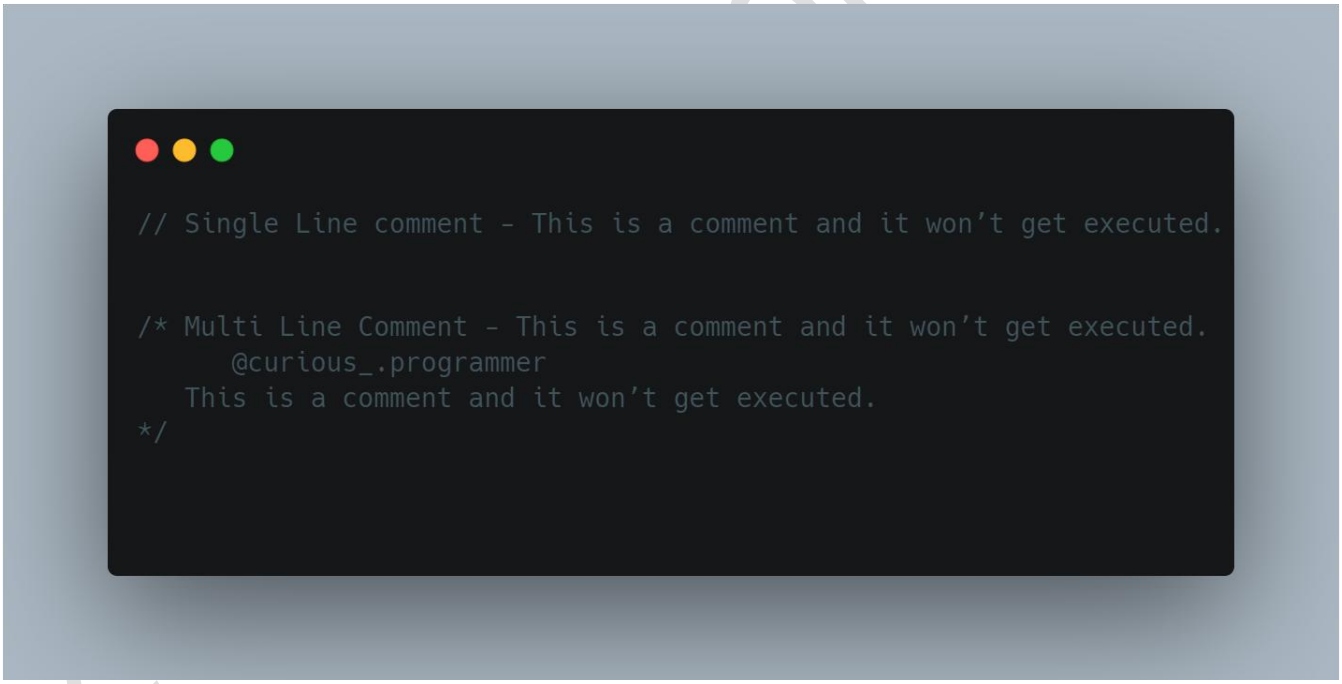
we should add L to it as a postfix.

- The default floating-point type in Java is double. To represent a float, we should append F to it as a postfix.

Comments

We use comments to add notes to our code. Two Types Of comments in Java commonly used.

- Single Line Comment:
- Multi Line Comment:



```
// Single Line comment - This is a comment and it won't get executed.


/* Multi Line Comment - This is a comment and it won't get executed.
      @curious_.programmer
   This is a comment and it won't get executed.
*/
```

Reference Types

In Java we have 8 primitive types. All the other types are reference types. Thesetypes don't store the actual objects in memory. They store the reference (or the address of) an object in memory.

To use reference types, we need to allocate memory using the new operator. Thememory gets automatically released when no longer used.

```
Date now = new Date();
```

Strings

In Java, a string is a sequence of characters. For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'. We use double quotes to represent a string in Java.

Strings are reference types but we don't need to use the new operator to allocate memory to them. We can declare string variables like the primitives since we usethem a lot.

```java
//Curious._programmer
class Main {
  public static void main(String[] args) {

    // create strings
    String first = "Java";
    String second = "Python";
    String third = "JavaScript";

    // print strings
    System.out.println(first);   // print Java
    System.out.println(second);  // print Python
    System.out.println(third);   // print JavaScript
  }
}
```

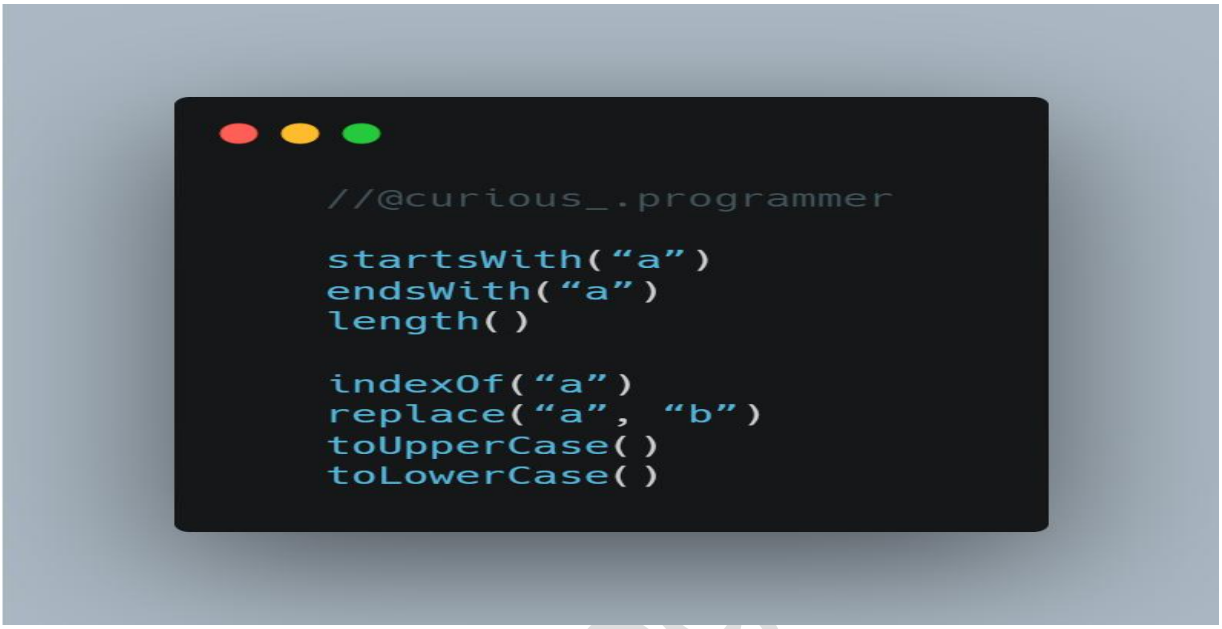
```
  return go(f, seed, [])
}
```

Output Clear

```
java -cp /tmp/zpNjJEpoWW Main
Java
Python
JavaScript
```

Useful String Methods

The String class in Java provides a number of useful methods:



Strings are immutable, which means once we initialize them, their value cannot bechanged. All methods that modify a string (like to Upper Case) return a new string object. The original string remains unaffected.

