Java Fundamentals

Agenda

01 Introduction to Core Java

02 Installation and First Program

03 Data Types in Java

04 Fundamentals of Java

Introduction to Core Java

Introduction to Core Java

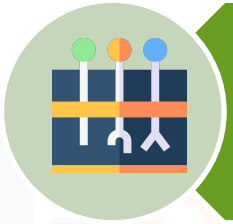
Introduction to Java

Java is a versatile, high-level, object-oriented programming language that was first developed by James Gosling and his team at Sun Microsystems in the mid-1990s. One of the key features that made Java stand out was its platform independence, allowing it to run on any device with a Java Virtual Machine (JVM). It quickly gained popularity due to its simplicity, portability, and security features.

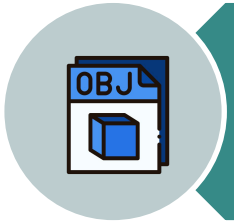
Introduction to Core Java

Key Features of Java

Platform Independence: Java programs can be written once and run on any platform with a compatible JVM, making it a "write once, run anywhere" language.



Multi-threading Support: Java facilitates the creation and management of multiple threads, enabling concurrent execution and improving performance in multi-core systems.



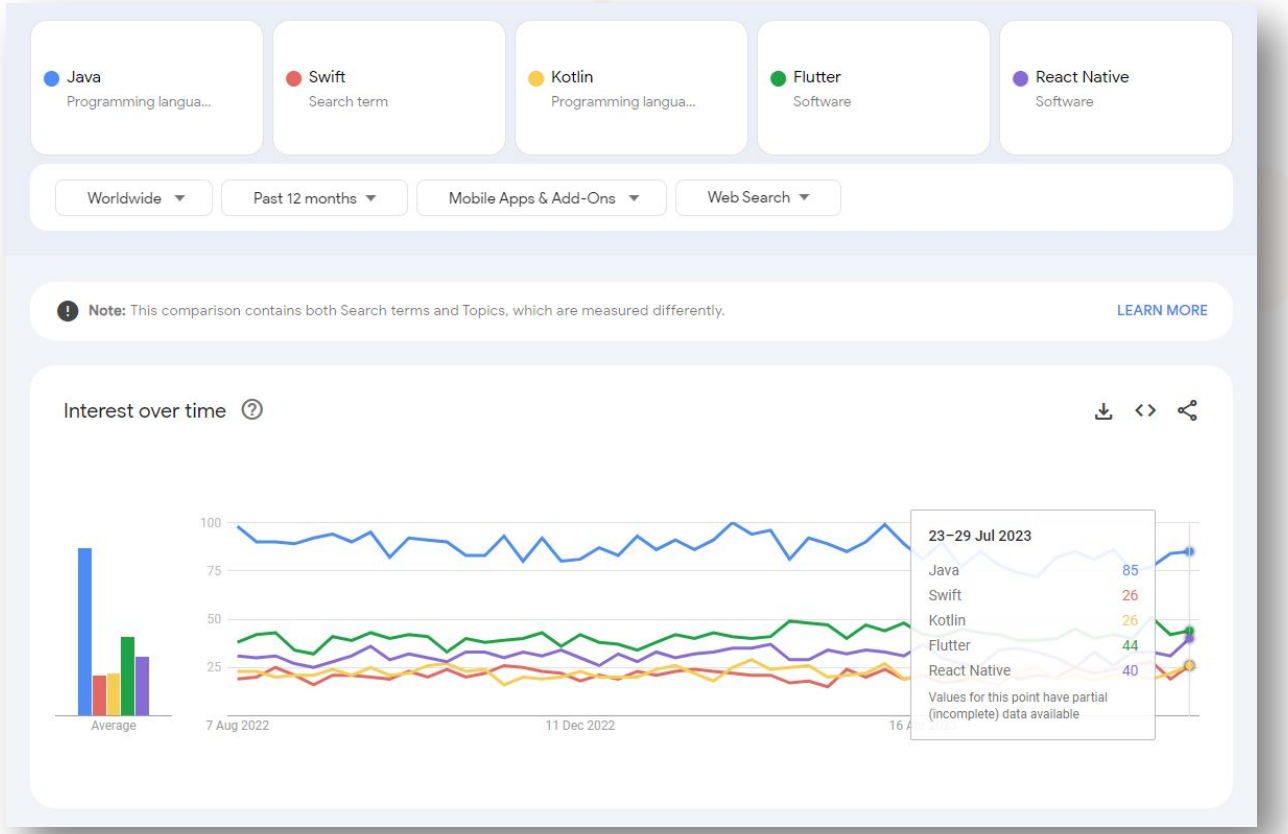
Object-Oriented: Java follows an object-oriented programming paradigm, where data and functions are encapsulated within objects, promoting modularity and code reusability.

Robust Exception Handling: Java has robust exception handling mechanisms, allowing developers to handle errors and exceptions gracefully, enhancing program stability.

Garbage Collection: Java includes automatic memory management through garbage collection, which relieves developers from manual memory allocation and deallocation.

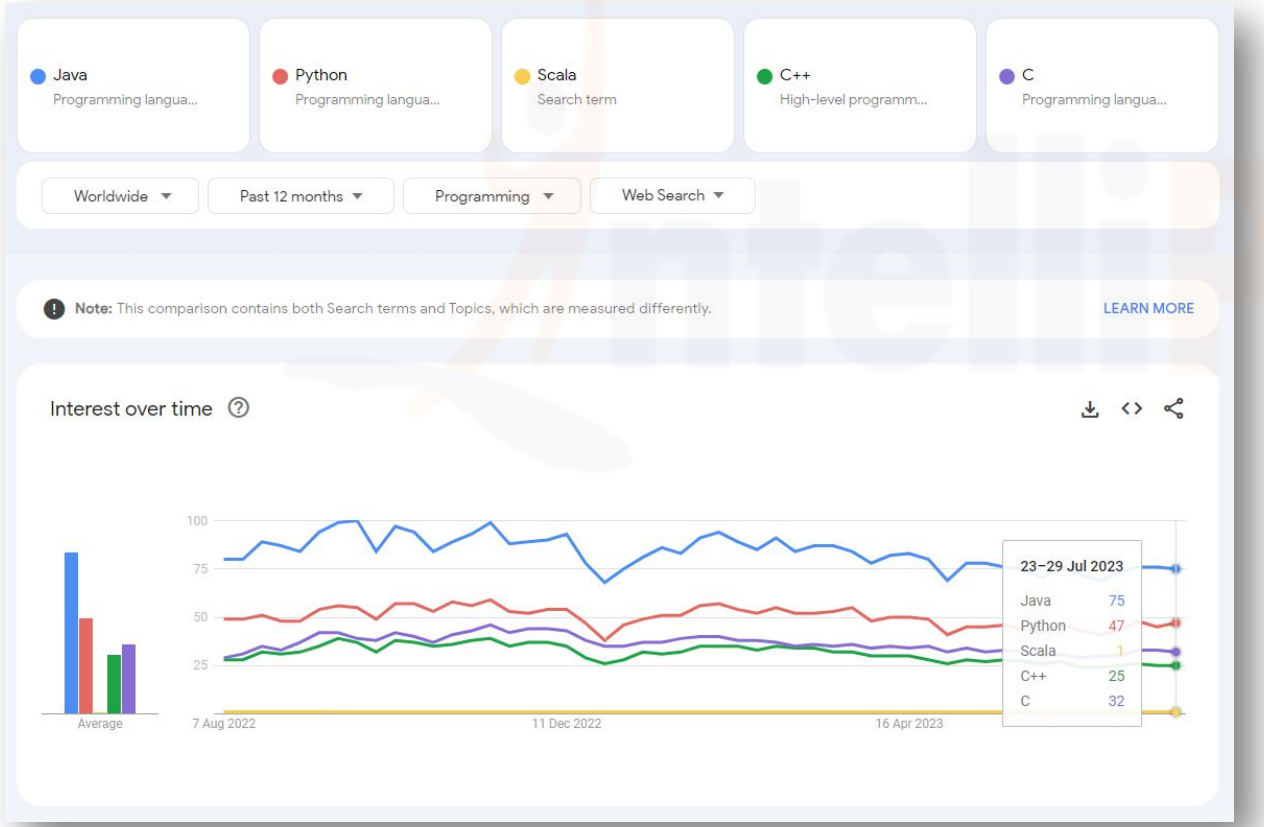
Rich Standard Library: Java offers a vast standard library that includes various utility classes for common programming tasks, simplifying development.

