

# Introduction

This is the pre-reading content for the Java workshop.

This document covers the following

1. How to get java running with an IDE on your machine
2. Introduction to basic java programming concepts
  - a. Variables
  - b. Operators
  - c. Structure of Java class
  - d. Inputs (Scanner)
3. Exercise 1

A simple exercise is also included at the end of this document alongside the boilerplate code. Do attempt it before coming for the workshop.

## Installing an IDE to run Java

For this workshop, we highly encourage everyone to use an Integrated Development Environment (IDE) for coding in Java. It contains a series of tools and in-built functions that are useful for development, and will come in handy for subsequent workshops.

If you already have your own IDE and Java setup, feel free to continue using it, we will be using IntelliJ for this workshop.

Please follow this guide to install IntelliJ IDEA Ultimate, using your student emails to sign up for the trial, as this version contains the full set of tools and framework support for Spring, which will be helpful for the next workshop.

<https://www.jetbrains.com/help/idea/installation-guide.html>

After completing installation, start IntelliJ IDEA Ultimate and optionally configure some settings.

<https://www.jetbrains.com/help/idea/run-for-the-first-time.html>

Afterwards, follow the guide below to create a simple Hello World Java application. We will use JDK 17 as per shown in the guide, but if you already have an older Java binary installed in your computer, you can choose to use that as we will not be using any syntax unique to the later versions of Java.

<https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html>

# Variables

In Java, variables are used to store and manipulate data. They are containers that hold values of a specific type. Variables can be used to perform calculations and pass values between different parts of a program. They can be used in mathematical operations, conditional statements, loops, method calls, and more. Below are some key aspects to understand about variables in Java.

## Variable Declaration

Before using a variable, it needs to be declared with a specific type. For example, you can declare an integer variable and assign a value to it using the “=” operator as follows:

```
int age = 25;
```

Do note that (most) lines of java code ends with a semicolon (;) and this is used to indicate end of line.

## Variable Types

Java has different types of variables to represent various kinds of data.

Common types include:

- int: Whole numbers (e.g., 1, 10, -5)
- double: Decimal numbers (e.g., 3.14, -0.5, 2.0)
- boolean: true or false values
- char: A single character, declared using single quotes (e.g., 'a', '9', '\$')
- String: A sequence of characters, declared using double quotes (e.g., "Hello World")

## Variable Naming

Variables should be given meaningful names that reflect their purpose. They must follow certain naming conventions:

- Variable names should start with a letter (a-z or A-Z) or underscore (\_).
- They can contain letters, digits, or underscores.
- Variable names are case-sensitive.

## Variable Assignment

Once a variable is declared and initialized, its value can be changed or updated. For example:

```
int age = 25;  
age = 20; // update value of number
```

# Operators

## Arithmetic Operators

- Addition (+): Adds two operands.
- Subtraction (-): Subtracts the second operand from the first.
- Multiplication (\*): Multiplies two operands.
- Division (/): Divides the first operand by the second.
- Modulus (%): Returns the remainder after division.

## Assignment Operators

- Assignment (=): Assigns a value to a variable.
- Compound assignment operators (e.g., +=, -=, \*=, /=): Perform an operation and assign the result to the left operand.

## Increment and Decrement Operators

- Increment (++) and Decrement (--): Increase or decrease the value of an operand by 1.

# Structure of a Java file

A typical Java file follows a specific structure that includes package declarations, import statements, a public class, and a main method.

## Package

A package declaration specifies the package to which the Java file belongs. Packages are used to organize related classes and provide a hierarchical structure. It is the first non-comment line in a Java file and is optional. It is defined using the package keyword followed by the package name.

## Import Statements

Import statements are used to include classes or entire packages from other packages into the current file. They allow you to refer to classes without specifying their full package names. Import statements are placed after the package declaration (if present) and before the class declaration. They are defined using the import keyword followed by the package or class name.

## Public Class

Java files can contain one or more classes, but only one class can be declared as public. The public class represents the entry point of the program and must have the same name as the file. The public keyword is used to indicate that the class can be accessed from other classes or packages.

