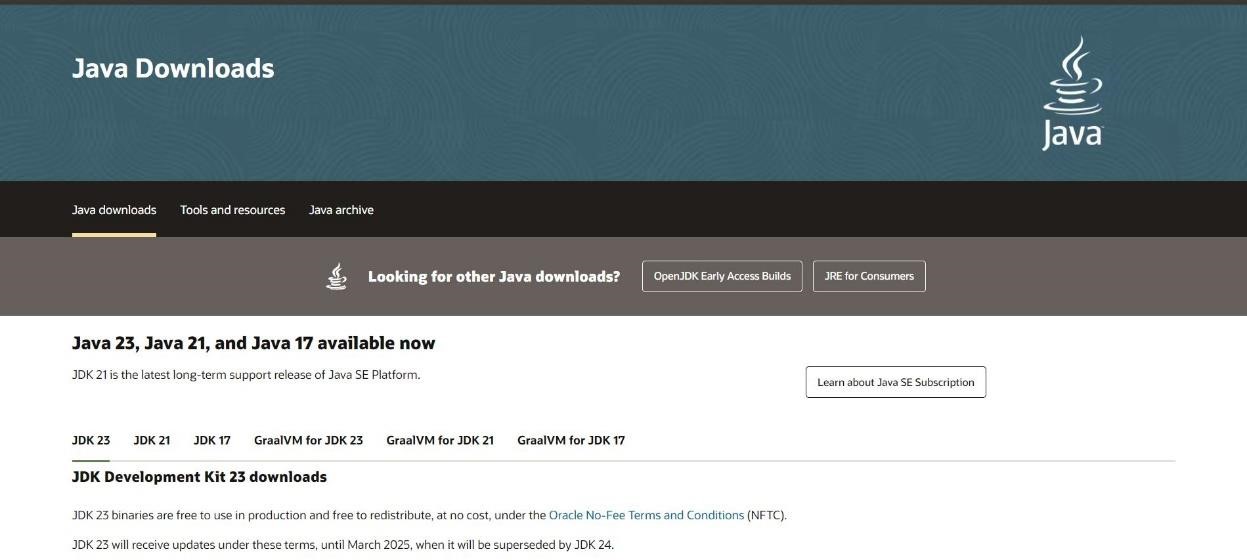


|  |  |
| --- | --- |
| **Lab 01** |  |
| **Topic** |  |
| **Objective** |  |

## Introduction to Java

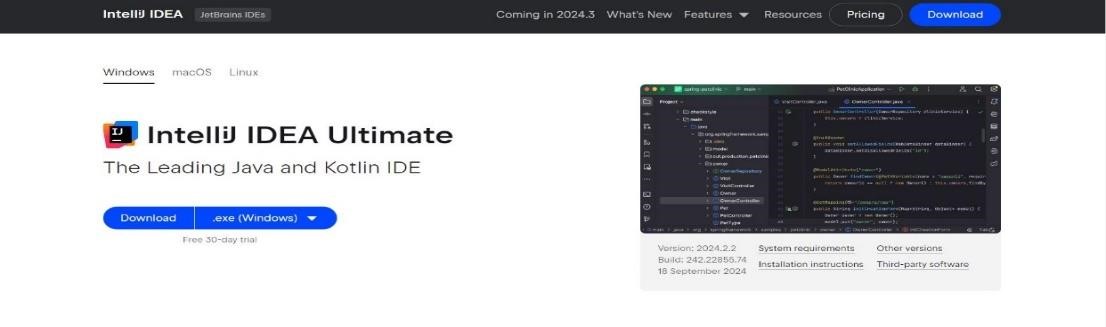
INSTALLATION PROCESS OF INTELLIJ TOOL

Intall JDK: Using any browser, download the JDK.



After successful download, install the JDK (IntelliJ).

