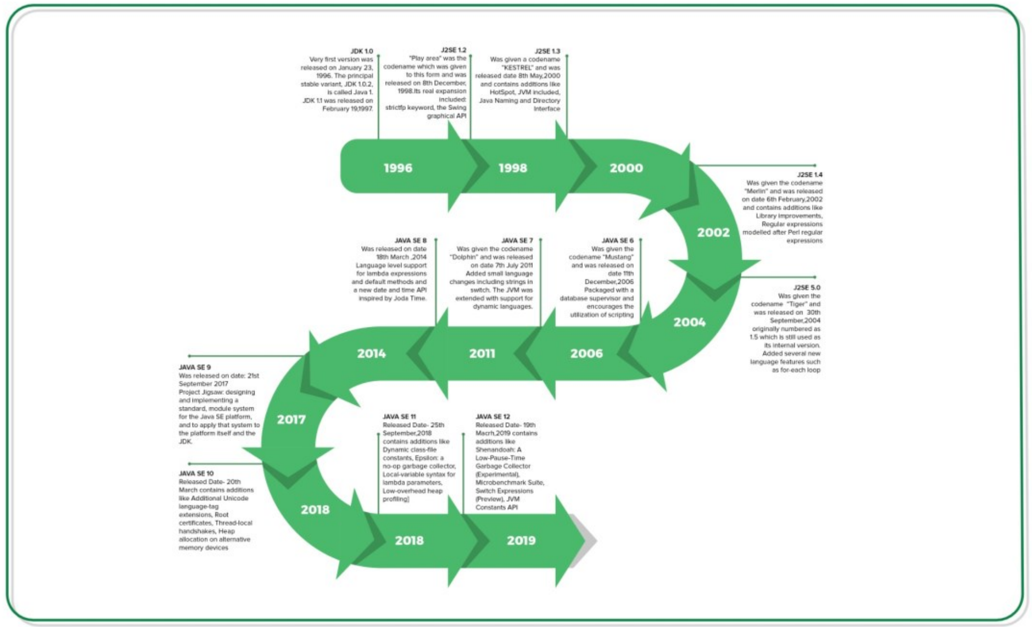
Session 1

Introduction to Java

- Overview of Java
- Setting up Java Development Environment
- Main method
- Basic Syntax and Data Types
- Variables and Constants
- Multiple Choice Questions (MCQs)
- First Java Program: Hello Worl

Overview of Java



JDK 1.0
Very first version was released on January 23, 1996. The principal stable variant, JDK 1.0.2, is called Java 1. JDK 1.1 was released on February 19,1997.

J2SE 1.2
"Play area" was the codename which was given to this form and was released on 8th December, 1998.Its real expansion included: strictfp keyword, the Swing graphical API

J2SE 1.3
Was given a codename "KESTREL" and was released date 8th May,2000 and contains additions like HotSpot, JVM included, Java Naming and Directory Interface

J2SE 1.4
Was given the codename "Merlin" and was released on date 6th February,2002 and contains additions like Library improvements, Regular expressions modelled after Perl regular expressions

J2SE 5.0
Was given the codename "Tiger" and was released on 30th September,2004 originally numbered as 1.5 which is still used as its internal version. Added several new language features such as for-each loop

JAVA SE 6
Was given the codename "Mustang" and was released on date 11th December,2006 Packaged with a database supervisor and encourages the utilization of scripting

JAVA SE 7
Was given the codename "Dolphin" and was released on date 7th July 2011 Added small language changes including strings in switch. The JVM was extended with support for dynamic languages.

JAVA SE 8
Was released on date 18th March ,2014 Language level support for lambda expressions and default methods and a new date and time API inspired by Joda Time.

JAVA SE 9
Was released on date: 21st September 2017 Project Jigsaw: designing and implementing a standard, module system for the Java SE platform, and to apply that system to the platform itself and the JDK.

JAVA SE 10
Released Date- 20th March contains additions like Additional Unicode language-tag extensions, Root certificates, Thread-local handshakes, Heap allocation on alternative memory devices

JAVA SE 11
Released Date- 25th September,2018 contains additions like Dynamic class-file constants, Epsilon: a no-op garbage collector, Local-variable syntax for lambda parameters, Low-overhead heap profiling)

JAVA SE 12
Released Date- 19th Macrh,2019 contains additions like Shenandoah: A Low-Pause-Time Garbage Collector (Experimental), Microbenchmark Suite, Switch Expressions (Preview), JVM Constants API

Java is a high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle Corporation) in 1995.

It was designed to be platform-independent, meaning that Java programs can run on any device with a Java Virtual Machine (JVM) installed, regardless of the underlying hardware and operating system.

Java's syntax is influenced by C and C++, but it eliminates certain low-level features such as pointers and operator overloading to enhance security and simplicity.

