JAVA BASICS

The team

01- AARUSHI SINGH

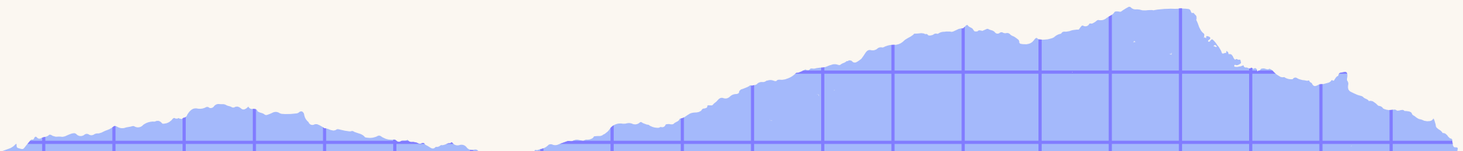
02- AASTHA SAXENA

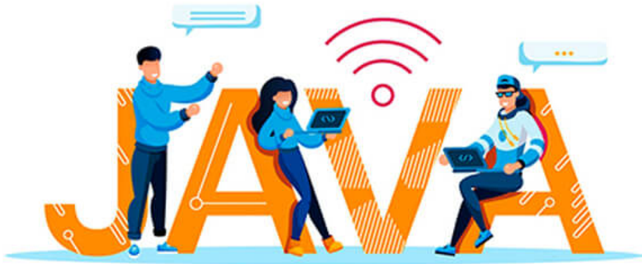
03- AKANCHHA SINGH

04- AMBIKA SOLANKI

What is Java?

Java is an object oriented programming language that is used in a distributed environment on the Internet. It is a high-level language that is easy to read and understand . Java is popularly used in console ,GUI ,web and mobile applications, game development and also to make embedded systems.

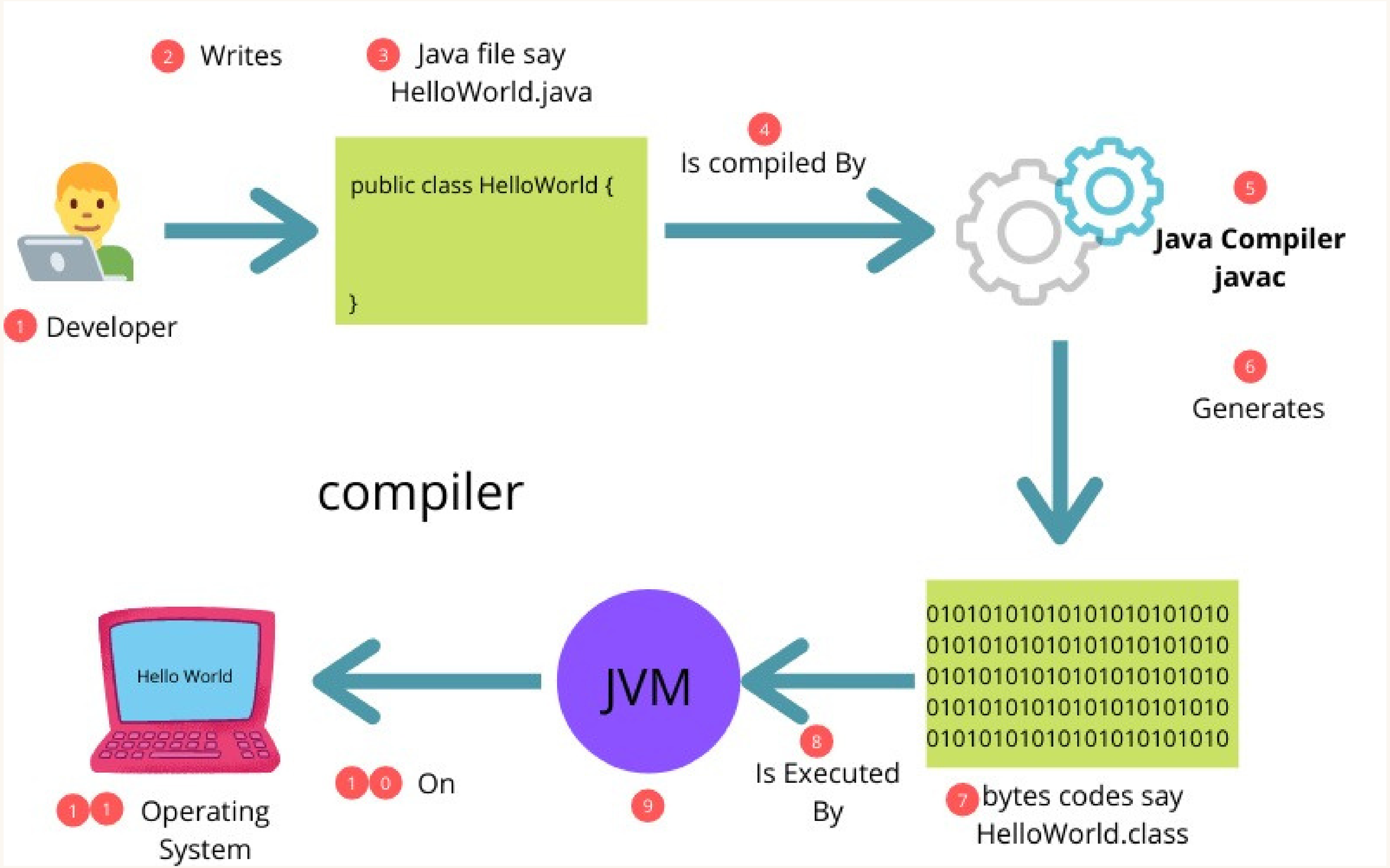




History of JAVA

Java is a computer based programming language invented by James Gosling and Sun microsystems in 1991. He has a single motto while creating the language, It was “Write Once, Run Anywhere” .

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.