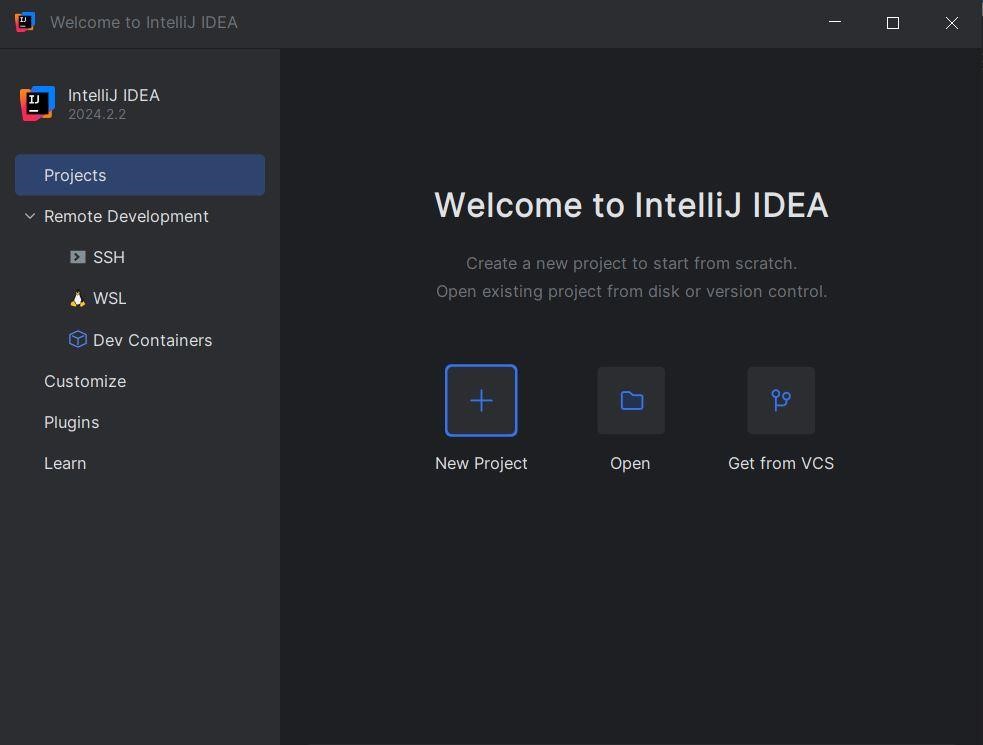
Then download the setup of IntelliJ using the following link: <https://www.jetbrains.com/idea/download/?section=windows>



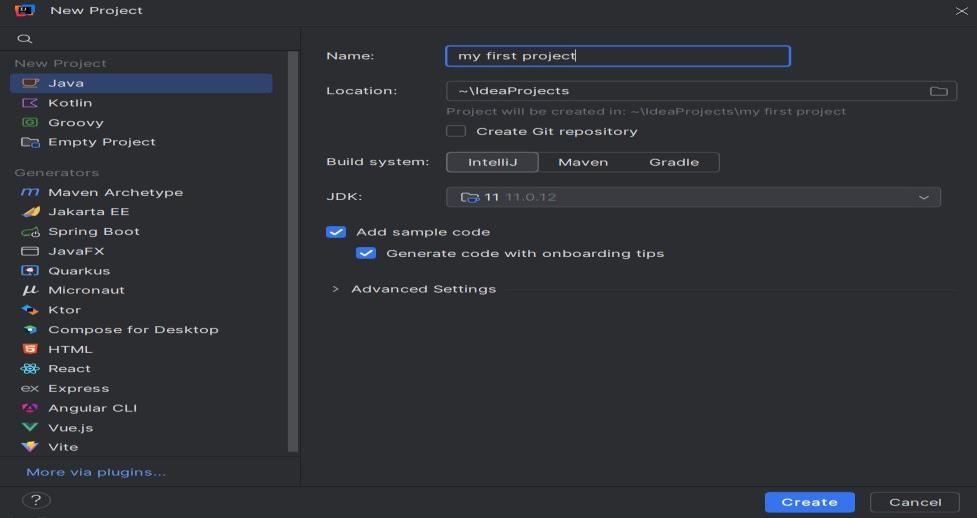
After successful installation, run the IntelliJ.