Escape Sequences

If you need to use a backslash or a double quotation mark in a string, you need toprefix it with a backslash. This is called escaping.

Common escape sequences:

- \\
- \"
- \n (new line)
- \t (tab)

Arrays

An array is a collection of similar types of data.

For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

```
String [] array = new String[100];
```

Here, the above array cannot store more than 100 names. The number of values in a Java array is always fixed.

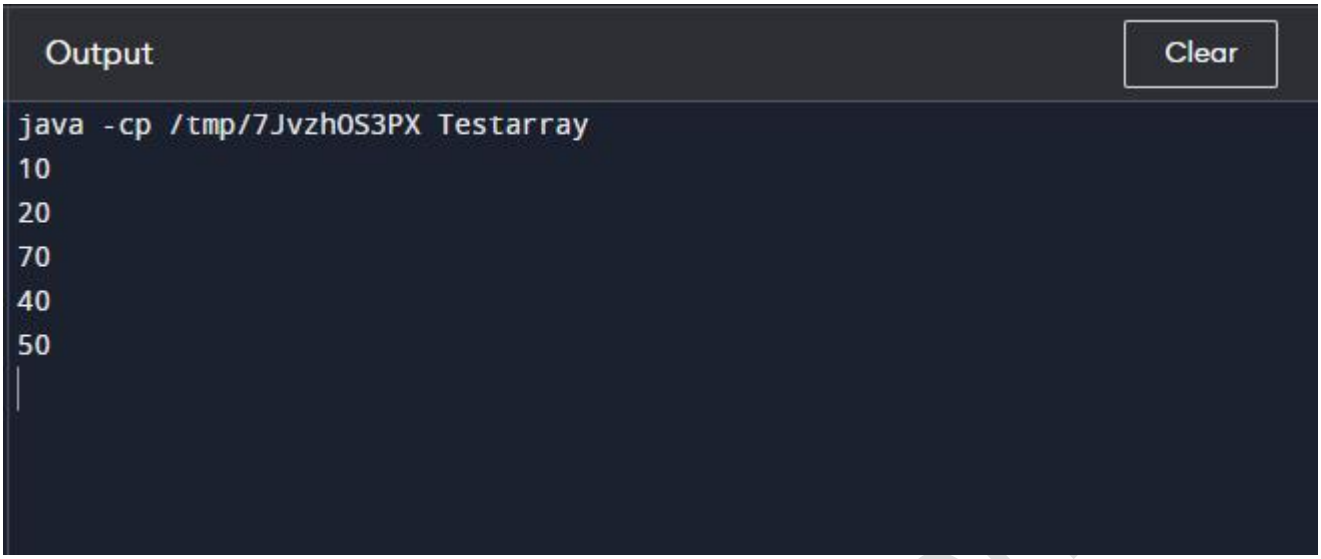
We use arrays to store a list of objects. We can store any type of object in an array (primitive or reference type). All items (also called elements) in an array have the same type.

Java arrays have a fixed length (size). You cannot add or remove new items once you instantiate an array. If you need to add new items or remove existing items, you need to use one of the collection classes.

```java
//@curious_.programmer

int[] numbers = new int[5];
numbers[0] = 10;
numbers[1] = 20;
numbers[2] = 30;
numbers[3] = 40;
numbers[4] = 50;


// Shortcut
int[] numbers = { 10, 20, 30, 40, 50};
```

// creating and initializing an array of 5 elements.

```java
//Curious.Programmer
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0;i<a.length;i++)//length is the property
of array
System.out.println(a[i]);
}}
```

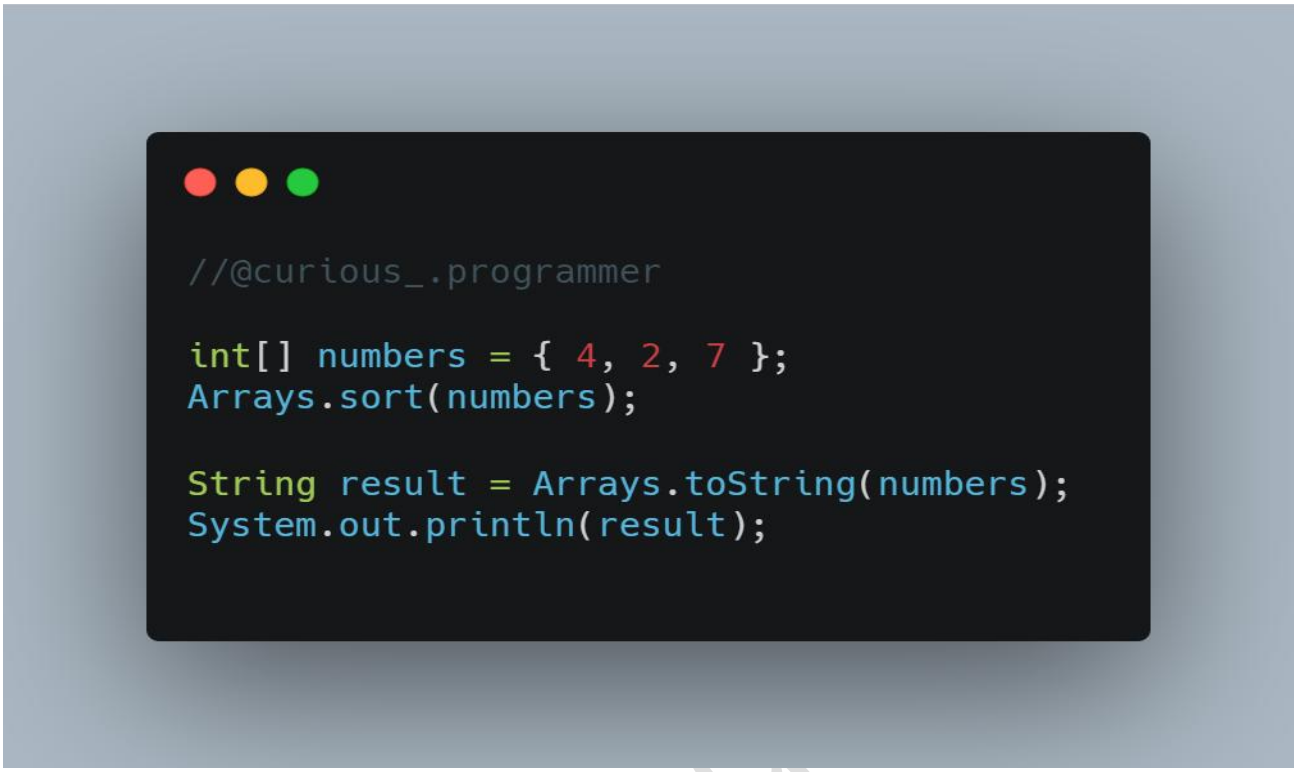


The Array Class

The **java.util.Arrays** class contains a static factory that allows arrays to be viewed as lists.
Following are the important points about Arrays -

- This class contains various methods for manipulating arrays (such as sorting and searching).
- The methods in this class throw a Null Pointer Exception if the specified array reference is null.