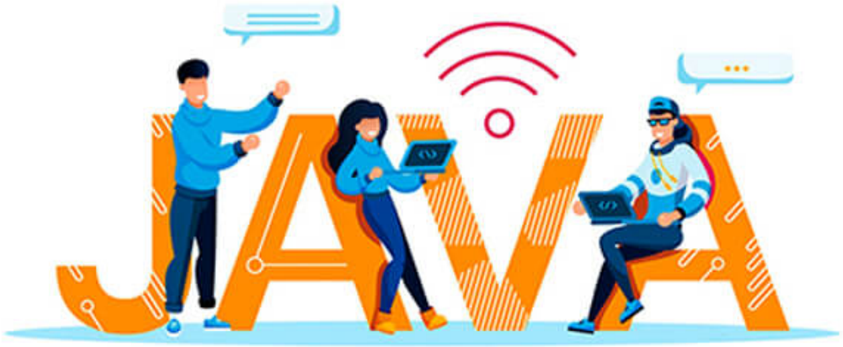

2 Writes
3 Java file say HelloWorld.java
public class HelloWorld {
}
4 Is compiled By
5 Java Compiler javac
1 Developer
6 Generates
compiler
01010101010101010101010
01010101010101010101010
01010101010101010101010
01010101010101010101010
01010101010101010101010
7 bytes codes say HelloWorld.class
8 Is Executed By
JVM
9
10 On
Hello World
11 Operating System

Features of JAVA

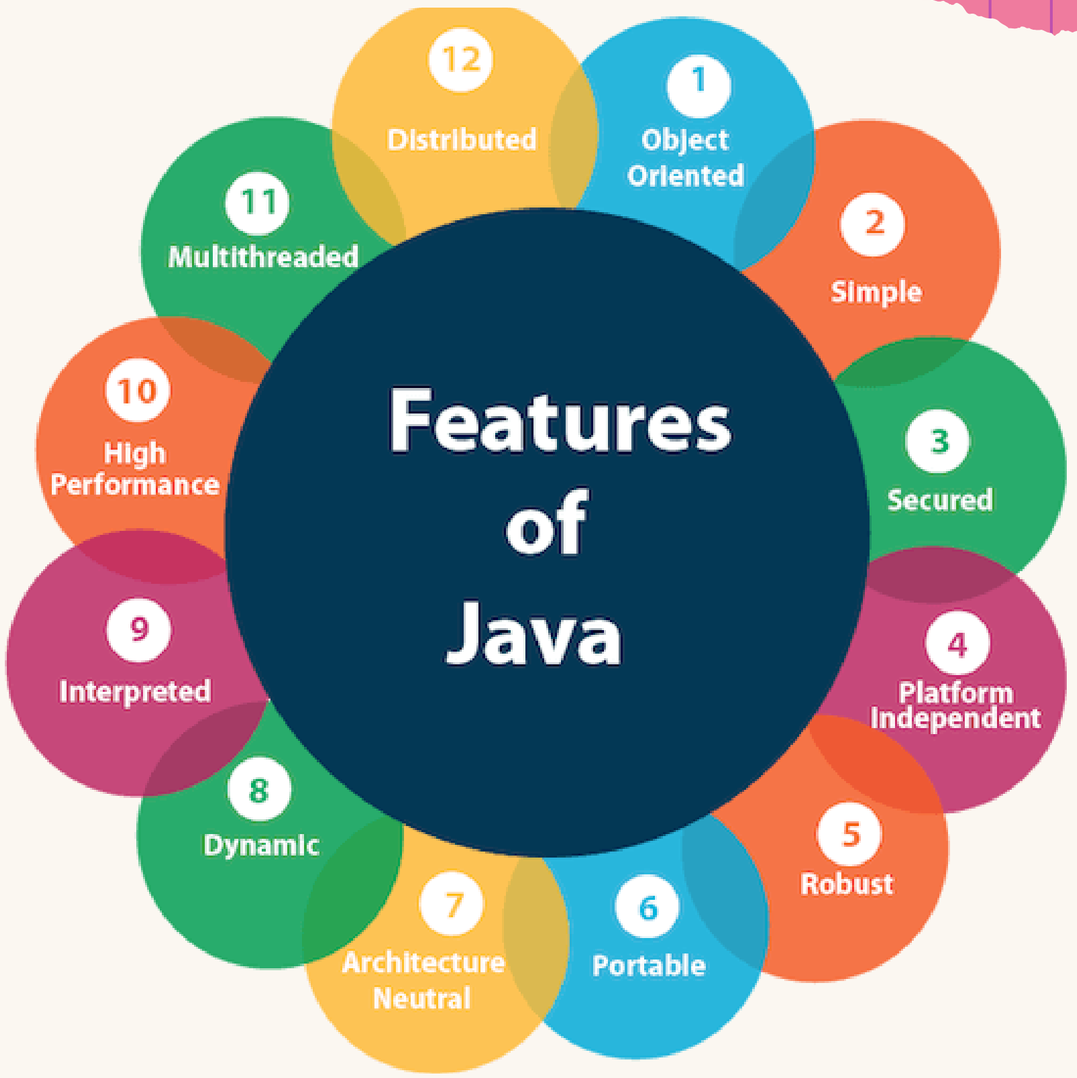
The primary objective of Java programming language creation was to make it portable, simple and secure programming language.

A list of the most important features of the Java language is given below.