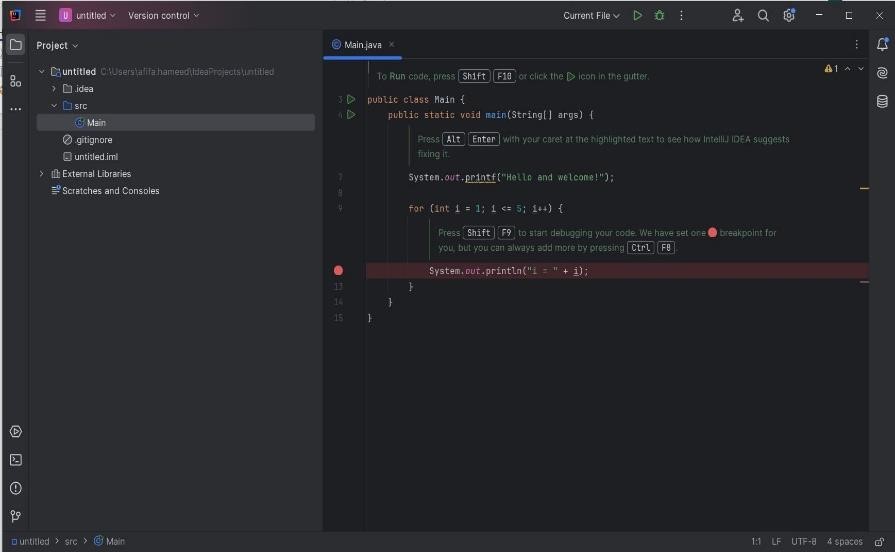
After running the setup, create a new project.



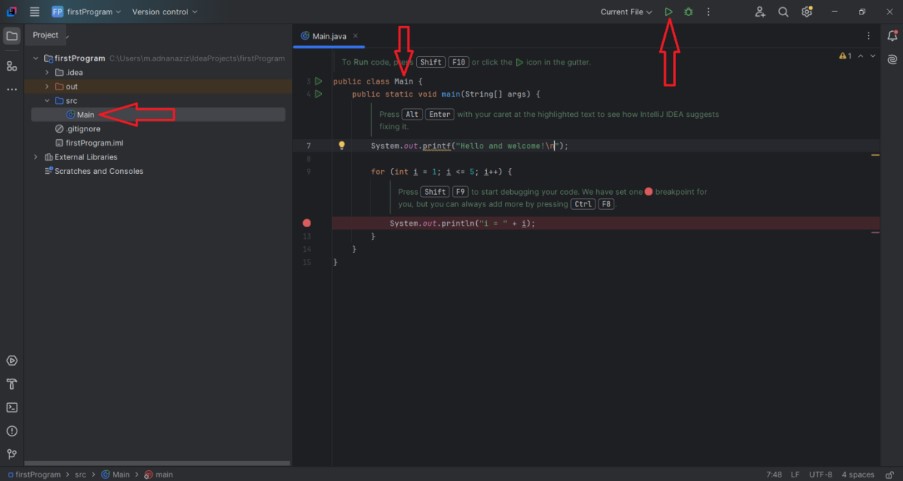
After clicking on the new project, select the project name.



