
Introduction to Java

Mahmudul Hasan

QA Automation Engineer
Email: mahmudul006@gmail.com

Outline

- How to learn effectively
- Setting up the environment
- Initialising a Java Maven project
- Welcome to Java
- Variables in Java
- Rules for naming variables
- Java Reserved Keywords
- Java Variable Naming Conventions
- Data Types in Java
- Data Types Default Values
- Methods in Java
- Types of Methods in Java
- Java Method Declaration
- Java Method Code Example

How to learn effectively

- 📝 **Notes**: Take notes while learning
- **Practice**: Practice what you learn
- **Promodoro**: Use the Pomodoro technique

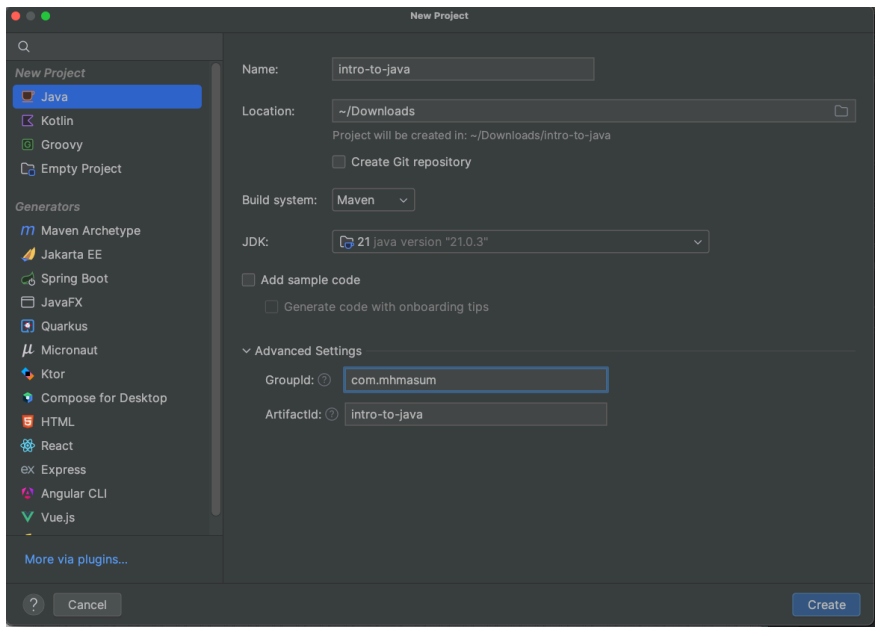
Setting up the environment

- Java: Install Java JDK
  Download JDK(21) from here
- IDE: Install IntelliJ Community Edition
  Download IntelliJ IDEA Community Edition from here

Initialising a Java Maven project

Steps to initialise a Java Maven project

- Open IntelliJ IDEA CE
- Click on `New Project`
- Select `Java` from the left pane
- Click `Create`



Welcome to Java

```
public class ExampleJava {
}
```

Welcome to Java

```
public class ExampleJava {
    public static void main(String[] args) {
    }
}
```

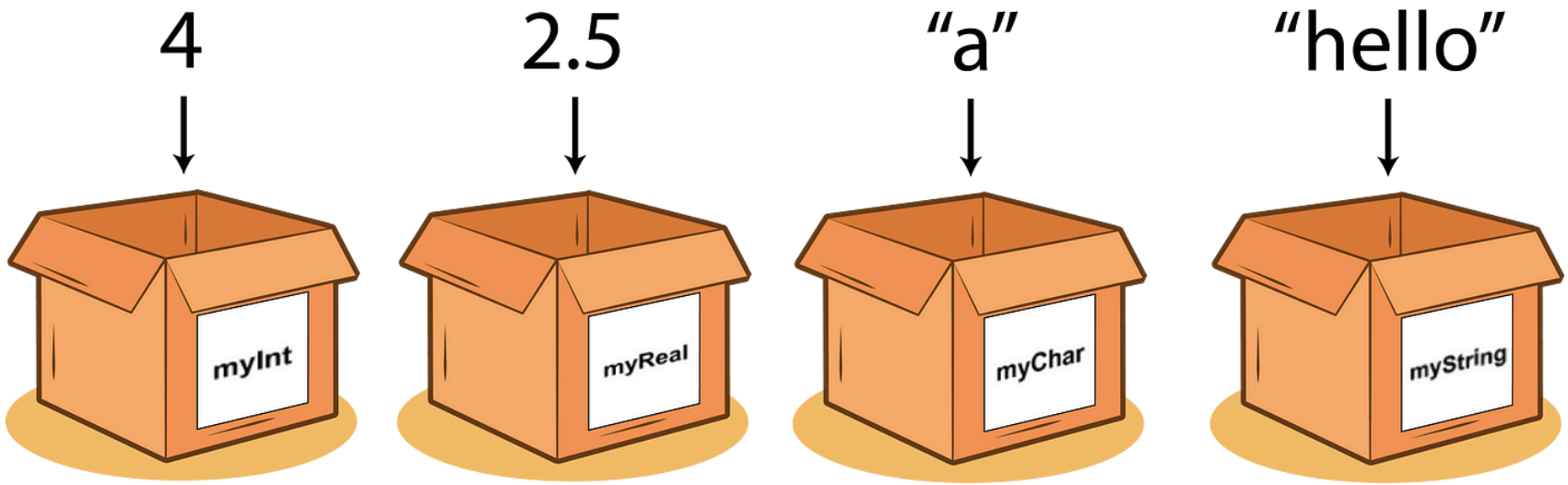
Welcome to Java

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

Variables in Java

Definition:

- A variable is a container that holds a value.
- Java Documentation: Variables



Rules for naming variables

- Variable names are case-sensitive
- Variable names can contain letters, digits, and the underscore character
- Variable names cannot start with a digit or contain spaces
- Variable names must not be a reserved keyword
- Variable names should be meaningful

Java Reserved Keywords

Keywords In Java

1. abstract
2. assert
3. boolean
4. break
5. byte
6. case
7. catch
8. char
9. class
10. continue
11. default
12. do
13. double
14. else
15. enum
16. extends
17. final
18. finally
19. float
20. for
21. if
22. implements
23. import
24. instanceof
25. int
26. interface
27. long
28. native
29. new
30. package
31. private
32. protected
33. public
34. return
35. short
36. static
37. strictfp
38. super
39. switch
40. synchronized
41. this
42. throw
43. throws
44. transient
45. try
46. void
47. volatile
48. while

Java Variable Naming Conventions

- Variable names should start with a lowercase letter
- Variable names should be in camelCase format. For example, `firstName`, `lastName`

Other conventions

- PascalCase: `FirstName`, `LastName`
- snake_case: `first_name`, `last_name`
- kebab-case: `first-name`, `last-name`
- UPPER_CASE: `FIRST_NAME`, `LAST_NAME`

Data Types in Java

- ***Definition:*** A data type is a classification of data.
- ***Java Documentation:*** Data Types
- ***Article:*** Java Data Types

Primitive Data Types in Java

- Definition: Primitive data types specify the size and type of variable values.
- Examples: `int`, `float`, `double`, `char`, `boolean` etc.

```
int age = 25;
char c = 'A';
```

Non-Primitive Data Types/Reference Data Types in Java

- Definition: Reference data types are used to store the reference/address of variables.
- Examples: `String`, `Array`, `Class`, `Interface` etc.

```
String name = "John";
```

Data Types Default Values

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'','\n',' β'	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

Methods in Java

- ***Definition:*** A method is a block of code that performs a specific task.
- ***Java Documentation:*** Methods

Benefits of using methods

- Reusability
- Modularity

Types of Methods in Java

- ***Definition:*** Methods in Java are classified into two types:
  - Built-in methods
  - User-defined methods

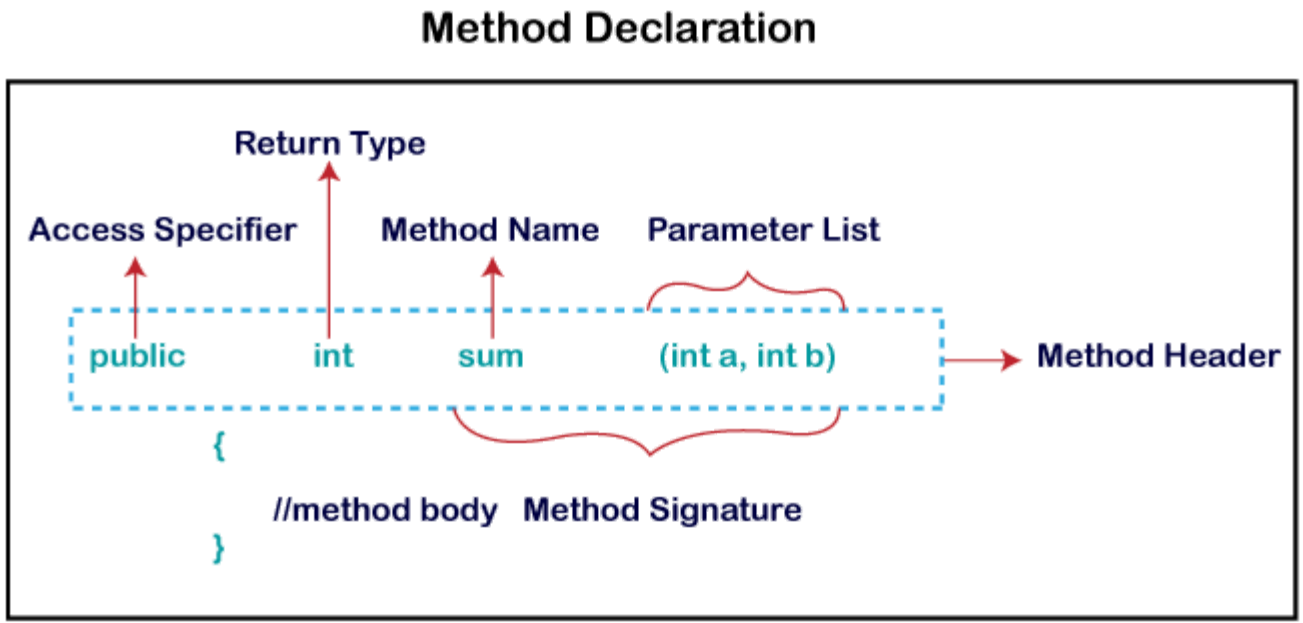
Built-in Methods

- ***Definition:*** Built-in methods are methods that are provided by Java.
- ***Examples:*** `println()`, `nextLine()`, `next()`, `nextInt()`, `nextDouble()`
- ***Java Documentation:*** Built-in Methods

User-defined Methods

- ***Definition:*** User-defined methods are methods that are created by the user.

Java Method Declaration



Java Method Code Example

```
public class Addition
{
    public static void main(String[] args)
    {
        int a = 19;
        int b = 5;
        //method calling
        int c = add(a, b);   //a and b are actual parameters
        System.out.println("The sum of a and b is= " + c);
    }
    //user defined method
    public static int add(int n1, int n2)   //n1 and n2 are formal parameters
    {
      int s;
      s=n1+n2;
      return s; //returning the sum
    }
}
```

Thank you ❤️

qa-june-2024-automation-with-java-slides