- Simple
- Object-Oriented
- Portable
- Platform independent
- Secured



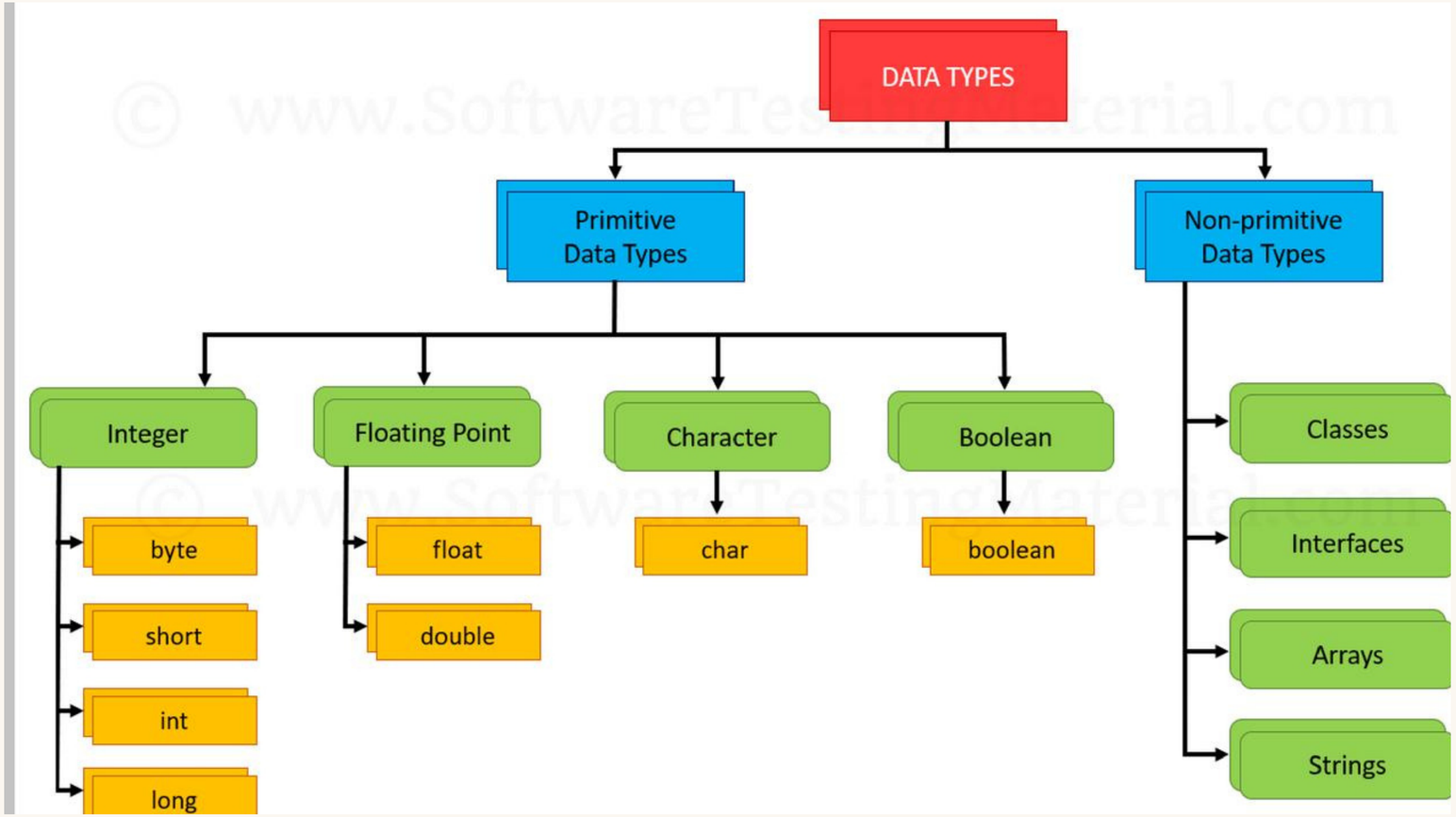
- Robust
- Architecture neutral
- Interpreted
- High Performance
- Multithreaded
- Distributed
- Dynamic



Java Data Types

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.For Example:
int speed;
Here, speed is a variable, and the data type of the variable is int

Two Types of Data types



Important Points

INTEGERS

- Java defines four integer types: byte, short, int, and long.
- All of these are signed, positive and negative values.

FLOATING-POINT TYPES

- Floating-point numbers, also known as real numbers, are used when evaluating expressions that require fractional precision.
- There are two kinds of floating-point types, float and double, which represent single- and double-precision numbers, respectively.

CHARACTERS

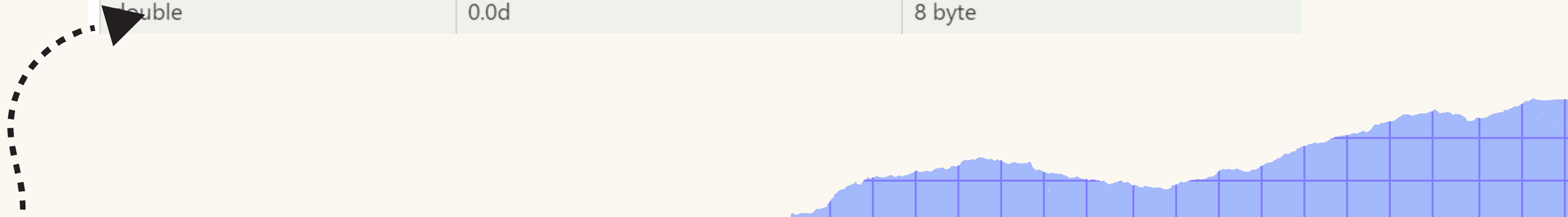
- The data type used to store characters is char.
- Java uses Unicode (Universal Character Encoding Standard)to represent characters. Unicode defines a fully international character set that can represent all of the characters found in all human languages
- There are no negative chars.

BOOLEAN

- It can have only one of two possible values, true or false. This is the type returned by all relational operators, such as a < b.
- Boolean is also the type required by the conditional expressions that govern the control statements such as if and for

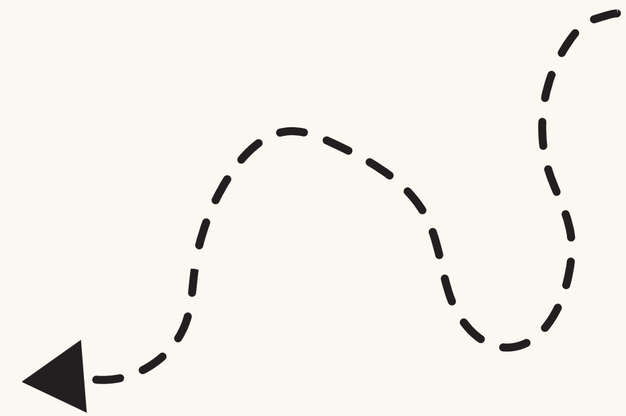
Primitive Data Type

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte



The width and ranges of these types vary widely, as shown in this table:

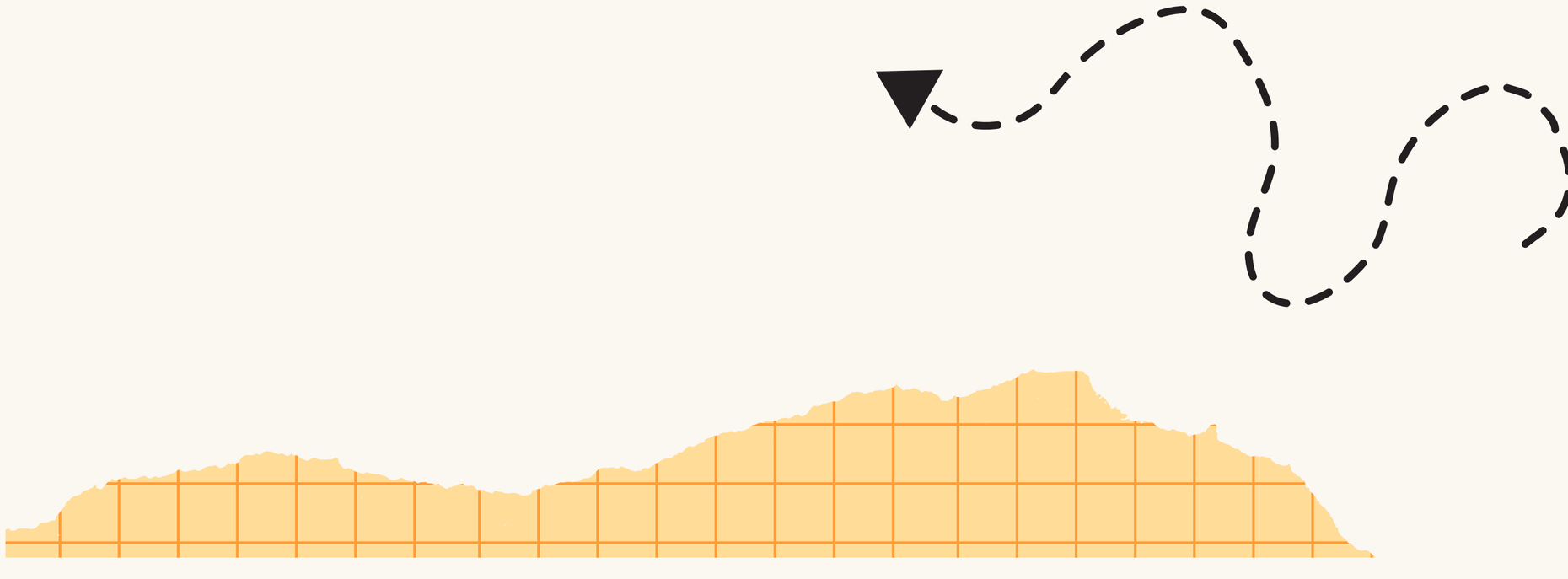
Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535
boolean	1	true or false



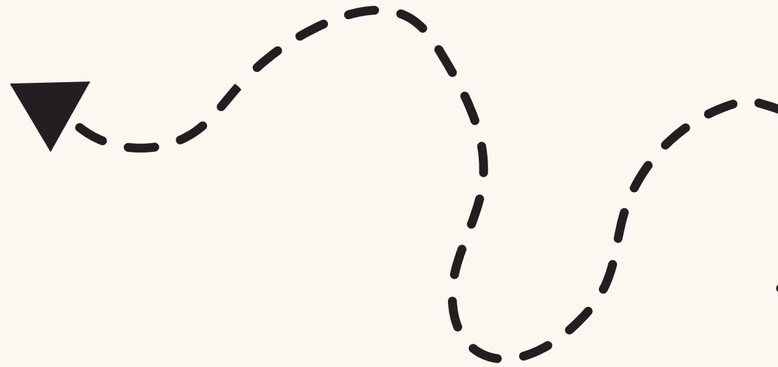
PROGRAMME

```
class GFG {

// Main driver method
    public static void main(String args[])
    {
        char a = 'G';
        int i = 89;
        // use byte and short
        // if memory is a constraint
        byte b = 4;
        // this will give error as number is
        // larger than byte range
        // byte b1 = 7888888955;
        short s = 56;
        // this will give error as number is
        // larger than short range
        // short s1 = 87878787878;
```



```
 double d = 4.355453532;
 float f = 4.7333434f
 // need to hold big range of numbers then we need
 // this data type
 long l = 12121;
System. out. println("char: " + a);
 System. out .println("integer: " + i);
 System. out. println("byte: " + b);
 System. out .println("short: " + s);
 System . out. println("float: " + f);
 System. out . println("double: " + d);
 System. out . println("long: " + l);
 }
}
```



Non-Primitive Data Types

Whenever a non-primitive data type is defined, it refers a memory location where the data is stored in heap memory i.e., it refers to the memory location where an object is placed. Therefore, a non-primitive data type variable is also called referenced data type or simply object reference variable.

.1.Class and objects:

A class in Java is a user defined data type i.e. it is created by the user. It acts a template to the data which consists of member variables and methods.

2. String:

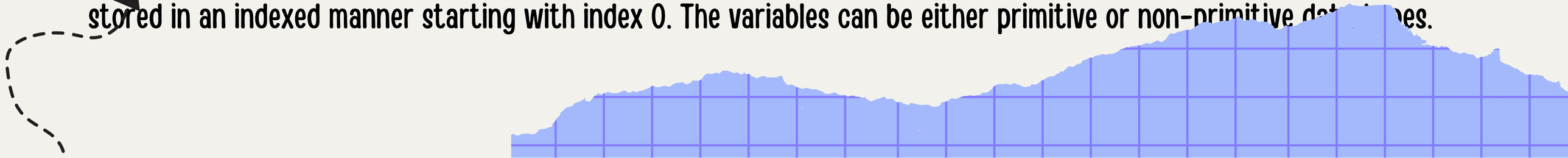
A string represents a sequence of characters for example "Javapoint", "Hello world", etc. String is the class of Java.

One of the ways to create a string and store a value in it is shown below:

String str = "You're the best";

3.Array:

An array is a data type which can store multiple homogenous variables i.e., variables of same type in a sequence. They are stored in an indexed manner starting with index 0. The variables can be either primitive or non-primitive data types.