The Array class provides a few useful methods for working with arrays.



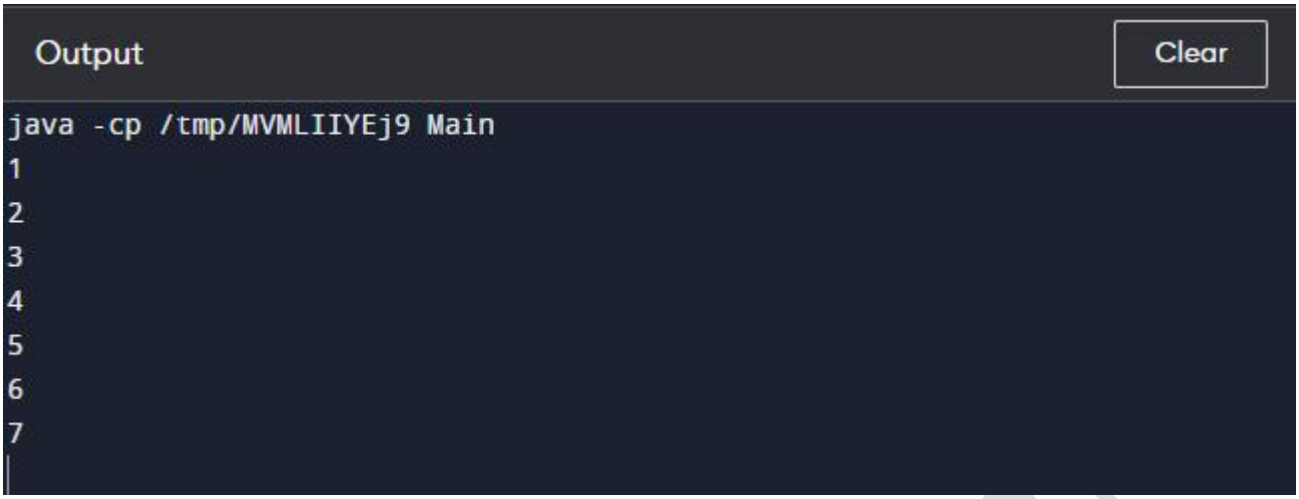
Multi-dimensional Arrays

A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself. For example,

```
// Creating a 2x3 array (two rows, three columns)
```

```java
//@curious_.programmer

// Creating a 2x3 array (two rows, three columns)
int[2][3] matrix = new int[2][3];
matrix[0][0] = 10;

// Shortcut
int[2][3] matrix = {
                       { 1, 2, 3 },
                       { 4, 5, 6 }
                   };
```

```java
//Courious._programmer
public class Main {
   public static void main(String[] args) {
     int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
     for (int i = 0; i < myNumbers.length; ++i) {
        for(int j = 0; j < myNumbers[i].length; ++j) {
           System.out.println(myNumbers[i][j]);
        }
     }
   }
}
```



Constants

Constants (also called final variables) have a fixed value. Once we set them, wecannot change them.

```
final float INTEREST_RATE = 0.04;
```

By convention, we use CAPITAL LETTERS to name constants. Multiple words canbe separated using an underscore.

Arithmetic Expressions

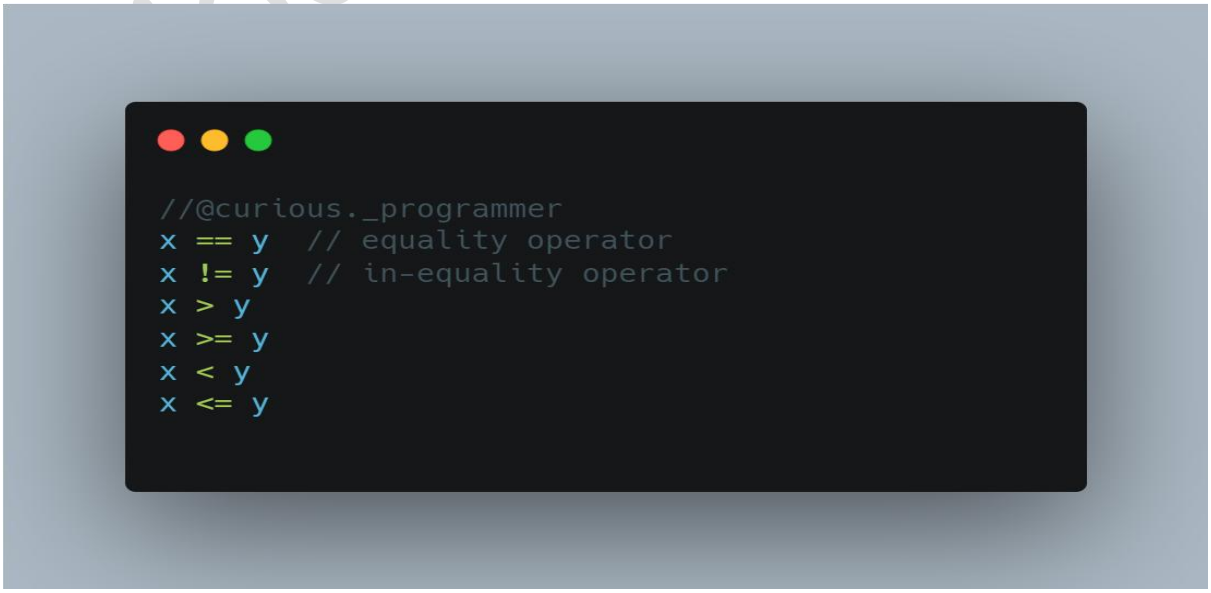
Arithmetic operators are used to perform arithmetic operations on variables and data. For example,

```
//@curious._programmer
int x = 10 + 3;
int x = 10 - 3;
int x = 10 * 3;
int x = 10 / 3;      // returns an int
float x = (float)10 / (float)3; // returns a float
int x = 10 % 3; // modulus (remainder of division)
```

Comparison Operators

operators are used to check the relationship between two operands. We use comparison operators to compare values.