Popularity of Java in App Development



Java is the primary language for Android app development. Android Studio, the official IDE for Android, supports Java and Kotlin as the main programming languages for building Android applications.

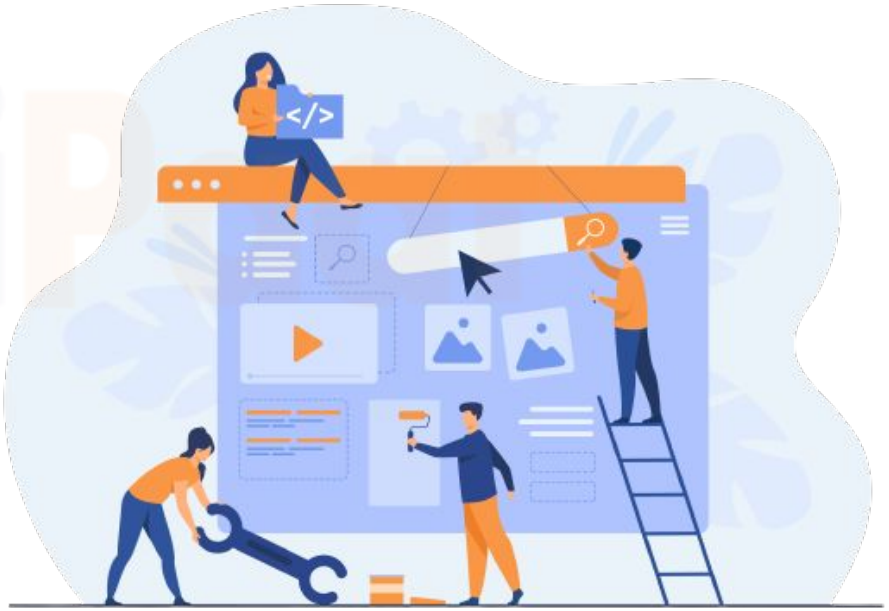
Popularity of Java as Programming Language



Java remains one of the most popular and widely-used programming languages in the software development industry. It has consistently ranked among the top programming languages in various popularity indexes, surveys, and industry reports.

Reasons for Creation

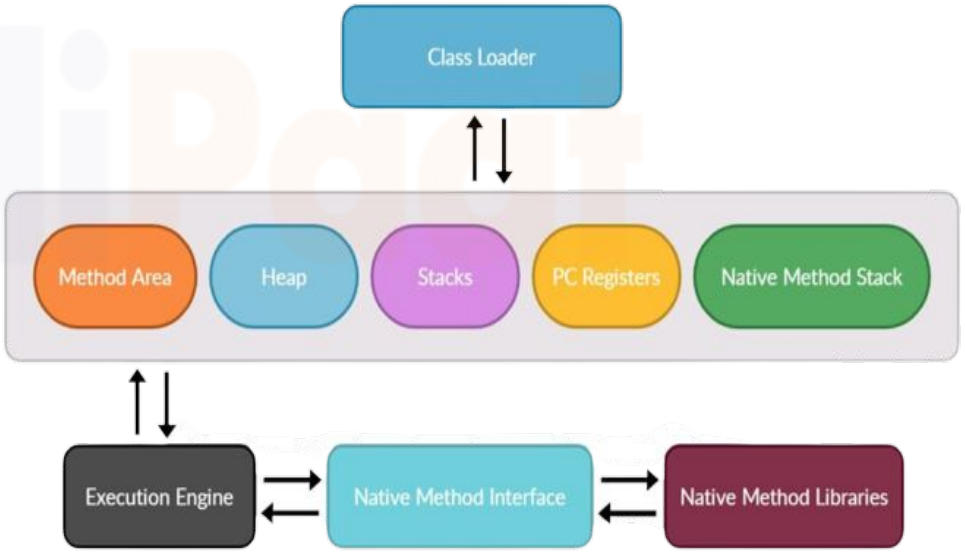
- The team wanted a language that could run on different devices and architectures without the need for recompilation, given the diverse hardware landscape.
- Portability and platform independence were crucial goals, enabling developers to deploy applications easily across a wide range of platforms.
- The original goal was to develop a language for "smart" consumer electronics, where each device could execute the same code regardless of the hardware.



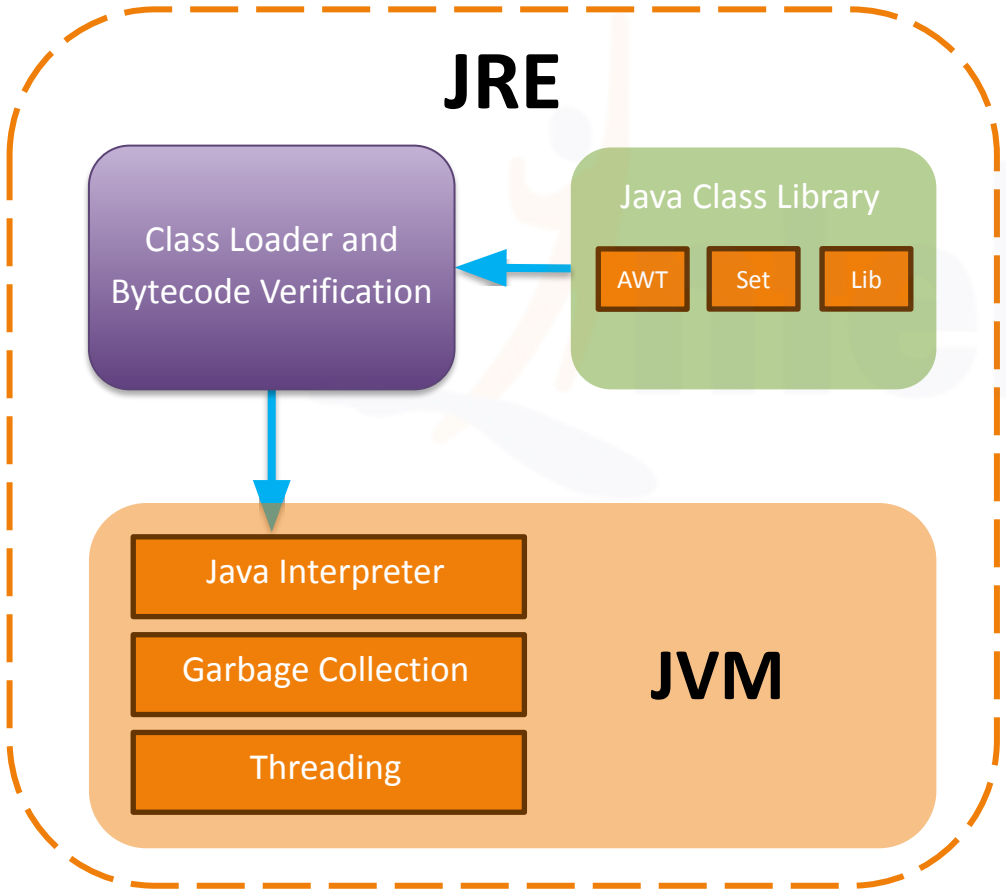
Introduction to Core Java

What is a Java Virtual Machine (JVM)?