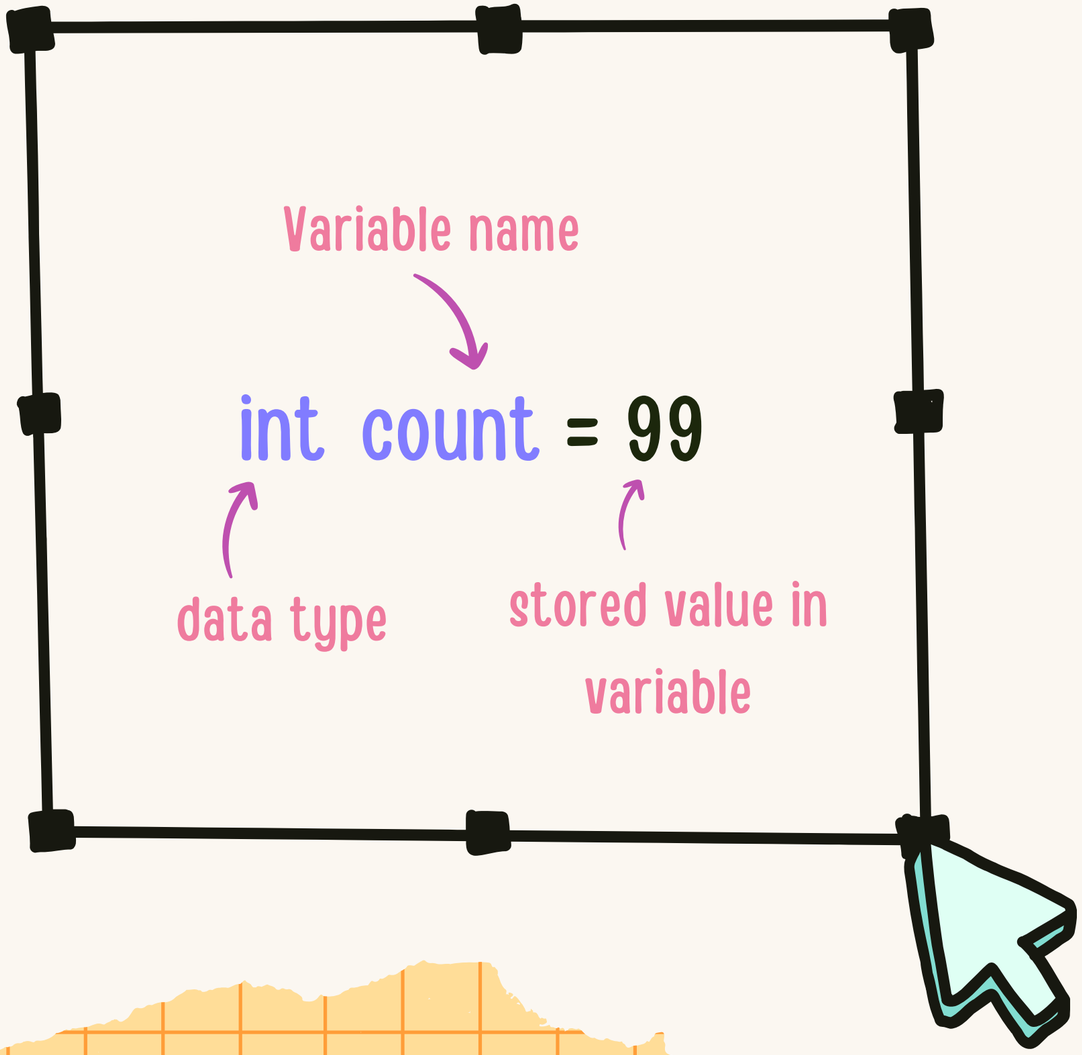


Following example shows how to declare array of primitive data type int:

int a[]={33,3,4,5};

4.Interface:

An interface is similar to a class however the only difference is that its methods are abstract by default i.e. they do not have body. An interface has only the final variables and method declarations. It is also called a fully abstract class.



Variables

A variable is like a labeled container in the computer's memory where you can store values or data.

-In java a variable can store a single value type.

-all variables must be declared before they can be used

Variable Declaration & Initialization

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

    int a, b, c;           // declares three ints, a, b, and c.
    int d = 3, e, f = 5; // declares three more ints, initializing
                           // d and f.
    byte z = 22;           // initializes z.
    double pi = 3.14159; // declares an approximation of pi.
    char x = 'x';          // the variable x has the value 'x'.
    boolean y = true;
  }
}
```

DYNAMIC INITIALIZATION

Dynamic initialization in Java allows variables to be initialized using any expression valid at the time of declaration or condition at runtime.

```java
public class Main {
  public static void main(String[] args) {
    // Dynamic initialization based on user input
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number: ");
    int userInput = scanner.nextInt();
    // Dynamic initialization based on calculation
    double a = 3.0, b = 4.0;
    double c = Math.sqrt(a * a + b * b);
    System.out.println("Hypotenuse is " + c);
  }
}
```

The Scope and Lifetime of Variables

- In Java, variables can be declared within any block, which defines a scope.
- Scopes determine the visibility of objects and their lifetime in a program.
- Java primarily defines two major scopes: class scope and method scope.

example of class and method scope

```java
public class Main {
  static int classVariable = 10; // Class scope variable

  public static void main(String[] args) {
     System.out.println("Class scope variable inside: " + classVariable);

     int methodVariable = 20; // Method scope variable
     System.out.println("Method scope variable: " + methodVariable);
  }
}
```

```
Class scope variable inside: 10
Method scope variable: 20
```

VARIABLE VISIBILITY

Variables declared inside a scope are not accessible to code outside that scope. This localization of variables protects them from unauthorized access or modification.