It returns either true or false.

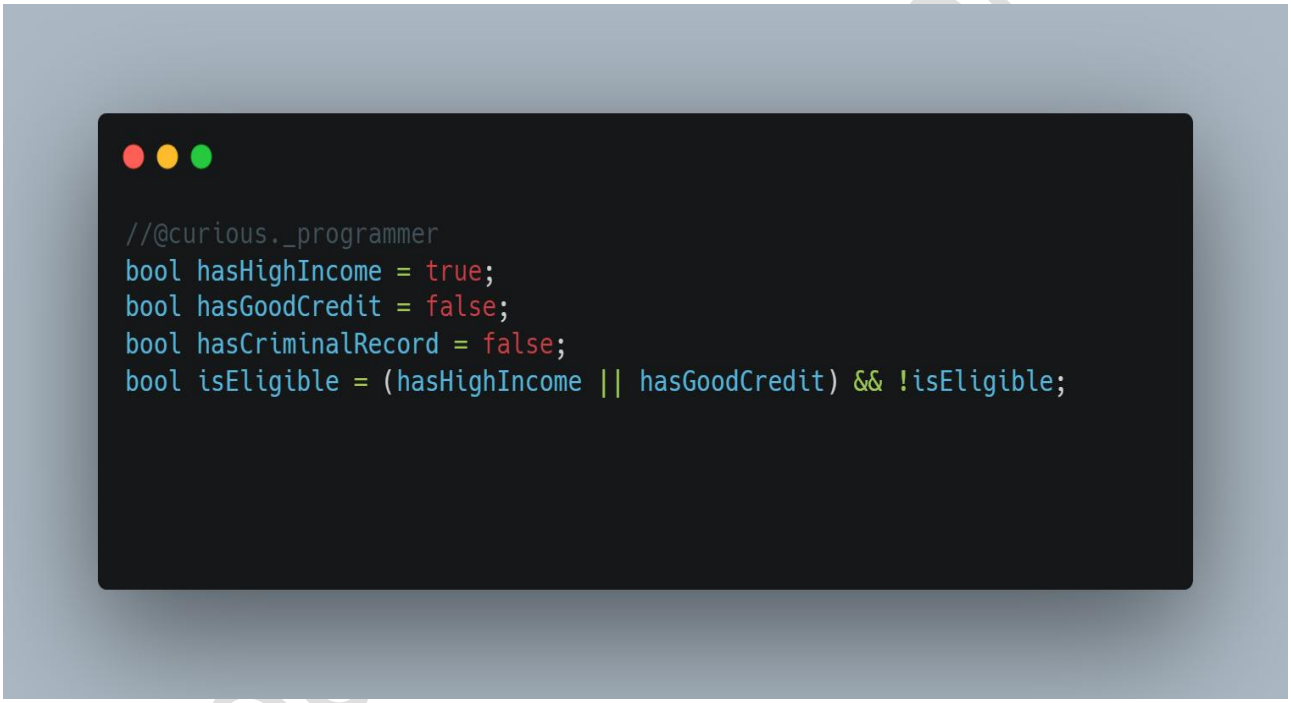


```
//@curious._programmer
x == y  // equality operator
x != y  // in-equality operator
x > y
x >= y
x < y
x <= y
```

Logical Operators

We use logical operators to combine multiple boolean values/expressions.

- x && y (AND): if both x and y are true, the result will be true.
- x || y (OR): if either x or y or both are true, the result will be true.
- !x (NOT): reverses a boolean value. True becomes false.



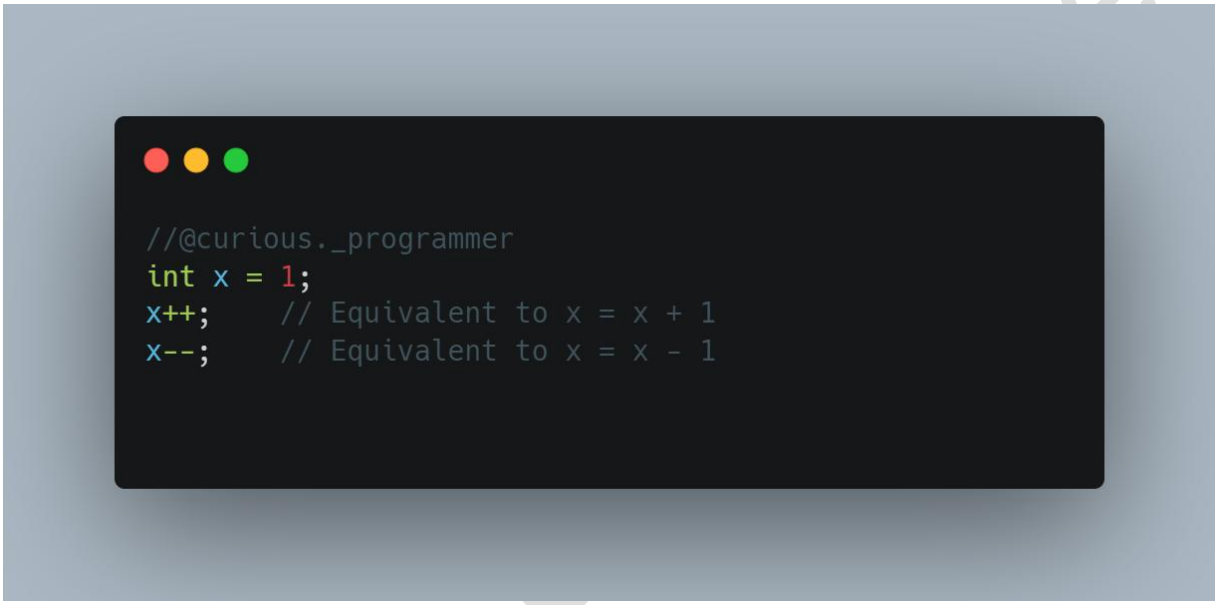
The Ternary Operator

A ternary operator evaluates the test condition and executes a block of code based on the result of the condition.

```
String className = (income > 100_000) ? “First” : “Economy”;
```

Increment and Decrement Operators

Java also provides increment and decrement operators: ++ and -- respectively. ++
increases the value of the operand by 1, while -- decrease it by 1.