- **JVM** is a crucial component of the Java platform. It is a virtual machine that enables **Java bytecode** to be **executed** on any device or **operating system**.
- When Java source code is compiled, it is transformed into **platform-independent** bytecode, which the JVM can interpret and execute.
- The JVM **abstracts** the underlying **hardware**, providing a consistent environment for **Java applications.**



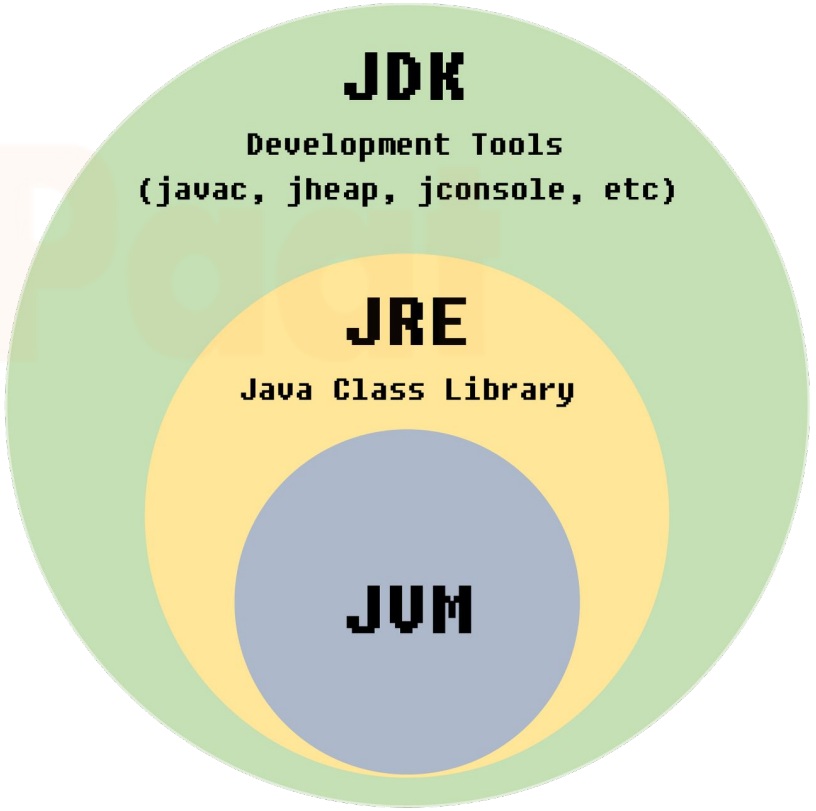
What is a Java Runtime Environment (JRE)?



- JRE is a software package that includes the JVM along with other necessary libraries and files required to run Java applications.
- It provides the runtime environment for executing Java applications without the need for additional development tools.

What is Java Development Kit (JDK)?

- JDK is a software development kit that includes the tools and utilities necessary for Java application development.
- It includes the JRE along with additional tools like the Java Compiler (javac), the Java debugger (jdb), and other utilities.
- JDK allows developers to write, compile, and run Java applications.



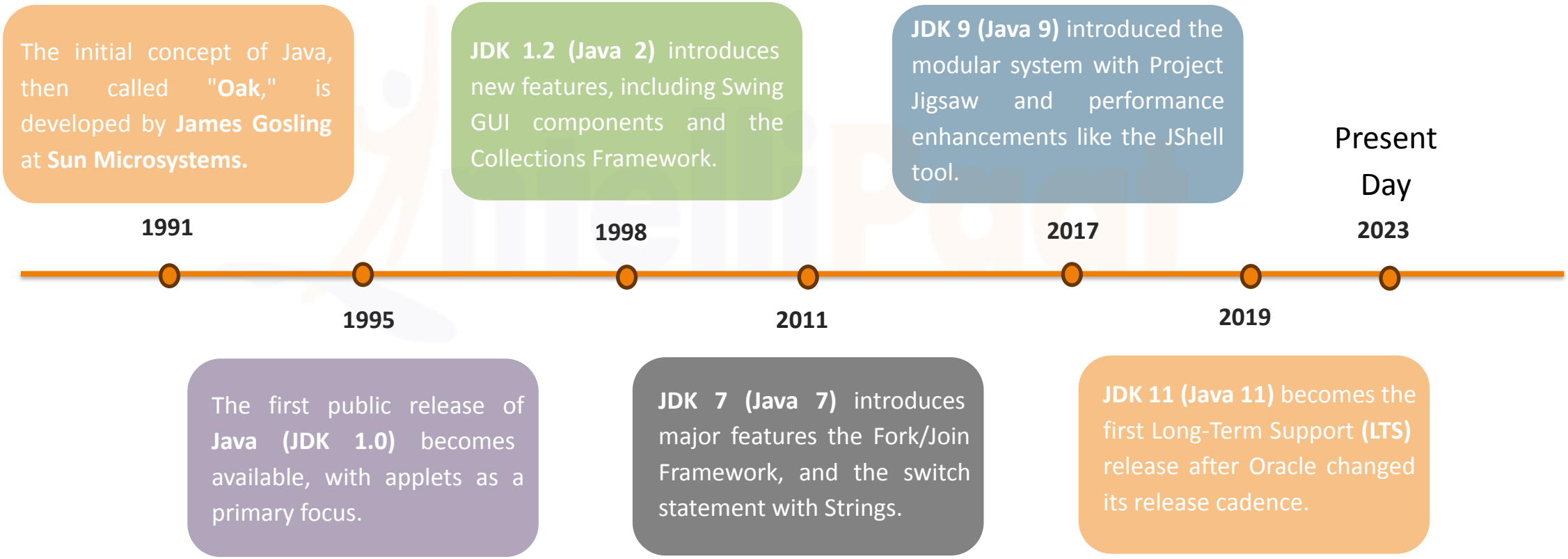
Introduction to Core Java

Real-Life Examples of Java Applications

- Android Apps
- Online Banking Systems
- Customer Relationship Management (CRM)
- Online Banking Portals
- Scientific Applications

Introduction to Core Java

Timeline and Milestones of Java



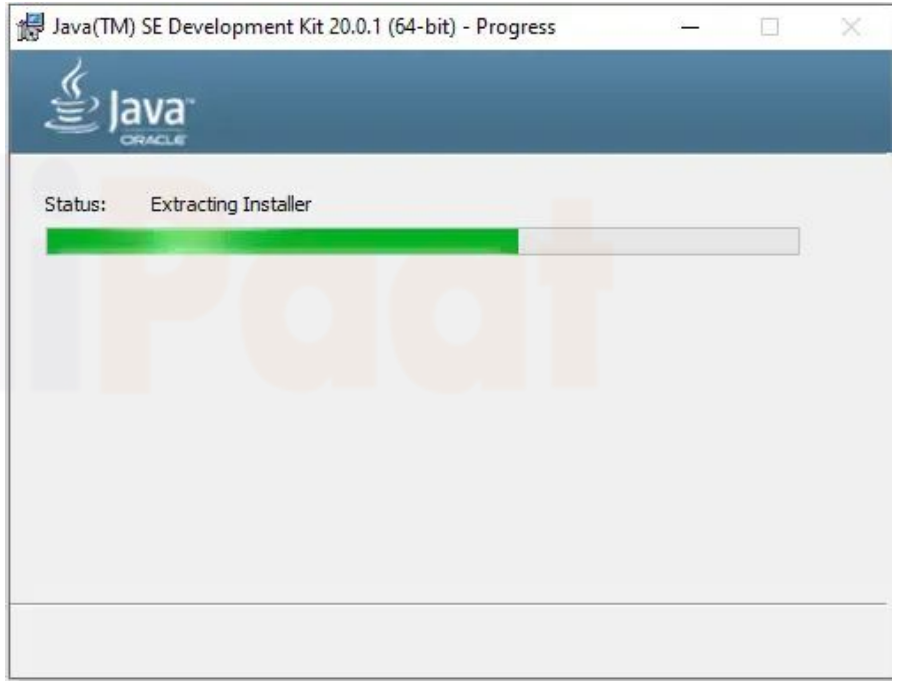
Installation and First Program

Installation and First Program

Setting up Java Environment

Step 01 :Installing JDK

Download and install the latest Java Development Kit (JDK) from Oracle's website or OpenJDK's distribution. Ensure the installation is successful and set the environment variables.