```java
public class Main {
  public static void main(String[] args) {
    int outerVariable = 10; // Outer scope variable

    {
        int innerVariable = 60; // Inner scope variable
        System.out.println("Inner scope variable: " + innerVariable);
        System.out.println("Outer scope variable from inner scope: " + outerVariable);
    }

    System.out.println("Outer scope variable: " + outerVariable);
    // System.out.println("Inner scope variable: " + innerVariable); // Error: innerVariable
not accessible here
  }
}
```

Type Conversion and Casting

Type conversion refers to the process of converting a value from one data type to another.

Casting is a form of explicit type conversion in Java. It allows the programmer to manually convert a value from one data type to another.

example

```java
public class Main {
  public static void main(String[] args) {
      // Implicit type conversion (widening)
      int intValue = 10;
      double doubleValue = intValue; // int to double (implicit)
      System.out.println("Widening conversion: " + doubleValue);

      // Explicit type conversion (narrowing)
      double doubleNumber = 20.5;
      int intNumber = (int) doubleNumber; // double to int (explicit)
      System.out.println("Narrowing conversion: " + intNumber);
  }
}
```

```
Widening conversion: 10.0
Narrowing conversion: 20
```

AUTOMATIC TYPE CONVERSION IN JAVA

```java
public class Main {
  public static void main(String[] args) {
    byte a = 10;
    short b = 20;
    int c = a + b; // Automatic conversion: byte and
short promoted to int
    System.out.println("a+b= "+c); // Prints 30
  }
}
```

```
a+b= 30
```

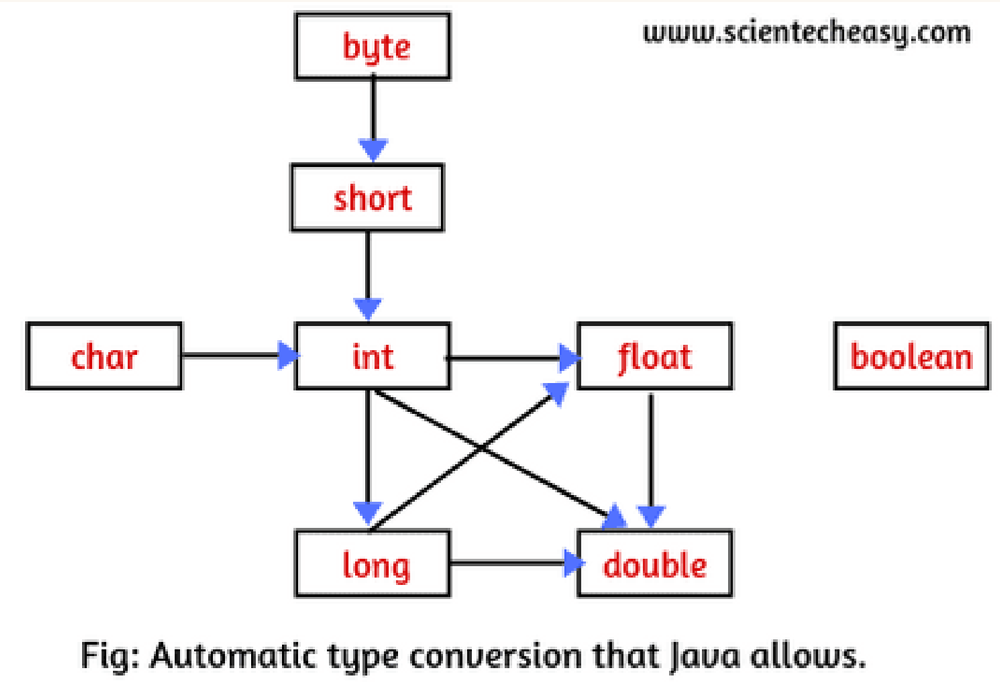


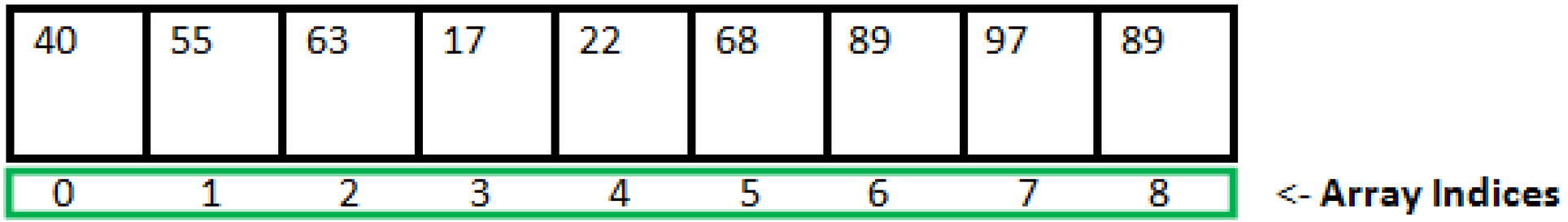
Fig: Automatic type conversion that Java allows.

Arrays

An array is a group of like-typed variables that are referred to by a common name. Arrays of any type can be created and may have one or more dimensions. A specific element in an array is accessed by its index. Arrays offer a convenient means of grouping related information

The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on



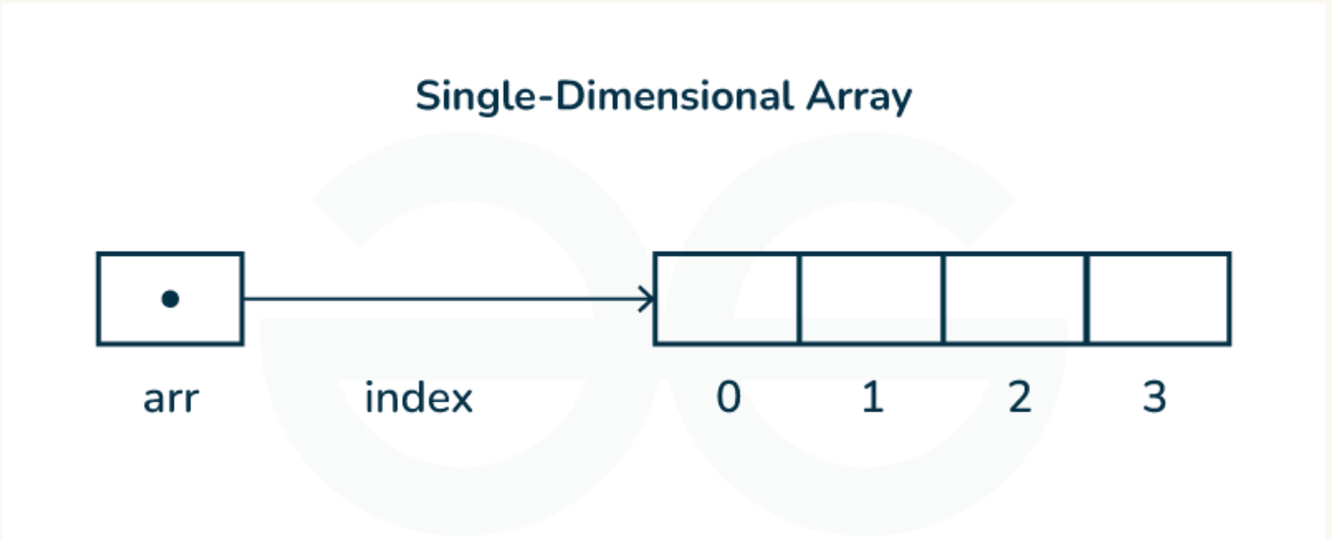
One-Dimensional

One-Dimensional Arrays Aone-dimensional array is, essentially, a list of like-typed variables. To create an array, you first must create an array variable of the desired type. The general form of a one-dimensional array declaration is

type var-name[ ];

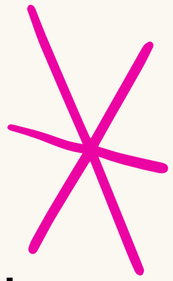
For example, the following declares an array named month_days with the type “array of int”:

int month_days[];



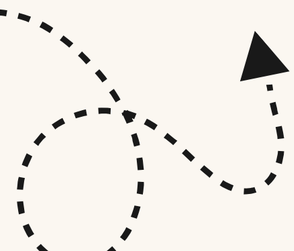
But you need to use new it now to allocate memoryfor arrays. The general form of new as it applies to one-dimensional arrays appears as follows:

array-var = new type[size];



That is, to use new to allocate an array, you must specify the type and number of elements to allocate

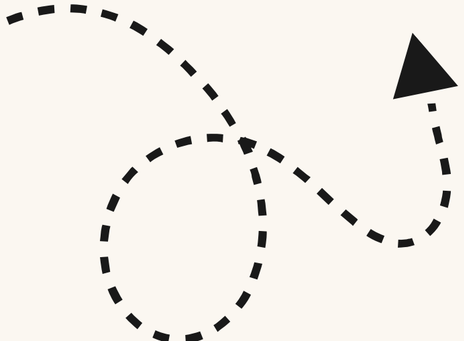
The elements in the array allocated by new will automatically be initialized to zero.



This example allocates a 12-element array of integers and links them to month_days.

month_days = new int[12]