Augmented Assignment Operator

```
int x = 1;
x += 5;   // Equivalent to x = x + 5
```

Order of Operations

Multiplication and division operators have a higher order than addition and subtraction. They get applied first. We can always change the order using parentheses.

```
int x = 10 + 3 * 2;          // 16
int x = (10 + 3) * 2;        // 26
```

Casting

The process of converting the value of one data type (int, float, double, etc.) to another data type is known as typecasting.

In Java, we have two types of casting:

- **Implicit**: happens automatically when we store a value in a larger or more precise data type.
- **Explicit**: we do it manually.

```
// Implicit casting happens because we try to store a short
// value (2 bytes) in an int (4 bytes).

short x = 1;
int y = x;

// Explicit casting

int x = 1;
short y = (short) x;
```

We can convert String to an int in java using Integer.parseInt() method. To convert String into Integer, we can use Integer.valueOf() method which returns instance of Integer class.
To convert a string to a number, we use one of the following methods:

```java
//Curious.Programmer
class Main {
  public static void main(String[] args) {
    // create int type variable
    int num = 10;
    System.out.println("The integer value: " + num);

    // convert into double type
    double data = num;
    System.out.println("The double value: " + data);
  }
}
```

Output

Clear

```
java -cp /tmp/N6a2EZhlb5 Main
The integer value: 10
The double value: 10.0
```

Formatting Numbers

```
//@curious._programmer
NumberFormat currency =NumberFormat.getCurrencyInstance();
String result = currency.format("123456"); // $123,456


NumberFormat percent = NumberFormat.getPercentInstance();
String result = percent("0.04"); // 4%
```

Reading Input

Java Scanner Class

Java Scanner class allows the user to take input from the console. It
It belongs to java.util package. It is used to read the input of primitive.
Types like int, double, long, short, float, and byte.
It is the easiest way to read input in Java program.

```
//@curious._programmer
Scanner scanner = new Scanner(system.in);
double number = scanner.nextDouble();
byte number = scanner.nextByte();
String name = scanner.next();
String line = scanner.nextLine();
```

• Control Flow

If Statements

In programming, we use the if..else statement to run a block of code among more than one alternatives.
For example, assigning grades (A, B, C) based on the percentage obtained by a student.

- if the percentage is above 90, assign grade A
- if the percentage is above 75, assign grade B
- if the percentage is above 65, assign grade C

Here is the basic structure of an if statement. If you want to execute multiple statements, you need to wrap them in curly braces.

```java
//Curious.Programmer
class Main {
  public static void main(String[] args) {

    int number = 0;
    // checks if number is greater than 0
    if (number > 0) {
      System.out.println("The number is positive.");
    }
    // checks if number is less than 0
    else if (number < 0) {
      System.out.println("The number is negative.");
    }
    // if both condition is false
    else {
      System.out.println("The number is 0.");
    }
  }
}
```

Output Clear

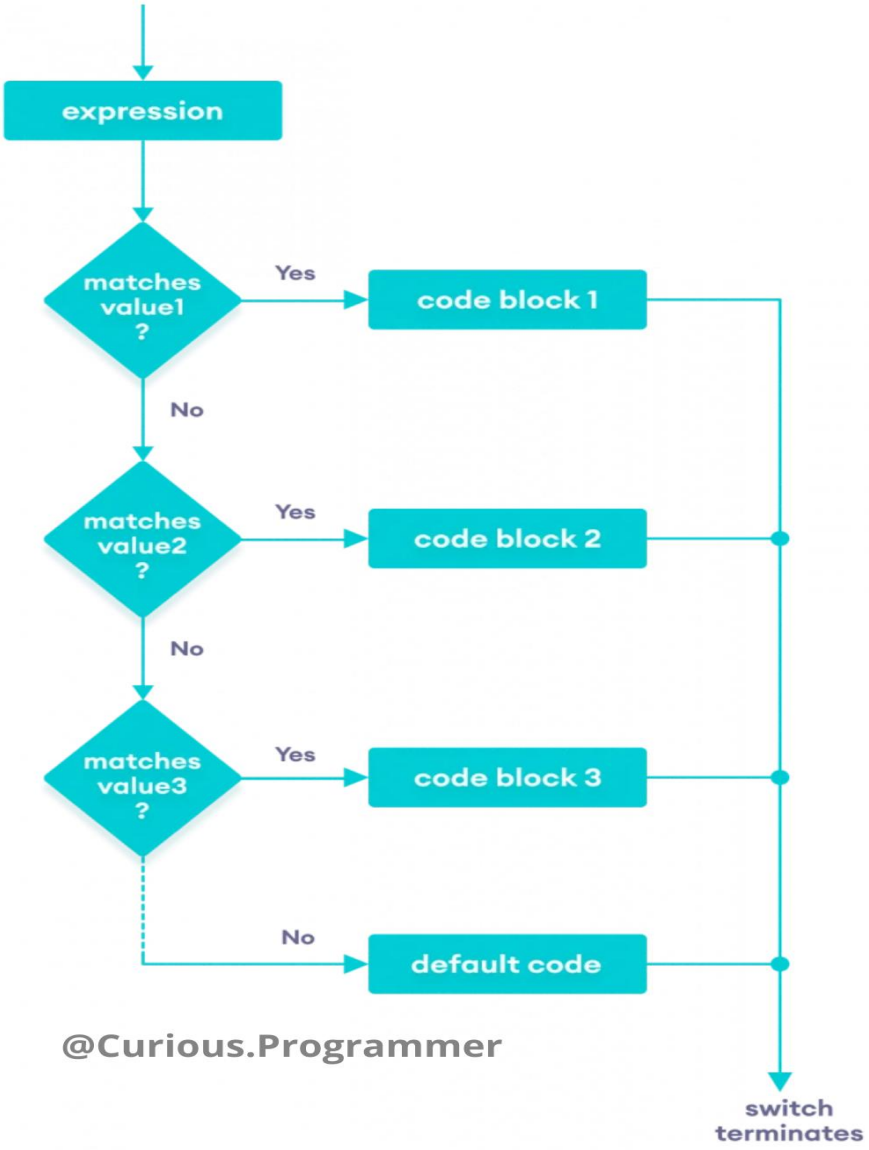
```
java -cp /tmp/VeOtGnR48a Main
The number is 0.
```

Switch Statements

We use switch statements to execute different parts of the code depending on thevalue of a variable.

After each **case** clause, we use the break statements to jump out of the switchblock.

The switch statement allows us to execute a block of code among many alternatives.



The expression is evaluated once and compared with the values of each case.

- If expression matches with value1, the code of case value1 are executed. Similarly, the code of case value2 is executed if expression matches with value2.
- If there is no match, the code of the default case is executed.