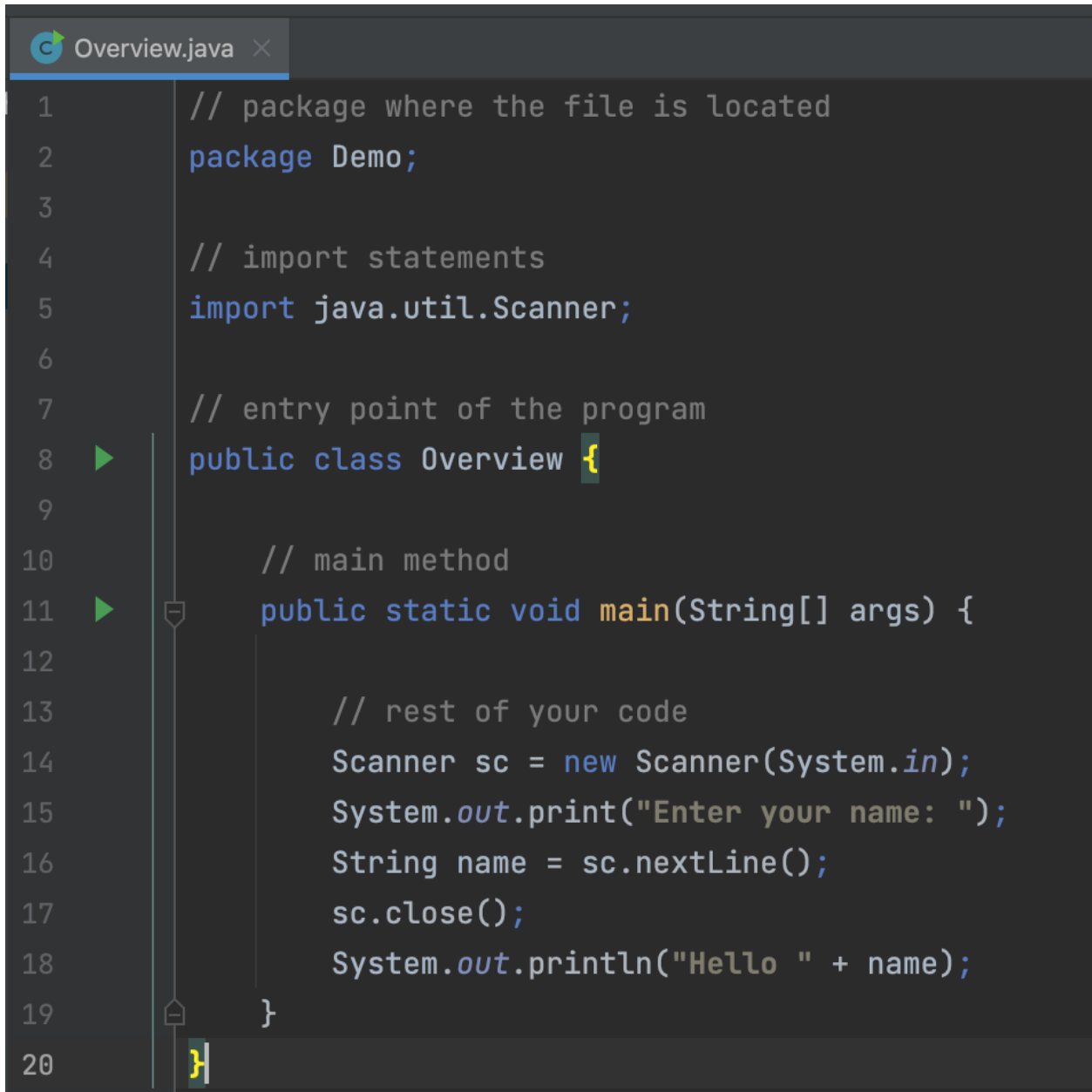
## Main Method

The main method is the starting point of execution for a Java program. It must be declared within the public class and has a specific signature:

```
public static void main(String[] args)
```

The main method contains the instructions that are executed when the program runs.

# Sample



```
Overview.java x
1 // package where the file is located
2 package Demo;
3
4 // import statements
5 import java.util.Scanner;
6
7 // entry point of the program
8 public class Overview {
9
10     // main method
11     public static void main(String[] args) {
12
13         // rest of your code
14         Scanner sc = new Scanner(System.in);
15         System.out.print("Enter your name: ");
16         String name = sc.nextLine();
17         sc.close();
18         System.out.println("Hello " + name);
19     }
20 }
```

The image shows a code editor window titled "Overview.java". The code is a Java program with the following structure: a package declaration "package Demo;", an import statement "import java.util.Scanner;", and a public class "Overview" containing a "main" method. The "main" method uses a "Scanner" object to read input from "System.in", prints the input, and then prints a greeting "Hello " followed by the input name. The code is formatted with standard Java conventions, including comments and indentation. The editor has a dark theme and a line number margin on the left.

# Scanner

In Java, input is read from the user by using the `Scanner` class. It is a part of the `java.util` package and provides convenient methods for parsing and retrieving different types of data.