Key Features of Java

- Platform Independence: Java code is compiled into bytecode, which can be executed on any device with a JVM.
- Object-Oriented: Java is based on the object-oriented programming paradigm, facilitating modular and reusable code.
- Robust and Secure: Built-in features like exception handling and automatic memory management (garbage collection) enhance program reliability and security.
- Multithreading: Java supports concurrent programming through its built-in threading capabilities, enabling efficient utilization of system resources.
- Dynamic: Java supports dynamic memory allocation, dynamic linking, and reflection, allowing for runtime flexibility and extensibility.
- Architecture-neutral and Portable: Java's architecture-neutral bytecode can be executed on any platform, making Java programs highly portable.

Advantages of Java

- Write Once, Run Anywhere (WORA): Java's platform independence allows developers to write code once and deploy it on any platform, reducing development time and effort.
- Strong Community Support: Java has a large and active community of developers, providing extensive documentation, libraries, and frameworks to support development efforts.
- Rich APIs and Libraries: Java's standard library (Java API) includes thousands of classes and methods for common tasks, enabling developers to build robust and feature-rich applications with ease.
- Automatic Memory Management: Java's garbage collector automatically manages memory allocation and deallocation, reducing the risk of memory leaks and simplifying memory management for developers.
- High Performance: Java's Just-In-Time (JIT) compiler optimizes bytecode into native machine code at runtime, delivering high performance comparable to native languages like C and C++.

Official Java JDK :

Code Editer :

Java Book Reference:

1. **Effective Java" by Joshua Bloch:** This book offers best practices and advanced tips for writing better Java code. It covers various topics such as object creation, classes and interfaces, generics, enums, annotations, and more.

2. **Java Concurrency in Practice by Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, Doug Lea**: This book is a must-read for mastering concurrent programming in Java. It covers topics such as thread safety, synchronization, concurrent collections, and executor frameworks.

3. **Java: The Complete Reference" by Herbert Schildt**: This comprehensive book covers all aspects of the Java programming language, including syntax, libraries, and advanced features. It's suitable for both beginners and experienced developers looking to deepen their understanding of Java.

Path setting

1. Install JDK: Download and install the Java Development Kit (JDK).

2. Find JDK Directory: Note the directory where JDK is installed.

3. Windows:

Open System Properties -> Environment Variables.

Edit "Path" variable, add JDK bin directory path.

4. macOS & Linux:

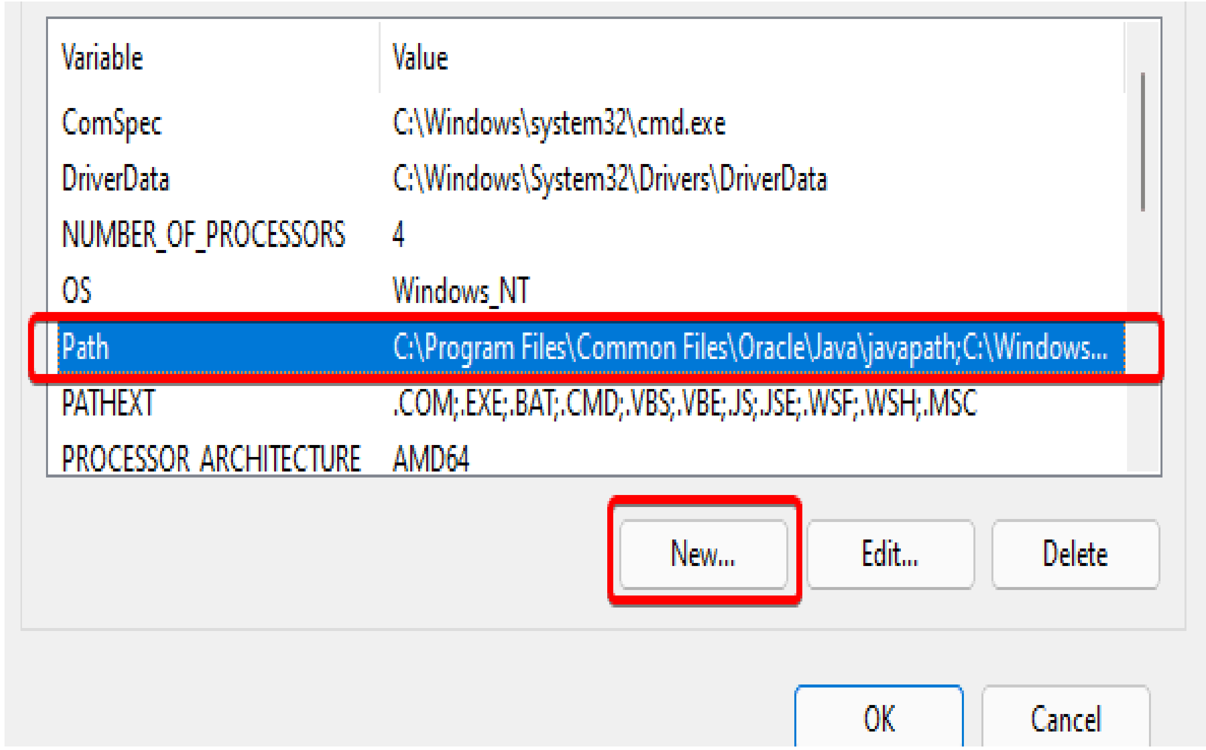
Open shell configuration file (e.g., `.bashrc`).