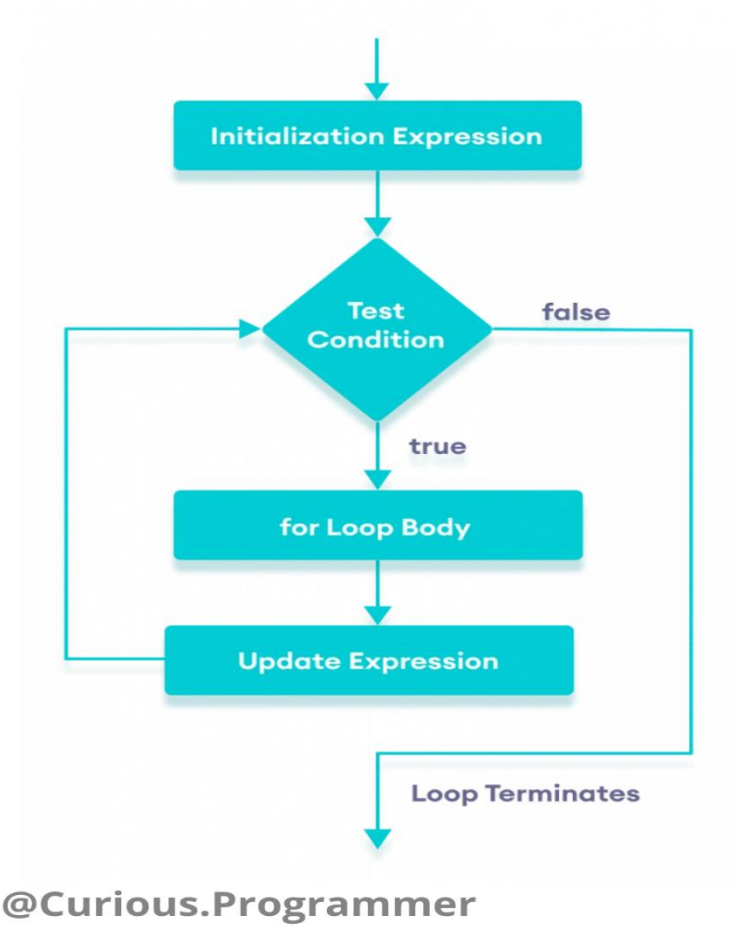
```java
//Curious.Programmer
class Main {
  public static void main(String[] args) {

    int number = 42;
    String size;

    // switch statement to check size
    switch (number) {

      case 29:
        size = "Small";
        break;

      case 42:
        size = "Large";
        break;

      default:
        size = "Unknown";
        break;

    }
    System.out.println("Size: " + size);
  }
}
```

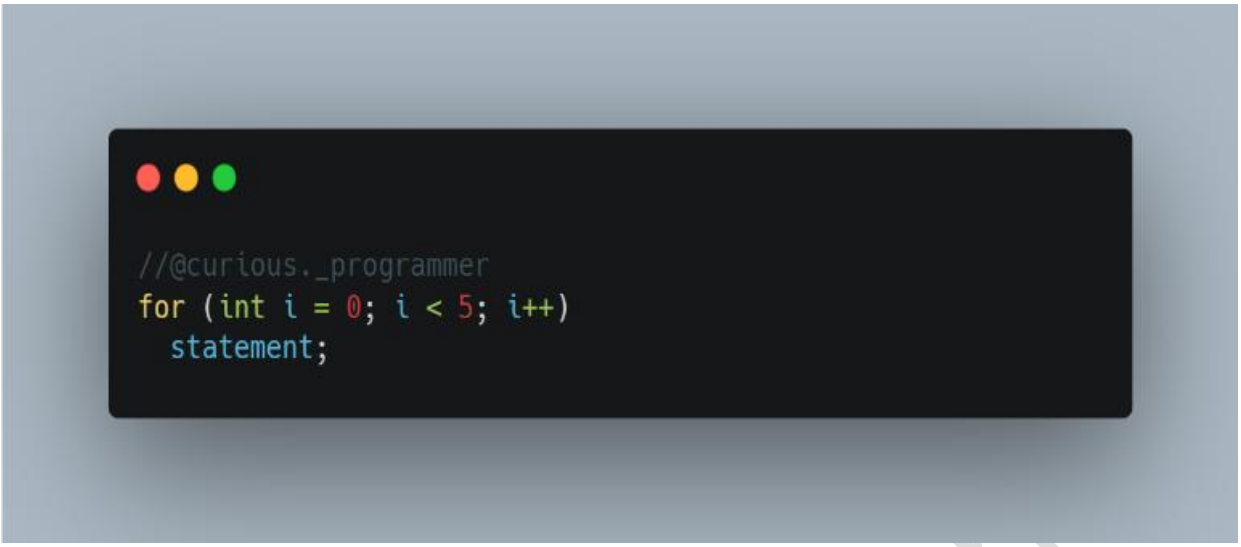
Output Clear

```
java -cp /tmp/iU6AUtxgna Main
Size: Large
```