Here's a brief explanation of how to use the `Scanner` class:

## Importing the `Scanner` class

Before using the `Scanner` class, you need to import it at the top of your Java file:

```
import java.util.Scanner;
```

This allows to use the Scanner class and its functions to read input from the user

## Creating a `Scanner` object

To use the `Scanner` class, you need to create a Scanner object and pass the input source as an argument to its constructor. Input sources can be from standard input, from a file or even from a URL. This will be explained in further detail when we reach the classes section of the workshop.

To read input from the keyboard (System.in), you can create a `Scanner` object as follows:

```
Scanner sc = new Scanner(System.in);
```

## Reading input

The `Scanner` class provides various methods to read different types of input. Some commonly used methods include:

- `nextLine()`: Reads a line of input as a string.
- `nextInt()`: Reads the next token as an integer.
- `nextDouble()`: Reads the next token as a double.
- `nextBoolean()`: Reads the next token as a boolean.

You can call these methods on the `Scanner` object to read the input. For example:

```
System.out.print("Enter your name: ");  
String name = sc.nextLine();  
System.out.print("Enter your age: ");  
int age = sc.nextInt();
```

## Closing the `Scanner` object

It's good practice to close the `Scanner` object when you're done reading input to release any associated resources. You can do this by calling the `close()` method on the `Scanner` object:

```
sc.close();
```

## Example code and output

```
import java.util.Scanner;  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Please enter your name:");  
        String name = sc.nextLine();  
        System.out.print("Enter your age: ");  
        int age = sc.nextInt(); // Read the next token as an integer  
        System.out.println("Your name is " + name + ", and your age is " + age);  
        sc.close();  
    }  
}
```

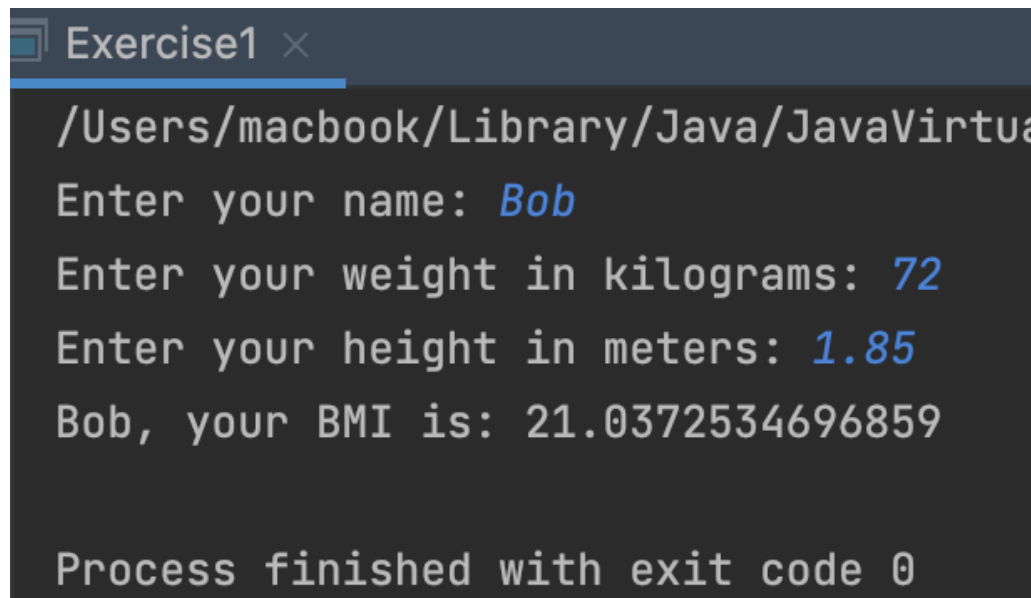
```
Please enter your name: Bob  
Enter your age: 20  
Your name is Bob, and your age is 20
```

# Exercise 1

Please attempt this Exercise prior to the Java Workshop.

Create a program that does the following

1. Prompts the user to enter the following (*hint, use scanner*)
  - a. Name
  - b. Weight in kilograms
  - c. Height in meters
2. Store the following entered values in their respective variables with correct types
3. Calculate the user's BMI with the following formula
  - a. Weight divided by height squared
4. Return the following output



```
Exercise1 x
/Users/macbook/Library/Java/JavaVirtual
Enter your name: Bob
Enter your weight in kilograms: 72
Enter your height in meters: 1.85
Bob, your BMI is: 21.0372534696859

Process finished with exit code 0
```

A boilerplate intellij project is included alongside this document, you may import that project into your IDE.