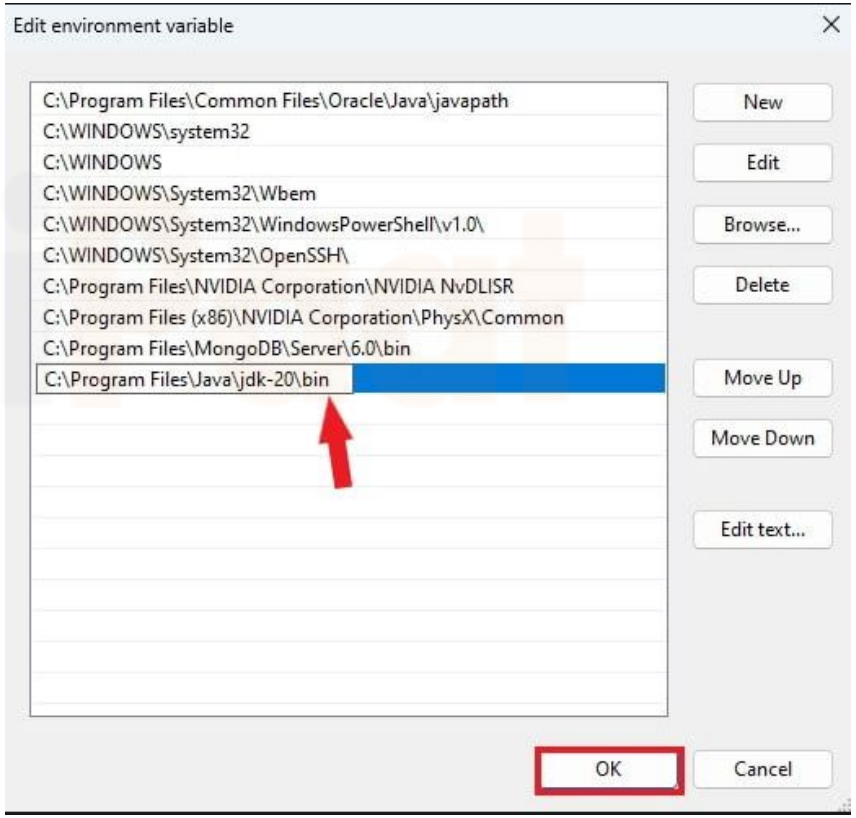
https://www.oracle.com/in/java/technologies/downloads/#jdk20-windows

Installation and First Program

Setting up Java Environment

Step 02 : Set PATH and JAVA_HOME

Update the system PATH variable to include the JDK's "bin" directory, and set the JAVA_HOME variable to the JDK installation directory.



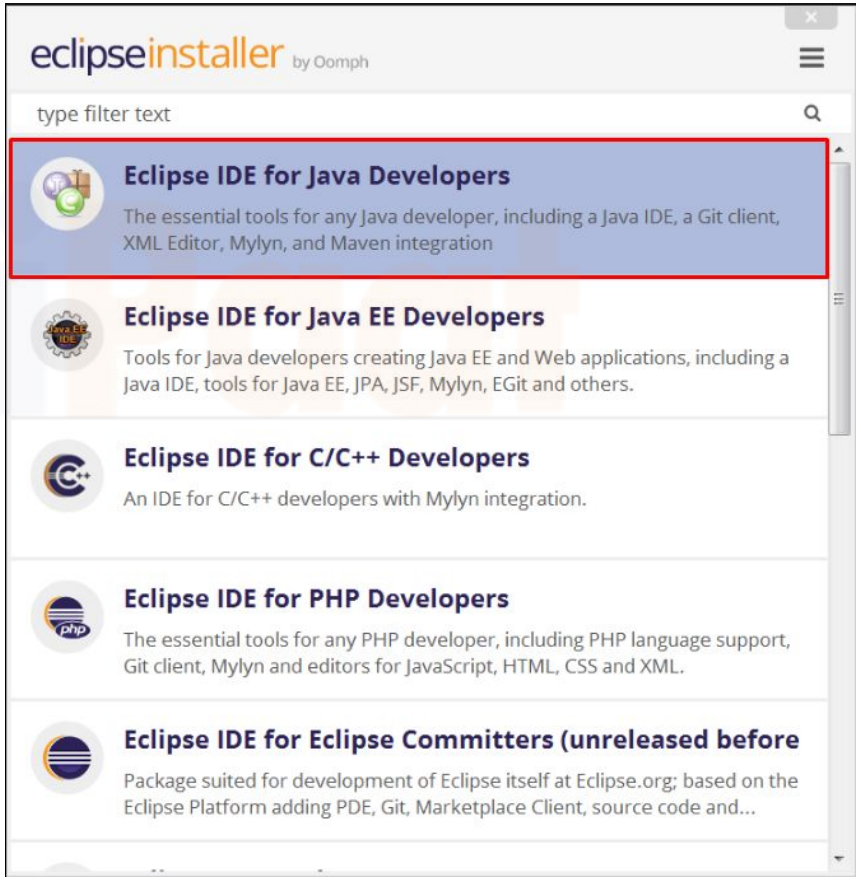
Installation and First Program

Setting up Java Environment

Step 03 : Install Eclipse (IDE)

Consider using popular IDEs like Eclipse, IntelliJ IDEA, or NetBeans to streamline Java development. Download the installer from the link given below :

https://www.eclipse.org/downloads

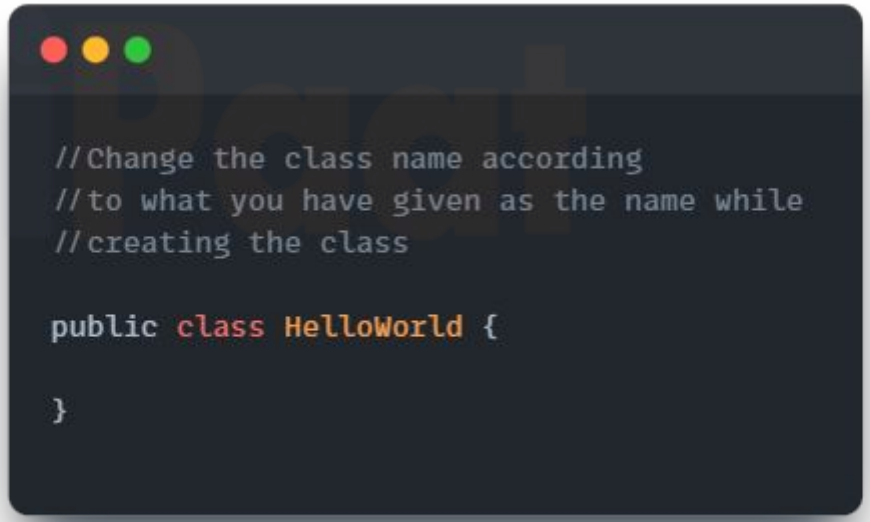


Installation and First Program

Creating the First Java Program

Step 01 : Creating Java Project and Java Class

- Open your **Eclipse IDE** and create a new **Java project**.
- Right-click on the project and choose **"New" -> "Class"** to create a new Java class.
- Give your **class** a meaningful **name.** In this example we have it as **HelloWorld.**



```
//Change the class name according
//to what you have given as the name while
//creating the class

public class HelloWorld {

}
```

Installation and First Program

Creating the First Java Program

Step 02 : Writing Main Method and Print Hello World

- The main method is the entry point of your Java program.
- It has the signature:
  public static void main(String[] args)
- Inside the main method, add a simple **"Hello, World!"** message using the **System.out.println** method.