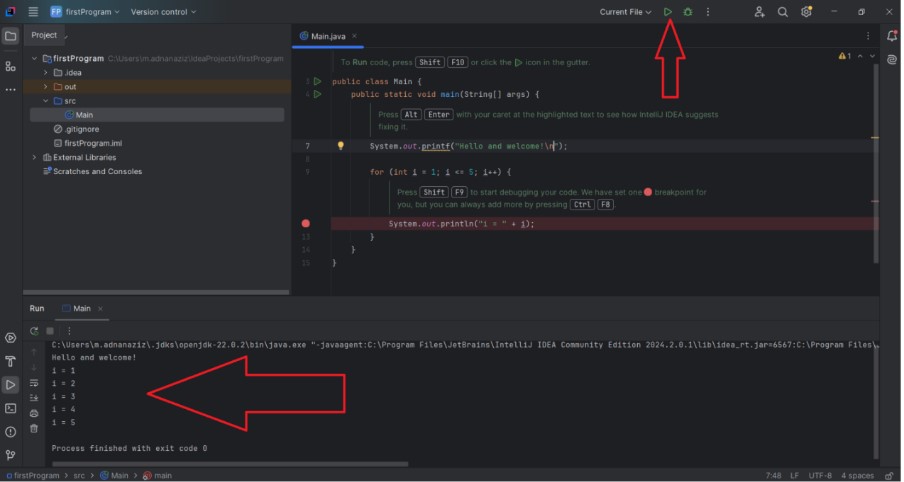
The main.java looks like this:



Mind the file name and class name are same



Run the program



### Setting the PATH for Java

|  |
| --- |
| Open IntelliJ IDEA.   * Go to File > Project Structure (or press Ctrl + Alt + Shift + S). * Under SDKs, you can find the JDK location. Note the installation directory * (e.g., C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.0.1) * Windows * Right-click on "This PC" or "My Computer" and select "Properties." * Click on "Advanced system settings."   In the System Properties window, click on the "Environment Variables" button.   * Set JAVA\_HOME: * Under "System variables," click "New" and create a variable: * Variable name: JAVA\_HOME * Variable value: Path to your JDK installation * (e.g., C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.0.1). * Edit the PATH Variable: * In the "System variables" section, find the Path variable and select it, then click "Edit." * Add a new entry: %JAVA\_HOME%\jbr\bin Apply Changes: * Click "OK" to close all dialog boxes. * Verify Setup: * Open a new command prompt and type: * java –version * You should see the Java version information if the setup is successful. |

### Basic Structure of Java

Java: Every piece of code must reside within a class. Java does not support global functions or variables.

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

### Taking input in Java

Java uses the scanner class for input handling.

In Java, after using nextInt() or nextDouble(), an additional nextLine() is required to consume the leftover newline character.

|  |
| --- |
| int x, y; int sum;  cout << "Type a number: ";  cin >> x;  cout << "Type another number: "; cin >> y; sum = x + y;  cout << "Sum is: " << sum; |

### Command Line arguments

|  |
| --- |
| class Test{  public static void main(String a[]){  String name=a[0];  System.out.println("Hello Mr "+name);  }  } |

|  |
| --- |
| class Test {  public static void main(String o[]){ int a,b,c;  a=Integer.parseInt(o[0]); b=Integer.parseInt(o[1]);  c=a+b;  System.out.println("Sum=" + c);  }  } |

### Console Input

|  |
| --- |
| import java.util.\*; class Test { public static void main(String o[]){ Scanner s=new Scanner(System.in);  System.out.println("Enter Name ");  String name=s.next();  System.out.println("Hello MR " + name);  }  } |

### Windows Programming

|  |
| --- |
| import javax.swing.\*;  class Test{  public static void main(String o[]){ String name;  name=JOptionPane.showInputDialog("Enter Name");  System.out.println("Hello Mr " + name);  }  } |

Memory Management:

Java:

Automatic memory management through garbage collection. No need for manual memory allocation or deallocation.

|  |
| --- |
| String name = "John"; |

### Primitive Data Types

Java:

Java has a more standardized set of primitive data types, and their sizes are fixed (e.g., int is always 32bit).

|  |
| --- |
| int x = 10;  float y = 5.5f; |

### Object Creation

Java:

In Java, objects are always created dynamically using the new keyword, and there are no destructors (finalizers can be used, but they are not the same as destructors).

|  |
| --- |
| MyClass obj = new MyClass(); |

### Exceptional Handling

Java:

Java has checked and unchecked exceptions. It forces you to handle exceptions through try-catch or by declaring them in the method signature.

|  |
| --- |
| try {  int result = 10 / 0;  } catch (ArithmeticException e) {  System.out.println("Division by zero");  } |

### Practice Task 01

Write, compile, and run a simple Java program (Hello World, Welcome message) to ensure the environment is set up correctly.