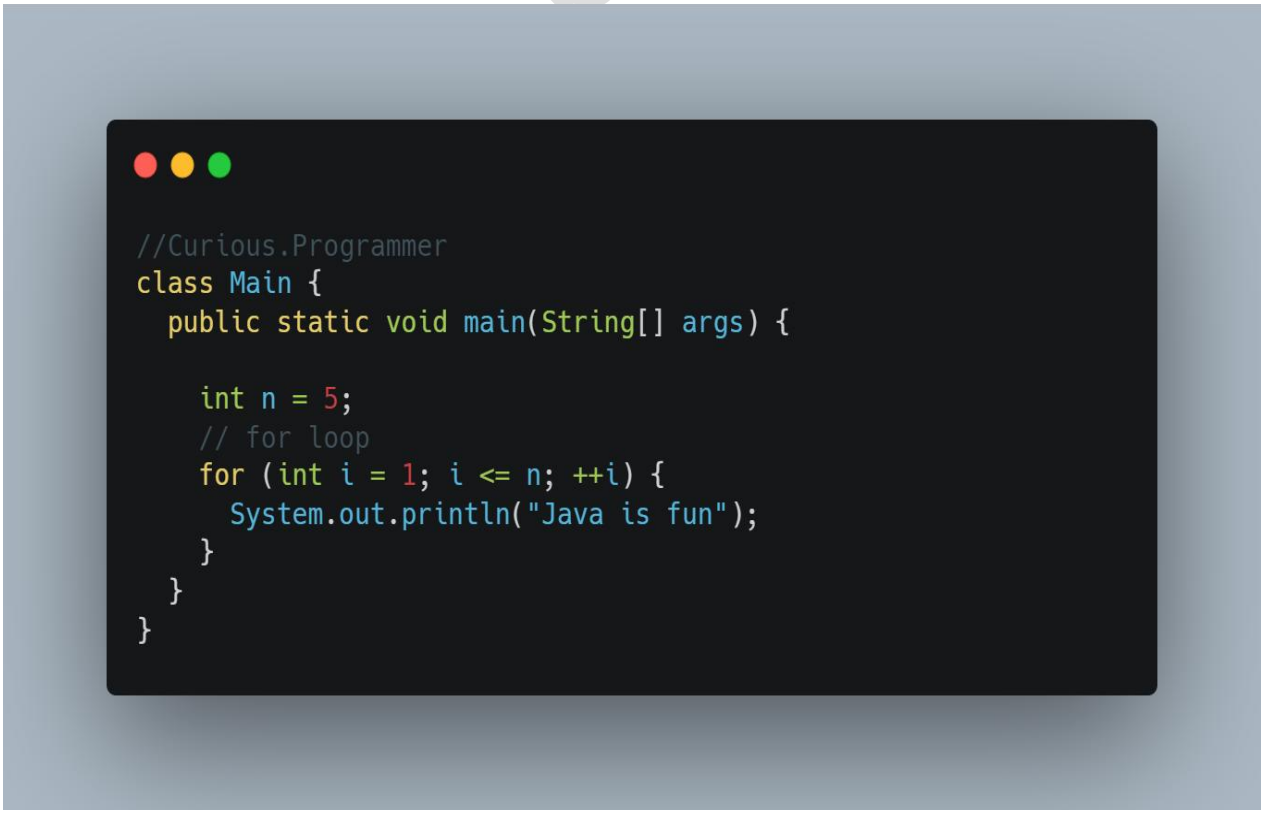
For Loops

For loops are useful when we know ahead of time how many times we want to repeat something. We declare a loop variable (or loop counter) and in each iteration we increment it until we reach the number of times we want to execute some code.



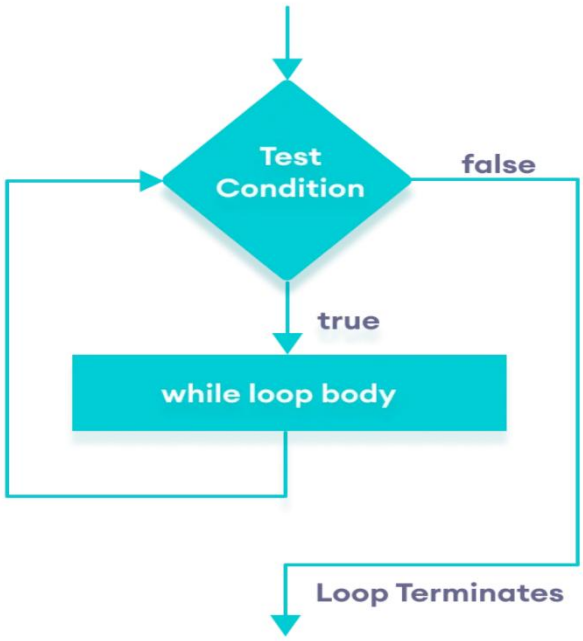


Here, The initial Expression initializes and/or declares variables and executes only once. The condition is evaluated. If the condition is true, the body of the for loop is executed. The update Expression updates the value of initial Expression. The condition is evaluated again. The process continues until the condition is false.



While Loops

While loops are useful when we don't know ahead of time how many times we want to repeat something. This may be dependent on the values at run-time (e.g. what the user enters).
We use the **break** statement to jump out of a loop.



@Curious.Programmer

```
//@curious._programmer
while (someCondition) {
…
if (someCondition) break;
}
```

```java
// Program to display numbers from 1 to 5
//Curious.Programmer

class Main {
  public static void main(String[] args) {

    // declare variables
    int i = 1, n = 5;

    // while loop from 1 to 5
    while(i <= n) {
      System.out.println(i);
      i++;
    }
  }
}
```

Output Clear

```
java -cp /tmp/K5WJ6B42D3 Main
1
2
3
4
5
```

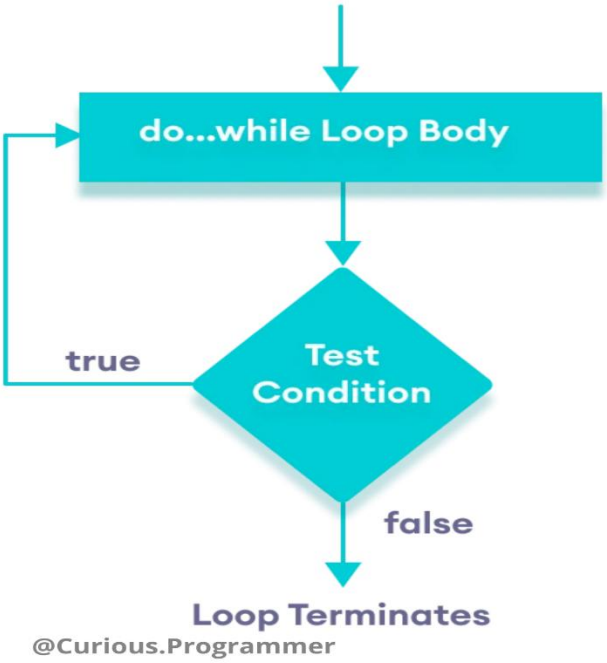
Do..While Loops

Do..While loops are very similar to **while** loops but they executed at least once. Incontrast, a while loop may never get executed if the condition is initially false.

Here,

- The body of the loop is executed at first. Then the textExpression is evaluated.
- If the textExpression evaluates to true, the body of the loop inside the do statement is executed again.
- The textExpression is evaluated once again.

- If the textExpression evaluates to true, the body of the loop inside the do statement is executed again.
- This process continues until the textExpression evaluates to false. Then the loop stops.