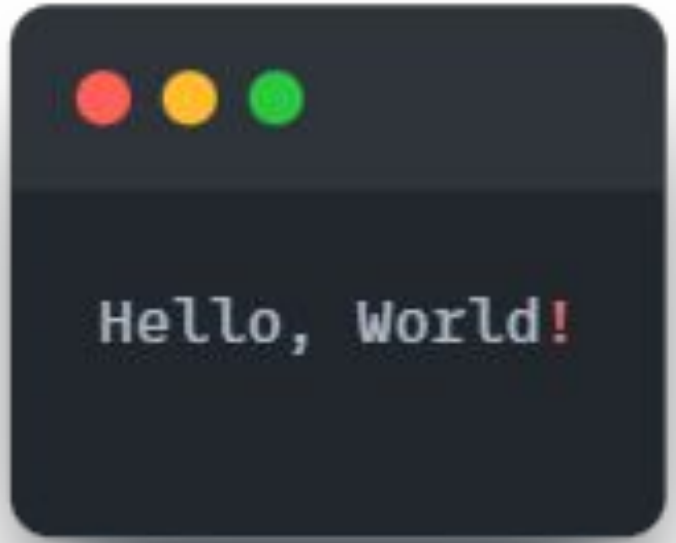
```
//Change the class name according
//to what you have given as the name while
//creating the class

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Installation and First Program

Creating the First Java Program

Step 03 : Saving and Running the Program

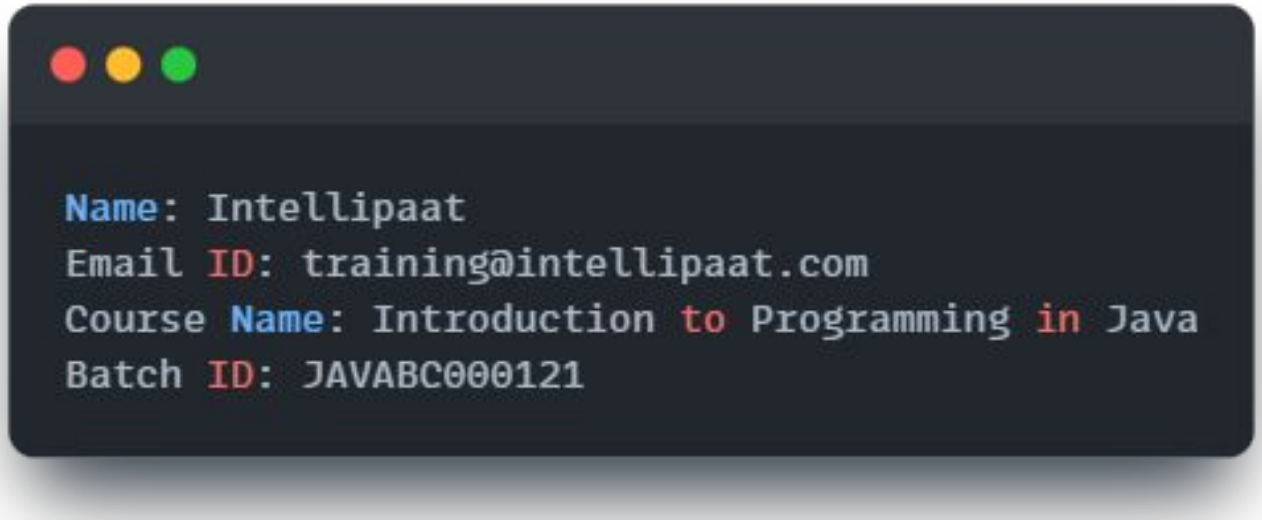
- **Save** your Java file using **CTRL + S** and click on the **"Run"** button to execute your program.
- You should see the output of the program as **Hello World.**



Try Yourself

Write Your First Java Program

- Create a Java program that prints your personal information, including your Name, Email ID, Course Name, and Batch ID on the screen.



Try Yourself

Answer to Your First Java Program