```
class Array {
public static void main(String args[]) {
int month_days[];
month_days = new int[12];
month_days[0] = 31;
month_days[1] = 28;
month_days[2] = 31;
month_days[3] = 30;
month_days[4] = 31;
month_days[5] = 30;
month_days[6] = 31;
month_days[7] = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[11] = 31;
System.out.println("April has " + month_days[3] + " days.");
}
}
```



Example

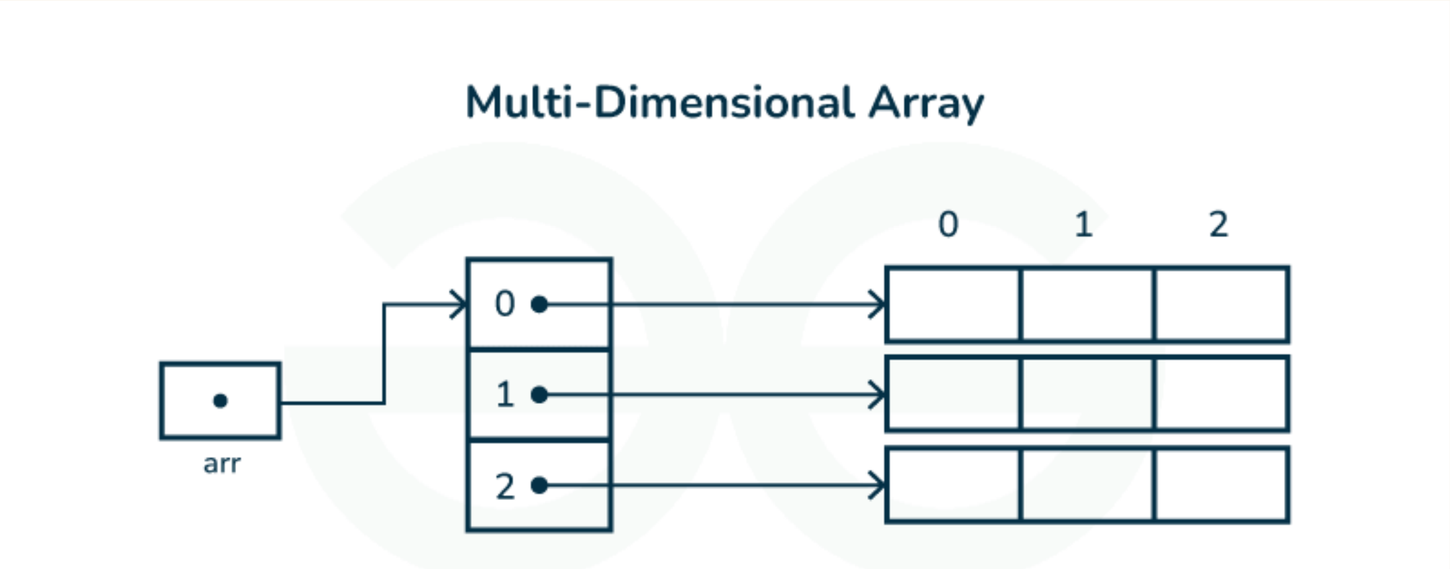
```
 class Average { public static void main(String args[]) {
double nums[] = {10.1, 11.2, 12.3, 13.4, 14.5};
double result = 0;
int i;
 for(i=0; i<5; i++)
result = result + nums[i];
System.out.println("Average is " + result / 5);
 }
 }
```

Multidimensional Arrays

In multidimensional arrays are actually arrays of arrays.These,as you might expect,look and act like regular multidimensional arrays. However, as you will see, there are a couple of subtle differences.

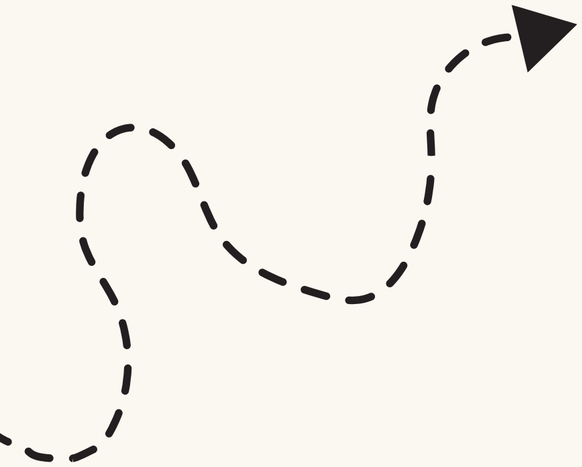
To declare a multidimensional array variable, specify each additional index using another set of square brackets.For example,the following declares a two dimensional array variable called twoD.

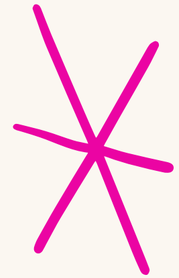
```
int twoD[][] = new int[4][5];