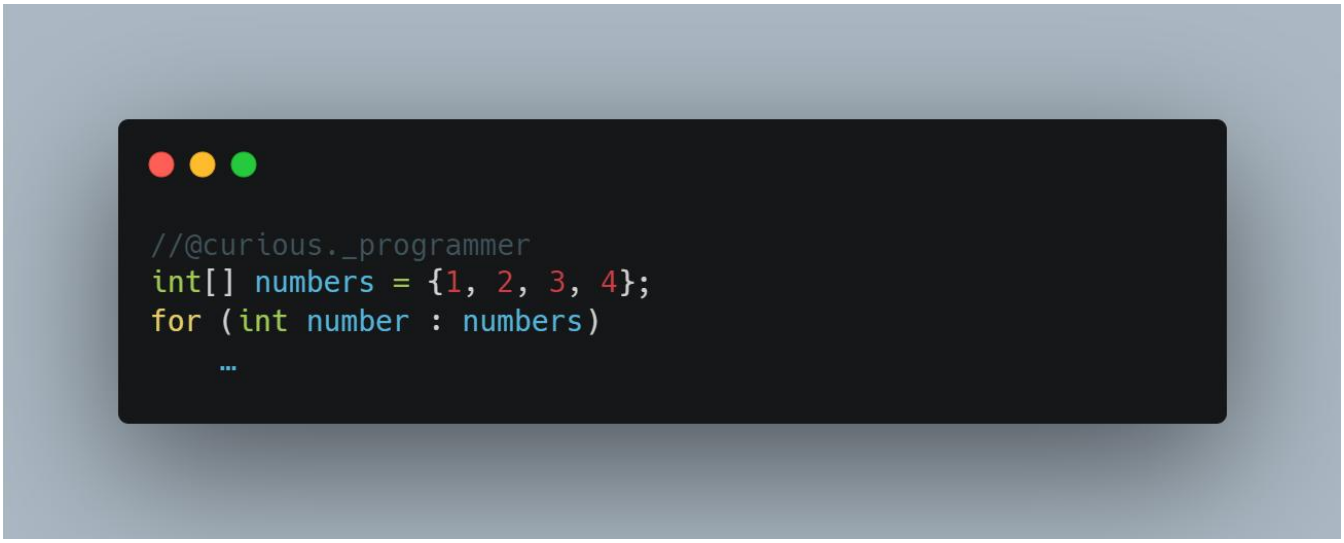

```java
// Java Program to display numbers from 1 to 5
//Curious.Programmer
import java.util.Scanner;

// Program to find the sum of natural numbers from 1 to 100.

class Main {
  public static void main(String[] args) {

    int i = 1, n = 5;

    // do...while loop from 1 to 5
    do {
      System.out.println(i);
      i++;
    } while(i <= n);
  }
}
```

```
Output                                                    Clear
java -cp /tmp/oZHLIXuoBn Main
1
2
3
4
5
```

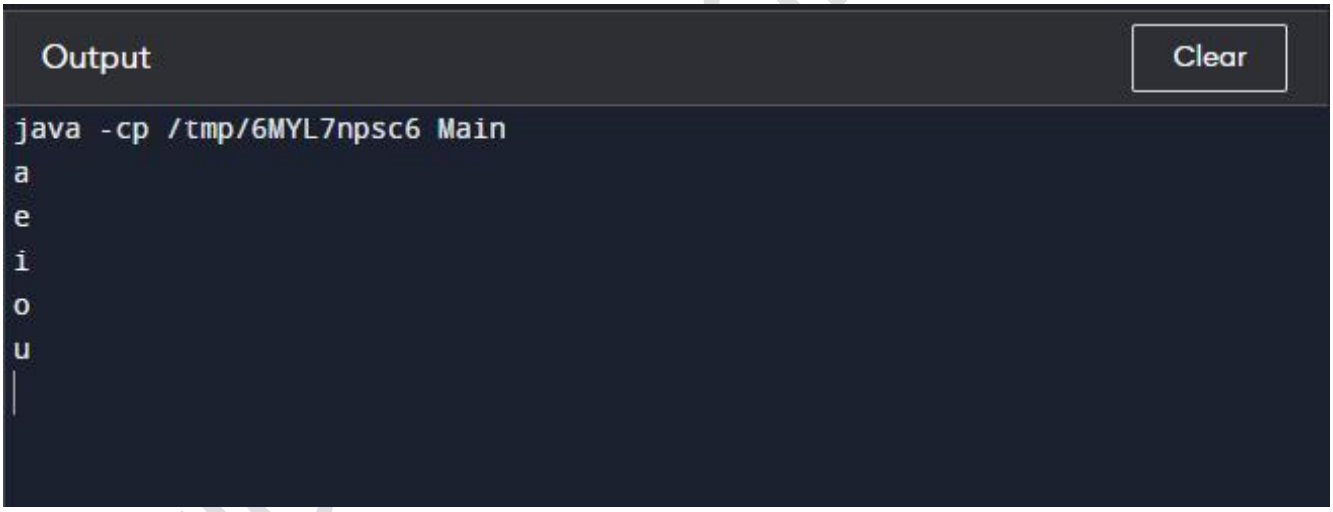
For-each Loops

In Java, the for-each loop is used to iterate through elements of arrays and collections (like ArrayList). It is also known as the enhanced for loop.

For-each loops are useful for iterating over an array or a collection.



```java
//Curious._Programmer
class Main {
 public static void main(String[] args) {

   char[] vowels = {'a', 'e', 'i', 'o', 'u'};

   // iterating through an array using the for-each loop
   for (char item: vowels) {
     System.out.println(item);
   }
 }
}
```



Output

Clear

```
java -cp /tmp/6MYL7npsc6 Main
a
e
i
o
u
```

Java AWT

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Component Class

All the elements like the button, text fields, scroll bars, etc. are called components. In Java AWT, there are classes for each component.

In order to place every component in a particular position on a screen, we need to add them to a container.

Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as **Frame, Dialog** and **Panel**.

It is basically a screen where the where the components are placed at their specific locations. Thus it contains and controls the

layout of components.

There are four types of containers in Java AWT -

- Window
- Panel
- Frame
- Dialog

Consider the following simple example of Java AWT in which we have shown an awt component button by setting its placement and window frame size.

```
//Curious._Programmer
import java.awt.*;
public class AwtProgram1 {
public AwtProgram1()
    {
Frame f = new Frame();
        Button btn=new Button("Hello World");
        btn.setBounds(80, 80, 100, 50);
        f.add(btn);          //adding a new Button.
        f.setSize(300, 250);         //setting size.
        f.setTitle("JavaTPoint");  //setting title.
        f.setLayout(null);   //set default layout for frame.
        f.setVisible(true);           //set frame visibility true.
    }

public static void main(String[] args) {
// TODO Auto-generated method stub
        AwtProgram1 awt = new AwtProgram1();   //creating a frame.
    }
}
```

Output:

- # Java Swing

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and Lightweight components.
The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

```java
//Curious._Programmer
import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame

JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height

f.add(b);//adding button in JFrame

f.setSize(400,500);//400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
```

- ## Difference Between AWT and Swing

AWT	SWING
• Platform Dependent	• Platform Independent
• Does not follow MVC	• Follows MVC
• Lesser Components	• More powerful components
• Does not support pluggable look and feel	• Supports pluggable look and feel
• Heavyweight	• Lightweight