```java
public class IntellipaatPersonalInformation
{
    public static void main(String[] args)
    {
        System.out.println("Name: Intellipaat");
        System.out.println("Email ID: training@intellipaat.com");
        System.out.println("Course Name: Introduction to Programming in Java");
        System.out.println("Batch ID: JAVABC000121");
    }
}
```

Data Types in Java

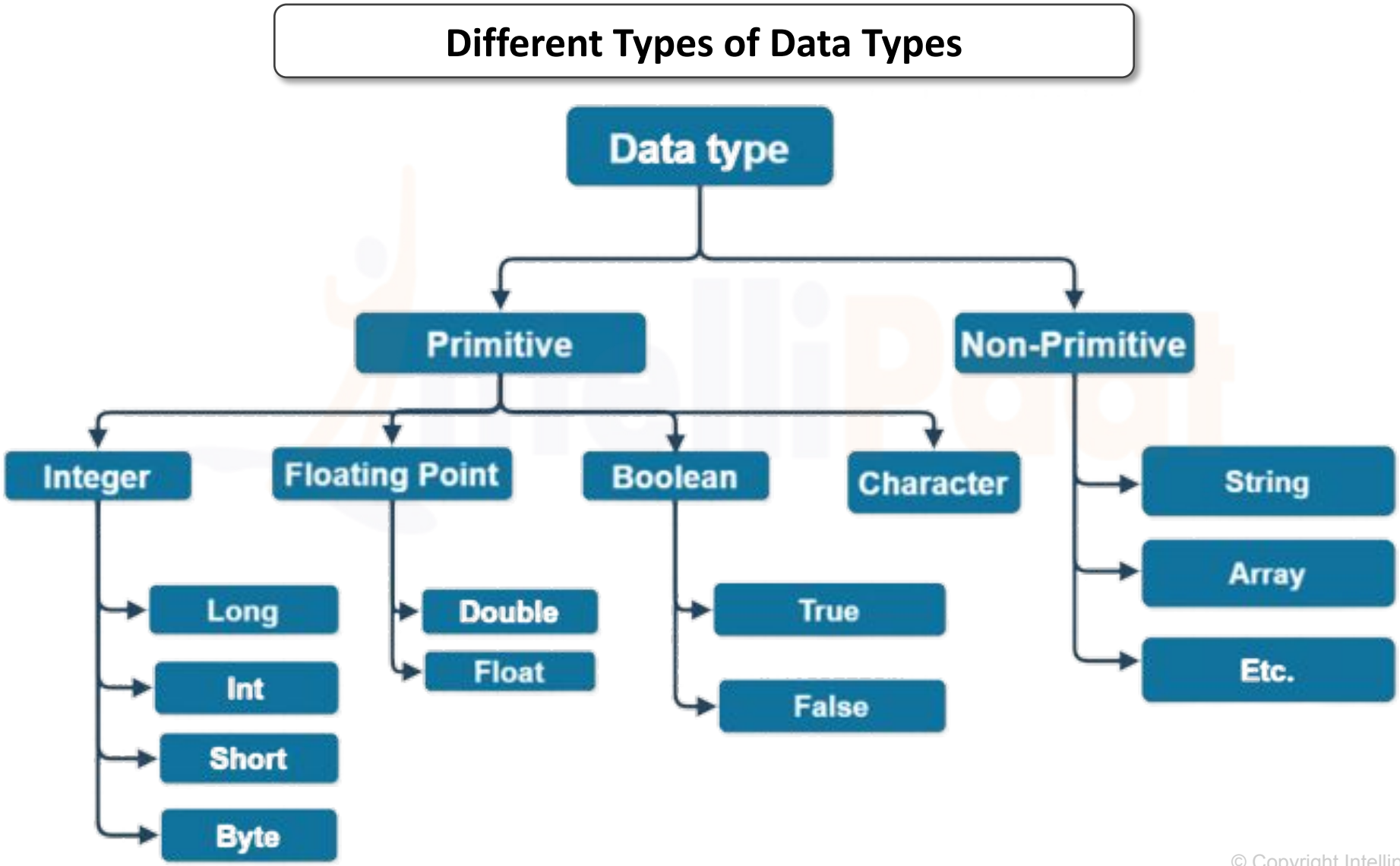
Data Types in Java

What is Data Type?

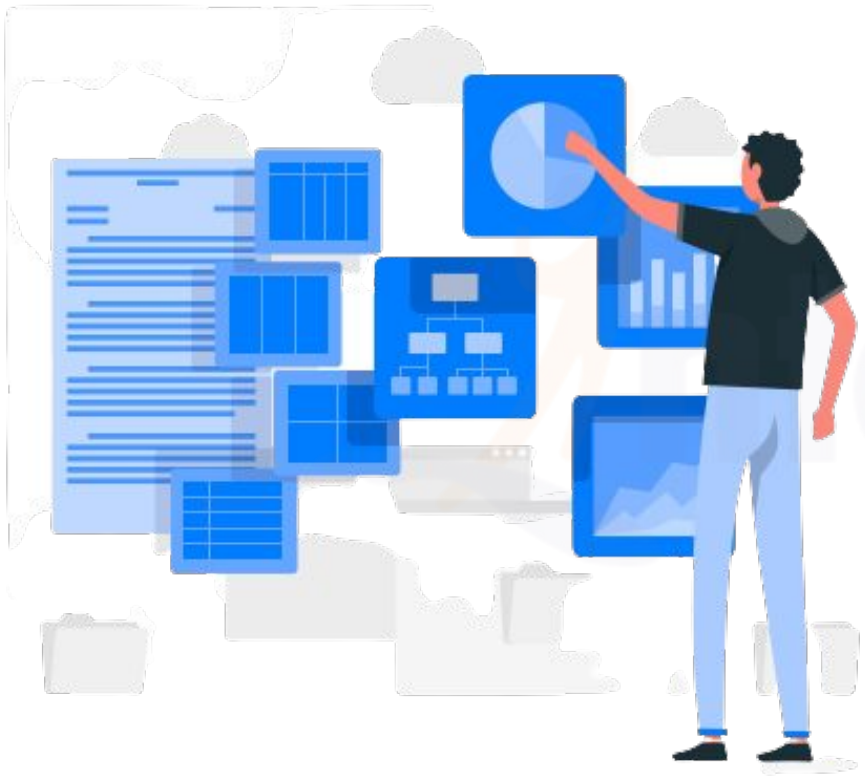
Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

- **Primitive data types:** The primitive data types include Boolean, char, byte, short, int, long, float and double.
- **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Data Types in Java



Data Types in Java



What are Primitive Data Types?

In Java, the **primitive data types** are the **predefined data types** of Java. They specify the size and type of any standard values. Java has **8 primitive data types** namely byte, short, int, long, float, double, char and Boolean.

Data Types in Java

Learning about Primitive Data Types

Data Type	Description
Boolean	The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.
char	The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.
byte	The byte data type is an example of primitive data type. It's a 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128,maximum value is 127
short	The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767
int	The int data type is a 32-bit signed two's complement integer. Its value-range lies between - 2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive). Its minimum value is -2,147,483,648and maximum value is 2,147,483,647
long	The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63 -1)(inclusive). Its minimum value is - 9,223,372,036,854,775,808and maximum value is 9,223,372,036,854,775,807. The long data type is used when you need a range of values more than those provided by int.

Data Types in Java

Learning about Primitive Data Types Cont.

Data Type	Description
float	The float data type is a single-precision 32-bit IEEE 754 floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency
double	The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency

Data Types in Java

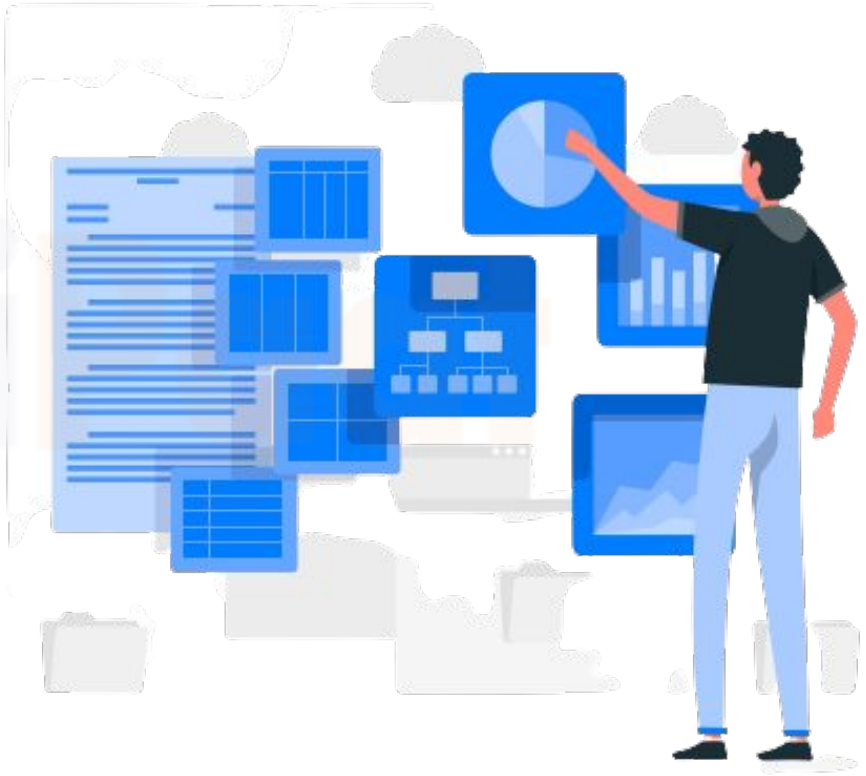
Default Size and Values of Primitive Data Types

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

Data Types in Java

What are Non-Primitive Data Types?

Unlike primitive data types, non-primitive data types are not predefined. These are user-defined data types created by programmers. These data types are used to store multiple values. There are five types of non-primitive data types – Class, Object, String, Array, Interface

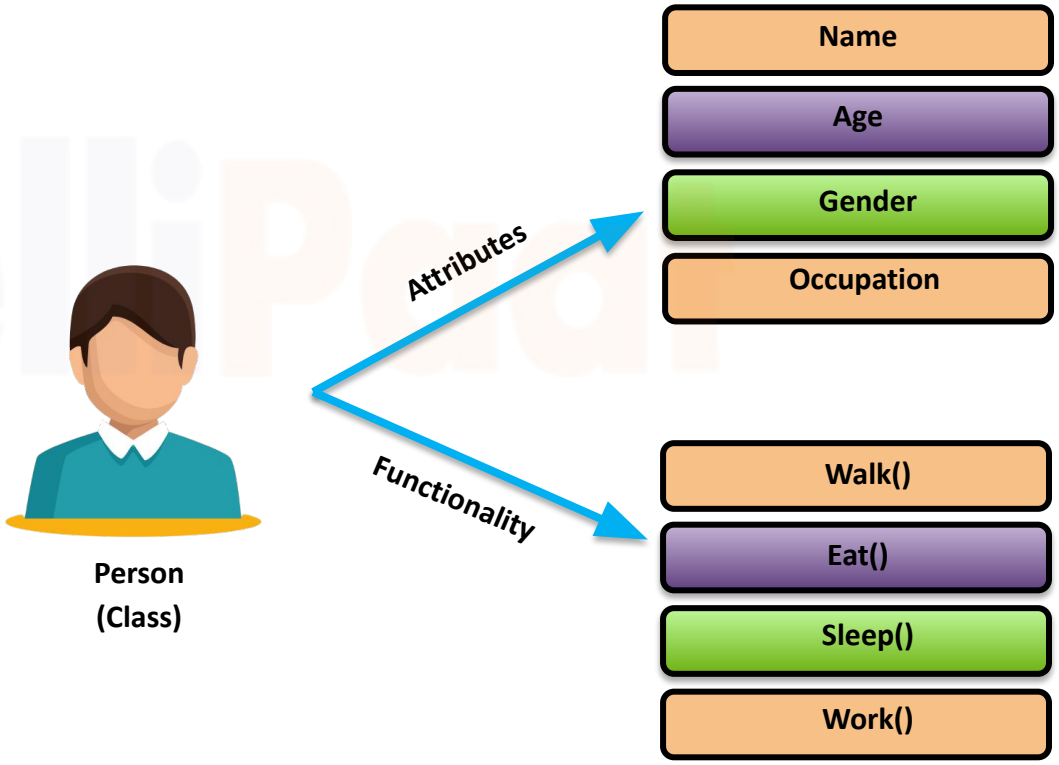


Data Types in Java

What is 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.

An **object** is the variable of the class, which can access the elements of class i.e. methods and variables.



Lets see an example in the next slide :

Data Types in Java

Class and Objects - Example

```java
public class Intellipaat{

        // defining the variables of class
        int a = 20;
        int b = 10;
        int c;

        // defining the methods of class
        public void add () {
            int c = a + b;
            System.out.println("Addition of numbers is: " + c);
        }
    // main method
    public static void main (String[] args) {
        // creating the object of class
        Intellipaat obj = new Intellipaat();

        // calling the methods
        obj.add();
    }
}
```

- Program flow enters the **main** function. Declare and initialize obj as a **new** instance of the **Intellipaat class**.
- Call the **add()** method on the obj object.
- Inside the add() method, calculate the **sum of a and b, and print the result.**
- **Return** the main method and reach its end.
- Program **terminates**.