```



Demonstrate a two-dimensional array

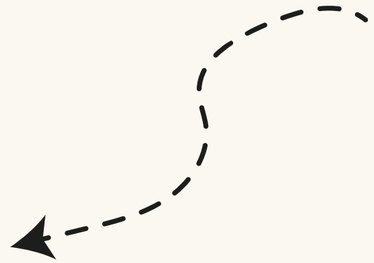
```
class TwoDArray {
public static void main(String args[]) {
int twoD[][]= new int[4][5];
int i, j, k = 0;
for(i=0; i<4; i++)
for(j=0; j<5; j++) {
twoD[i][j] = k;
k++;
 }
for(i=0; i<4; i++) {
for(j=0; j<5; j++)
System.out.print(twoD[i][j] + " ");
System.out.println();
 }
 }
 }
```



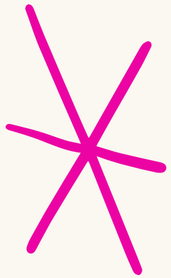


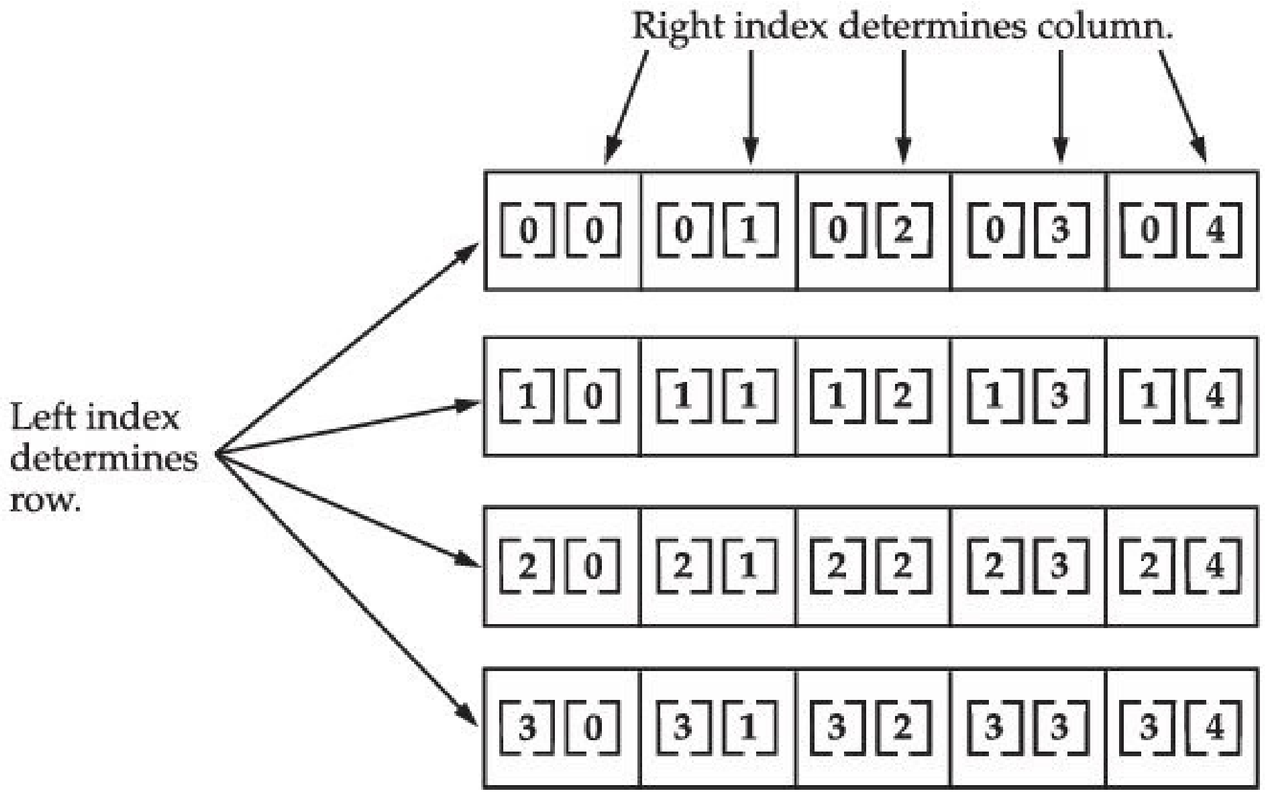
OUTPUT

This program generates the following output:



```
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```





Given: int twoD [ ] [ ] = new int [4] [5] ;

Alternative Array Declaration

Syntax There is a second form that may be used to declare an array:

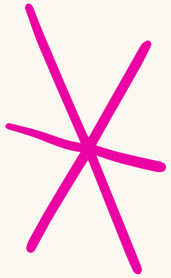
type[ ]var-name;

Here, the square brackets follow the type specifier, and not the name of the array variable.
For example, the following two declarations are equivalent:

```
int al[] = new int[3];
int[] a2 = new int[3];
```

The following declarations are also equivalent:

```
 char twod1[][] = new char[3][4];
char[][] twod2 = new char[3][4];
```



The alternative declaration form is also useful when specifying an array as a return type for a method

THANK YOU!