Add `export JAVA_HOME=/path/to/your/jdk` and `export PATH=$JAVA_HOME/bin:$PATH`.

Save & reload the shell configuration.

5. Verify: Open a new terminal, type `java -version` to ensure Java is recognized.

These steps ensure that your system can locate and use the Java Development Kit for running Java programs.



Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Windows...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

Main Method

In Java, the `main` method is the entry point for a Java application.
It is declared with the following signature:

~~~
```java
public static void main(String[] args) {
    // Main code goes here
}
```
~~~

The `main` method is where the execution of the Java program begins.
It must be declared as `public`, `static`, and `void`.
The `String[] args` parameter allows command-line arguments to be passed to the program.

```
//Example**:
//java

public class Main {
      public static void main(String[] args) {
          // Main code goes here
          System.out.println("Hello, world!");
      }
  }
```

Basic Syntax and Data Types

Basic Syntax and Data Types

Java is a structured programming language that follows a set of rules for writing code, known as syntax.

Key syntax rules include:

Statements end with a semicolon (;).

Code blocks are enclosed in curly braces {}.

Case sensitivity: Java is case-sensitive, meaning that uppercase and lowercase letters are treated differently.

Comments: Java supports single-line (//) and multi-line (/* */) comments for code documentation and explanation.

Overview of Primitive Data Types:

In Java, data types specify the size and type of values that can be stored in variables.

Primitive data types represent basic data types and are not objects.

Common primitive data types include:

int: Represents integer numbers.

double: Represents floating-point numbers with decimal values.

boolean: Represents true or false values.

char: Represents a single Unicode character.

byte, short, long, float: Additional primitive data types with specific size and range.

Example

-
  ```java
-     // Declaration and initialization of int variable
-     **int num1 = 10;**
- 
-     // Declaration of double variable
-     **double num2;**
-     // Initialization of double variable
-     **num2 = 3.14;**
- 
-     // Declaration and initialization of boolean variable
-     **boolean isJavaFun = true;**
- 
-     // Declaration and initialization of char variable
-     **char grade = 'A';**
-     ```

**Examples Demonstrating Variable Declaration and Initialization:**

Variable declaration in Java involves specifying the data type and name of the variable.

Initialization assigns an initial value to the variable.
  ```

Note:

It's essential to choose the appropriate data type based on the type of data being stored and the range of values it needs to represent.

Java also supports non-primitive data types, such as arrays, classes, and interfaces, which will be covered in later slides.

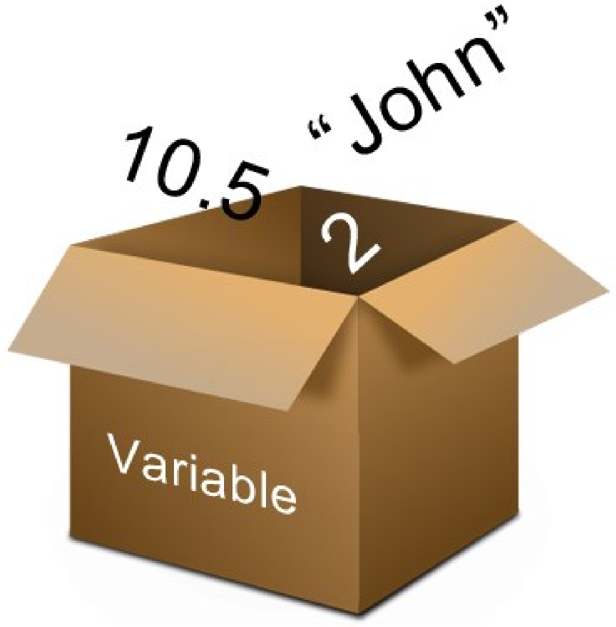
This content provides a foundational understanding of basic syntax and data types in Java, laying the groundwork for further exploration of Java programming concepts.

Variables and Constants

Understanding Variables and Constants in Java:

Variables in Java are named memory locations used to store data during program execution. They can vary in value.

Constants are variables whose values cannot be changed once assigned. They are declared using the `final` keyword.



Code Examples Illustrating Variable Declaration and Usage:

-
  ```java
- // Variable declaration and initialization
- **int numStudents = 30;**
- 
- // Variable declaration without initialization
- **double averageGrade;**
- 
- // Initialization of previously declared variable
- **averageGrade = 85.5;**
- 
- // Constants declaration
- **final double PI = 3.14159;**
- 
- // Attempting to reassign a constant (will result in a compilation error)
- // PI = 3.14; // Error: cannot assign a value to final variable PI
- ```

## Explanation of Variable Naming Conventions:

- Variable names in Java must follow certain rules:

- Must begin with a letter, underscore (_), or dollar sign ($).

- Subsequent characters can be letters, digits, underscores, or dollar signs.

- Cannot be a keyword or reserved word in Java.

- Should follow camelCase convention for readability.

- Example: `int studentAge;`
  ```

Note:

Variables and constants play a crucial role in storing and manipulating data in Java programs.

Following naming conventions and using descriptive variable names improves code readability and maintainability.

Constants are useful for defining values that should not change throughout the program's execution.

First Java Program: Hello World

```
//Writing the Code Snippet for Hello World
//Program:

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

Explanation of the Classic "Hello World" Program:

The "Hello World" program is a simple yet iconic example used to introduce beginners to programming languages.

It consists of a single statement that prints "Hello, World!" to the console.

- **Running the Program and Observing the Output:**

- To run the program:

1. Save the code in a file named `HelloWorld.java`.

2. Open a command prompt or terminal.

3. Navigate to the directory containing `HelloWorld.java`.

4. Compile the program using the command: `javac HelloWorld.java`.

5. Run the compiled program using the command: `java HelloWorld`.

The output will be:

```
Hello, World!
```

Note:

This simple program demonstrates the basic structure of a Java program, including the use of classes, methods, and statements.

Running the program and observing the output helps familiarize beginners with the Java development environment and execution process.

Basic Interview Questions (MCQs)

1. What is the correct syntax for declaring a variable of type `int` in Java?
   A) `int num = 10;`
   B) `integer num = 10;`
   C) `num = 10;`
   D) `int num = "10";`

2. Which keyword is used to declare a constant variable in Java?
   A) `var`
   B) `const`
   C) `final`
   D) `static`

3. What is the output of the following Java code?

```java
double num = 7.5;
System.out.println(num);
```

   A) 7
   B) 7.5
   C) 8
   D) Compilation Error

4. Which of the following is NOT a valid variable name in Java?
   A) `myVariable`
   B) `_myVariable`
   C) `$myVariable`
   D) `2myVariable`

5. What does the `main` method signify in a Java program?
   A) It is the starting point of program execution.
   B) It defines the primary functionality of the program.
   C) It is used to declare global variables.
   D) It is optional in Java programs.

Basic Interview Questions (MCQs)

6. Which data type is used to store characters in Java?
   A) `char`
   B) `string`
   C) `character`
   D) `varchar`

7. What is the purpose of the `System.out.println()` statement in Java?
   A) To print a message to the console.
   B) To read input from the user.
   C) To define a new class.
   D) To declare a variable.

8. What is the result of the expression `5 % 2` in Java?
   A) 2
   B) 2.5
   C) 0
   D) 1

9. Which of the following is NOT a primitive data type in Java?
   A) `int`
   B) `double`
   C) `string`
   D) `boolean`

10. What is the correct syntax for a single-line comment in Java?
   A) `/* This is a comment */`
   B) `// This is a comment`
   C) `# This is a comment`
   D) `<!-- This is a comment -->`

Example Problems

1. Problem:

- Write a Java program to calculate the area of a rectangle given its length and width.

```
//Solution:
 //java

public class RectangleArea {
       public static void main(String[] args) {
           double length = 5.0;
           double width = 3.0;

           double area = length * width;

           System.out.println("Area of the rectangle: " + area);
       }
   }
```

Example Problems

2. Problem:

- Write a Java program to convert temperature from Celsius to Fahrenheit.

```
//Solution:
 //java

public class CelsiusToFahrenheit {
    public static void main(String[] args) {
        double celsius = 20.0;

        double fahrenheit = (celsius * 9/5) + 32;

        System.out.println("Temperature in Fahrenheit: " + fahrenheit);
    }
  }
```

Example Problems

3. Problem:

- Write a Java program to check if a given number is even or odd.

```
//Solution:
 //java

public class EvenOrOdd {
    public static void main(String[] args) {
        int num = 7;

        if (num % 2 == 0) {
            System.out.println(num + " is even.");
        } else {
            System.out.println(num + " is odd.");
        }
    }
}
```

Thank You

By AMA