```
Addition of numbers is: 30
```

Data Types in Java

What is 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.

Interfaces enable you to achieve abstraction, multiple inheritance, and loose coupling in your code.

Lets see an example in the next slide :

Data Types in Java

Interface - Example

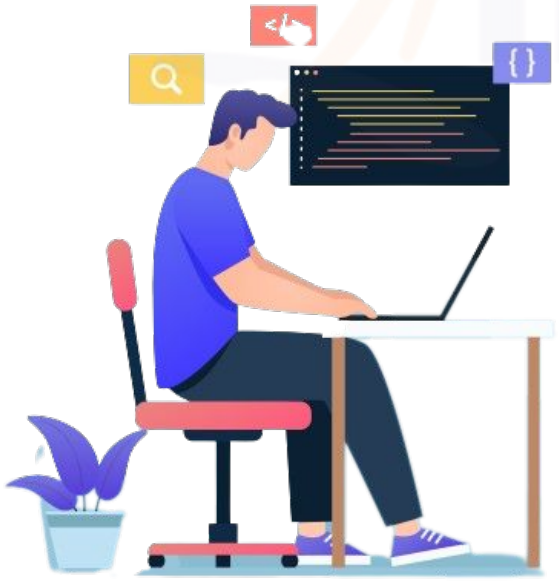
```java
interface AdditionInterface {
    void add();
}
public class Intellipaat implements AdditionInterface {

        // defining the variables of class
        int a = 10;
        int b = 20;
        int c;

        // implementing the interface methods
        public void add () {
            int c = a + b;
            System.out.println("Addition of numbers is: " + c);
        }

    // main method
    public static void main (String[] args) throws IOException {
        Intellipaat obj = new Intellipaat();
        // calling the methods
        obj.add();
    }
}
```

- Declare and initialize **obj** as a **new** instance of the **Intellipaat** class.
- Call the **add()** method on the obj object.
- Inside the **add()** method of the **Intellipaat** class:
- Calculate the **sum of a and b**, and **print** the **result**.
- **Return** to the **main** method and reach its end.
- Program **terminates**.

Data Types in Java

What is a String?

A string represents a sequence of characters. For example, “Intellipaat", "Hello world", "This is Java Tutorial”, etc. String is the class of Java.



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

        // creating a string and initializing it
        String str = "Intellipaat Welcomes You to Java Online Classes";

        // applying substring() on above string
        String subStr = str.substring(0,24);

        // printing the string
        System.out.println(subStr);
    }
}
```

```
Intellipaat Welcomes You
```

Data Types in Java

What is an Array?

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

        // Creating an array of integers
        int[] numbers = new int[5];

        // Assigning values to array elements
        numbers[0] = 10;
        numbers[1] = 20;
        numbers[2] = 30;
        numbers[3] = 40;
        numbers[4] = 50;

        // Accessing and printing array elements
        System.out.println("Array elements are:");
        for (int i = 0; i < numbers.length; i++) {
            System.out.println("Element at" + i + ": " + numbers[i]);
        }
    }
}
```

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.

```
Array elements are:
Element at 0: 10
Element at 1: 20
Element at 2: 30
Element at 3: 40
Element at 4: 50
```

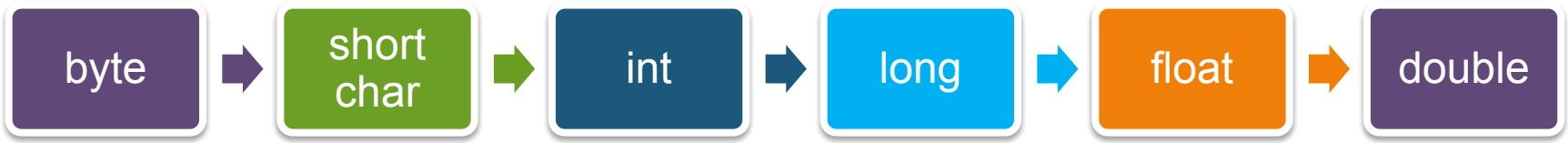
Data Types in Java

Primitive Type Conversion

Primitive type conversion, also known as **type casting**, is the process of converting a value of **one primitive data** type to **another primitive data type**.

There are **two** types of primitive type conversions in Java:

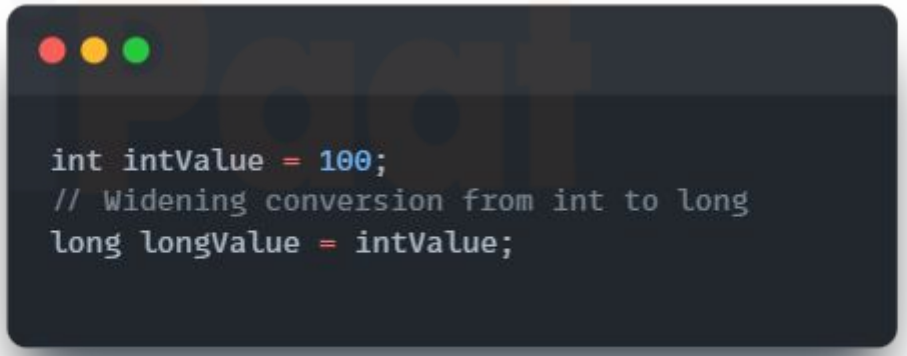
- **Widening** (implicit) conversion
- **Narrowing** (explicit) conversion.



Data Types in Java

Widening (implicit) Conversion

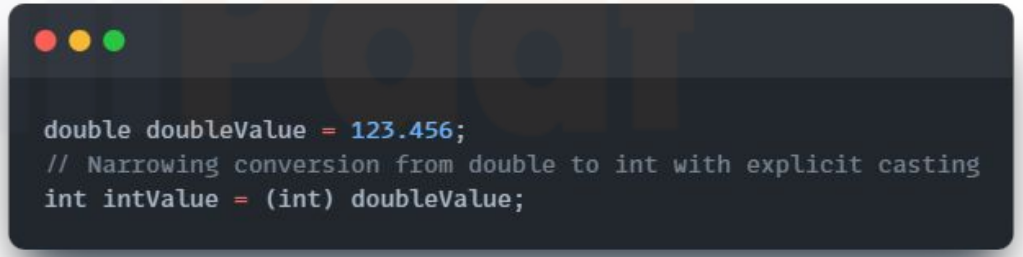
Widening conversion, also known as implicit conversion, occurs when you convert a lower-range data type to a higher-range data type. Java automatically performs widening conversions without requiring explicit casting. This is because there's no risk of data loss during such conversions.



Data Types in Java

Narrowing (explicit) Conversion

Narrowing conversion, also known as explicit conversion or casting, is the process of converting a higher-range data type to a lower-range data type. This requires explicit casting because there's a potential risk of data loss due to the reduced size or precision of the destination data type.



```
double doubleValue = 123.456;
// Narrowing conversion from double to int with explicit casting
int intValue = (int) doubleValue;
```

Try Yourself

Quiz

Question 1: Which of the following data types in Java represents a 64-bit integer?

a) int

b) long

c) byte

d) short

Question 2: What is the default value of a char variable in Java if it is not explicitly initialized?

a) 'A'

b) u0000

c) ' '

d) null

Try Yourself

Answers

Question 1: Which of the following data types in Java represents a 64-bit integer?

a) int

b) long

c) byte

d) short

Question 2: What is the default value of a char variable in Java if it is not explicitly initialized?

a) 'A'

b) u0000

c) ' '

d) null

Try Yourself

Quiz

Question 3: Which of the following data types is used to store a single-precision floating-point number in Java?

a) float

b) double

c) decimal

d) real

Question 4: In Java, which data type is used to store true/false values?

a) int

b) char

c) boolean

d) byte

Try Yourself

Answers

Question 3: Which of the following data types is used to store a single-precision floating-point number in Java?

a) float

b) double

c) decimal

d) real

Question 4: In Java, which data type is used to store true/false values?

a) int

b) char

c) boolean

d) byte

Try Yourself

Quiz

Question 5: What is the size of the boolean data type in Java?

a) 8 bits

b) 16 bits

c) 32 bits

d) 1 bit

Try Yourself

Answer

Question 5: What is the size of the boolean data type in Java?

a) 8 bits

b) 16 bits

c) 32 bits

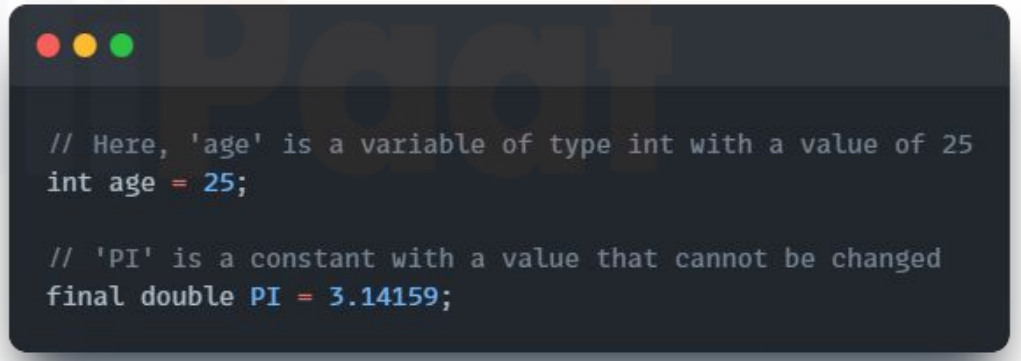
d) 1 bit

Fundamentals of Java

Variables and Constants

Variables are used to store and manage data in programming. They have a name, a data type, and a value. Variables can be assigned values, and those values can change during the program's execution.

Constants are similar to variables, but their values remain constant throughout the program's execution. They are useful for storing values that should not be changed during the program's runtime.



```
// Here, 'age' is a variable of type int with a value of 25
int age = 25;

// 'PI' is a constant with a value that cannot be changed
final double PI = 3.14159;
```

Keywords

Keywords are reserved words in a programming language that have special meanings and functionalities. They cannot be used as identifiers (names for variables, functions, etc.).

abstarct	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Separators

Separators are characters used in programming to delimit or separate different elements within code. They help in visually organizing and structuring code, making it easier to read and understand.

There are mainly **three** separators in Java :

1) **Whitespace:** Spaces, tabs, and line breaks used for formatting code.
2) **Semicolon (;):** Used to terminate statements in most programming languages.
3) **Comma (,):** Used to separate items in a list, such as function arguments.

Character Encoding

Character encoding is the representation of characters in computers.

1) **ASCII:** Originally used in early computers, ASCII encodes characters using 7 or 8 bits, representing English letters, digits, and symbols.
2) **UTF-8:** A variable-length encoding for Unicode that uses 8-bit units for common characters and more bits for less common characters.
3) **UTF-16:** Uses 16-bit units for characters. It can represent characters from the Basic Multilingual Plane (BMP) using one unit and characters from supplementary planes using two units.

Try Yourself

Quiz

Question 1: Which of the following is NOT a primitive data type in Java?

a) int

b) string

c) double

d) boolean

Question 2: What is the purpose of the final keyword in Java?

a) It declares a constant variable.

b) It declares an integer data type.

c) It marks a class as abstract.

d) It specifies the access level of a variable.

Try Yourself

Answers

Question 1: Which of the following is NOT a primitive data type in Java?

a) int

b) string

c) double

d) boolean

Question 2: What is the purpose of the final keyword in Java?

a) It declares a constant variable.

b) It declares an integer data type.

c) It marks a class as abstract.

d) It specifies the access level of a variable.

Try Yourself

Quiz

Question 3: Which of the following statements about variables in Java is true?

a) Variables must be declared with a specific data type.

b) Variables do not need to be assigned a value when declared.

c) Variables can be declared without specifying a name.

d) Variables can only be assigned a value once during their lifetime.

Question 4: What is the range of values that can be stored in a char data type in Java?

a) -128 to 127

b) 0 to 65535

c) -32768 to 32767

d) -2147483648 to 2147483647

Try Yourself

Answers

Question 3: Which of the following statements about variables in Java is true?

a) Variables must be declared with a specific data type.

b) Variables do not need to be assigned a value when declared.

c) Variables can be declared without specifying a name.

d) Variables can only be assigned a value once during their lifetime.

Question 4: What is the range of values that can be stored in a char data type in Java?

a) -128 to 127

b) 0 to 65535

c) -32768 to 32767

d) -2147483648 to 2147483647

Try Yourself

Quiz

Question 5: Which of the following is a valid way to declare a constant in Java?

a) constant int VALUE = 10;

b) final VALUE = 10;

c) static final int VALUE = 10;

d) const int VALUE = 10;

Try Yourself

Answer

Question 5: Which of the following is a valid way to declare a constant in Java?

a) constant int VALUE = 10;

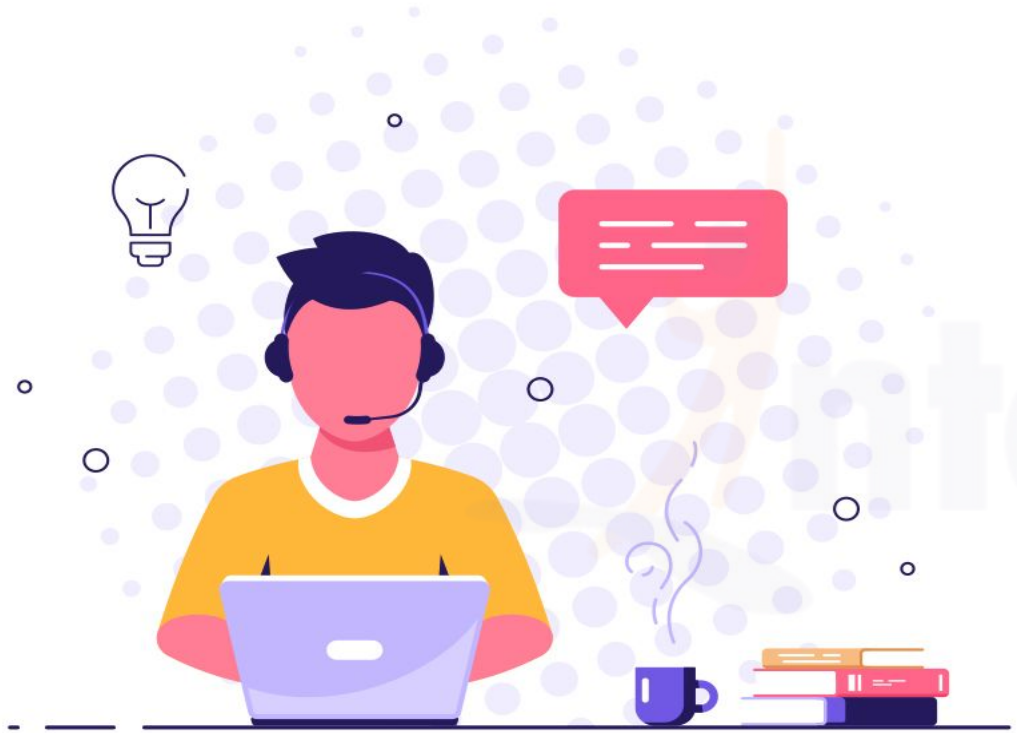
b) final VALUE = 10;

c) static final int VALUE = 10;

d) const int VALUE = 10;

Any Questions?

Thank You



Contact Us

Support Contact No: 080-4524-9465

support@intellipaat.com

If any doubt, please get in touch with us.