* Explore the basic structure of a Java program
* Create and run a Java project
* Compare System.out.println() in Java with cout in C++

Basic Syntax and Control Structures in Java

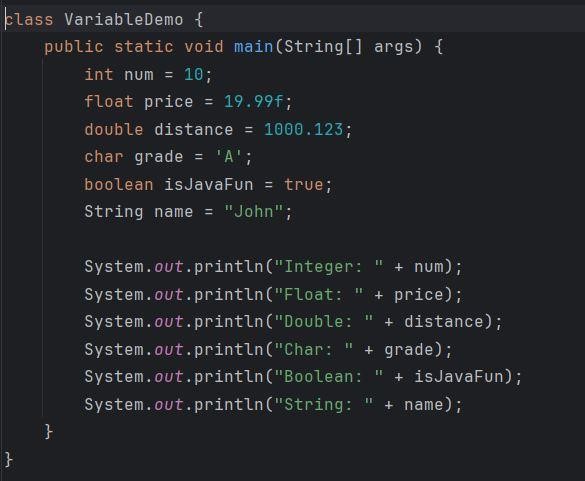
**Example Task 1:** Types of variables

Steps:

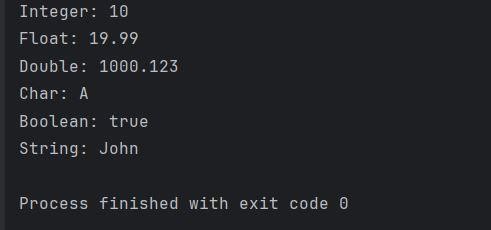
Declare variables of types int, float, double, char, boolean, and String.

Initialize them with some values.

Print the values of these variables.



Output:

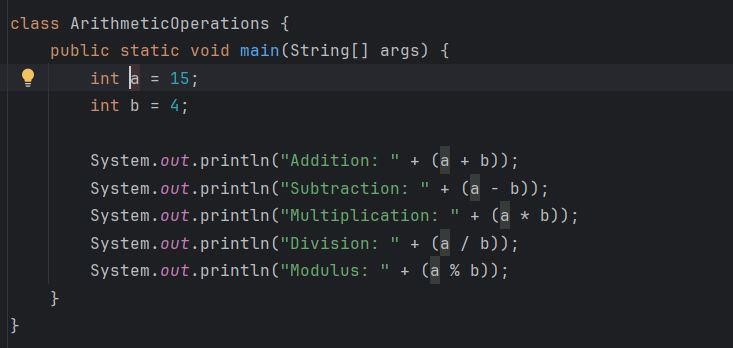


### Example Task 2: Arithmetic Operations

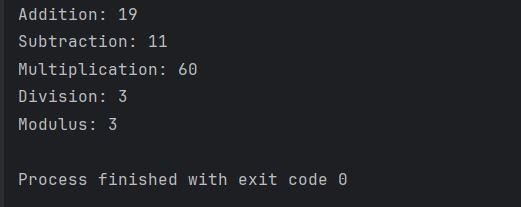
Steps:

Declare two integer variables.

Perform addition, subtraction, multiplication, division, and modulus operations. Print the results of each operation.



Output:

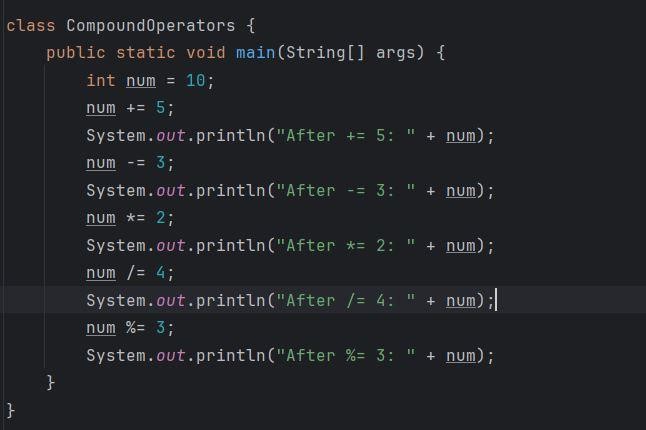


### Example Task 3: Compound Assignment

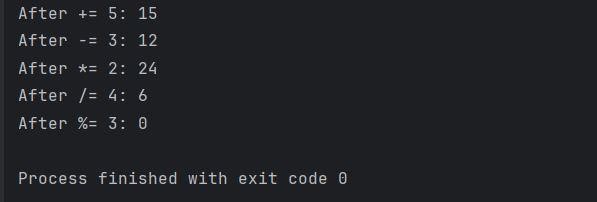
Steps:

Declare an integer variable and initialize it.

Use the +=, -=, \*=, /=, and %= operators to modify the variable. Print the updated value after each operation.



Output:



### Example Task 4: Unary Operators

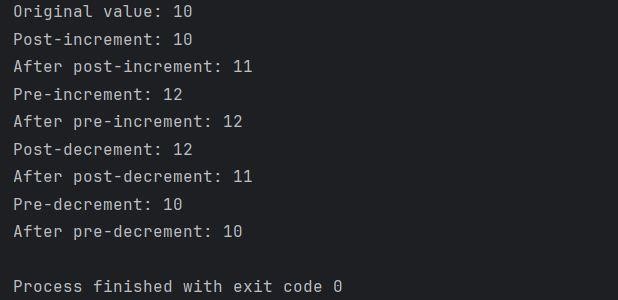
Steps:

Declare an integer variable.

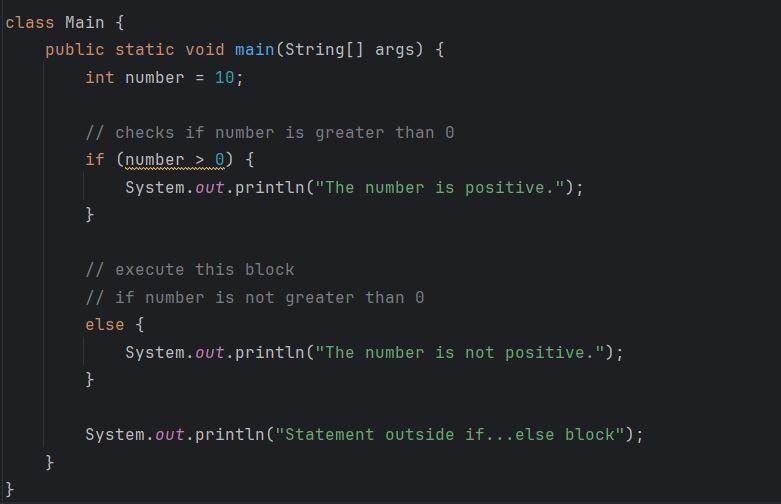
Use the increment (++) and decrement (--) operators in both pre and post forms. Print the variable before and after each operation



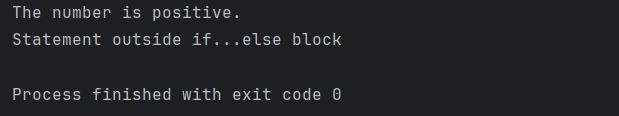
Output:



**Example Task 5: Use of Control Structures in Java**

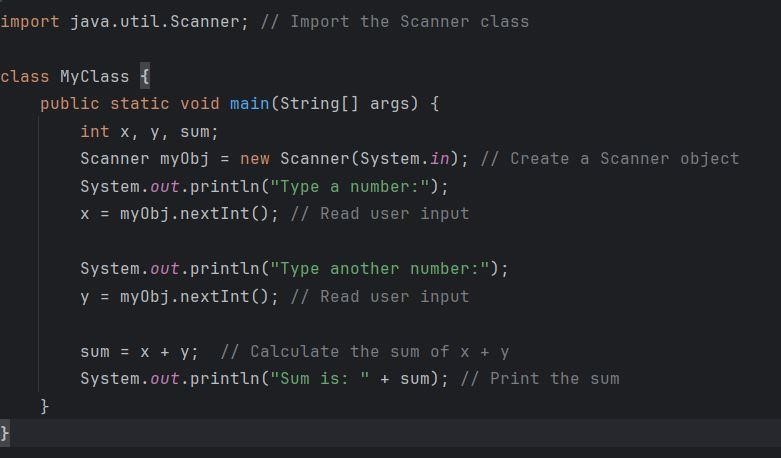


Output:

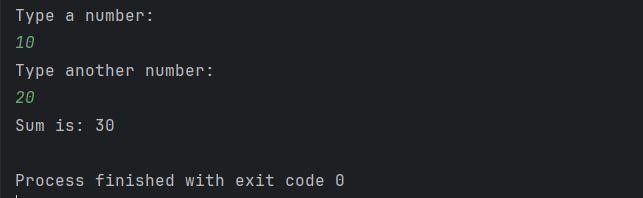


### Example Task 6: Taking Input from User

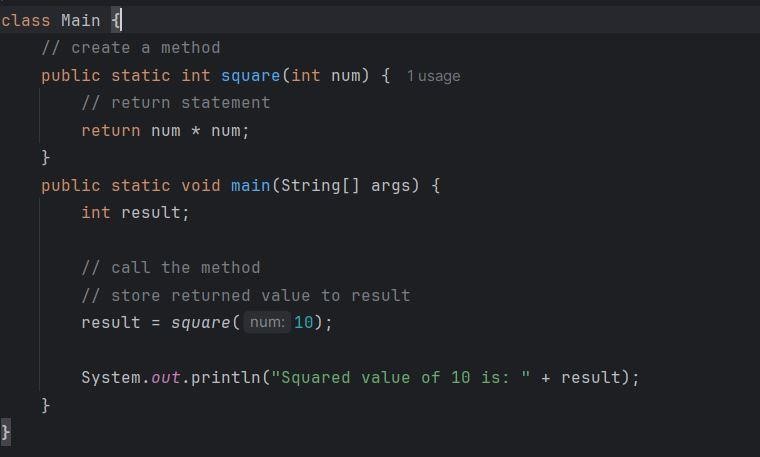
A program that defines a method to calculate and return the sum of two numbers taken as an input from the user.



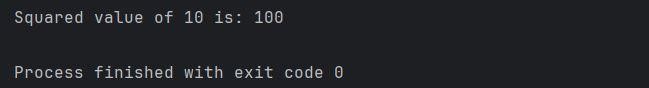
Output:



### Example Task 7: Methods with return type



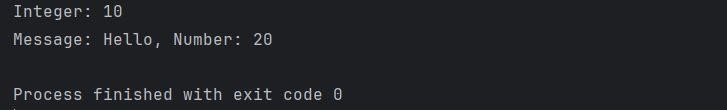
Output:



### Example Task 8: Method overloading



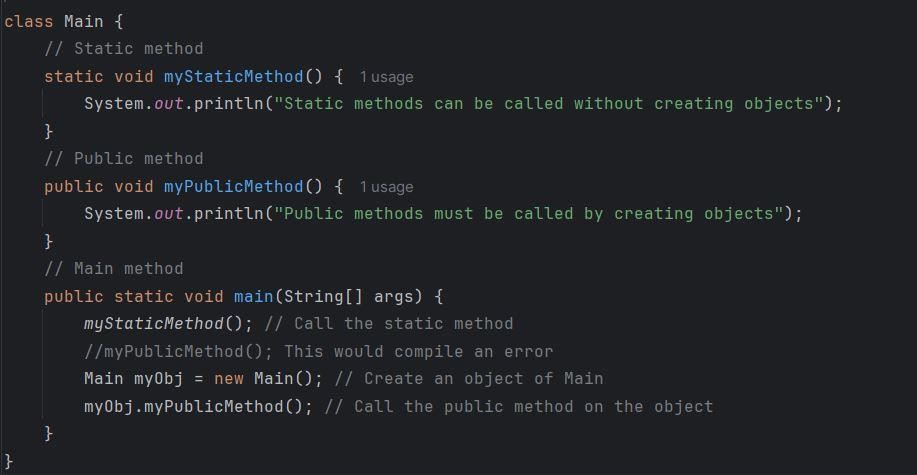
Output:



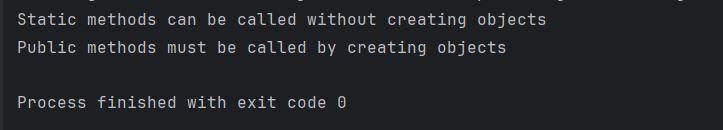
## Static Vs. Public Methods

Java programs can either have static or public attributes and methods.

In the example, a static method gets created, which means that it can be accessed without creating an object of the class, unlike public, which can only be accessed by objects:



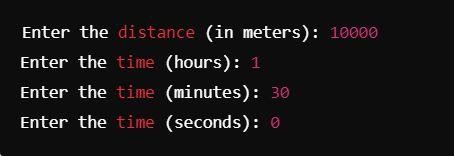
Output:



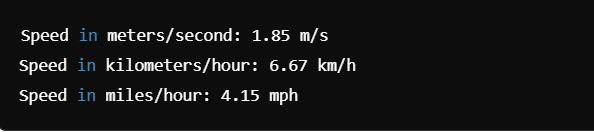
### Basic Practice Tasks

### Practice Task 01

Write a Java program that asks the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint: 1 mile = 1609 meters). Sample Input:



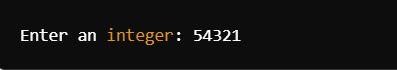
Sample Output:



**Practice Task 02**

Write a Java program to break an integer into a sequence of digits.

Sample Input:

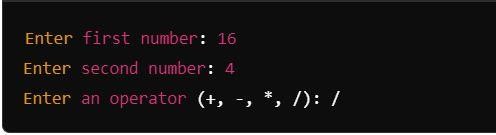


Sample Output:

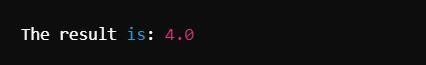


**Practice Task 03**

Write a basic calculator that performs addition, subtraction, multiplication, and division. Sample Input:

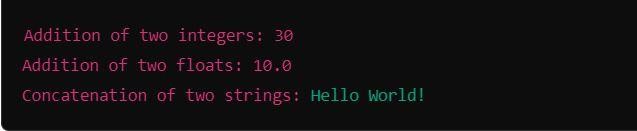


Sample Output:



**Practice Task 04**

Create a Java program with overloaded methods (e.g., to add two integers, two floats, and two strings) Sample Output:



**Practice Task 05**

Write a program that defines a method with multiple return statements to check if a number is positive, negative, or zero. Take the number as an input from the user. Sample Input:



Sample Output:



## Scenario Based Tasks

### Task 1.1: Data Types and Variables

Write a program that simulates turning a smart light on and off using variables to store its state. **Define the following:**

* boolean to store whether the light is on or off.
* int to store the brightness level (0 to 100).

**Instructions:**

* Define a boolean variable isLightOn to represent whether the light is on (true) or off (false).
* Define an int variable brightnessLevel to store the brightness (0 when off, 100 at maximum brightness).
* Use System.out.println to output the current state of the light.

|  |
| --- |
| Sample output:    Light status: Off  Brightness: 0% |

### Task 1.2: Operators and Adjusting Brightness

Use arithmetic operators to increase or decrease the brightness level. Let’s create a method to handle brightness control.

**Instructions:**

* Use the += and -= operators to modify the brightnessLevel.
* Write a method adjustBrightness that increases or decreases brightness based on a user-provided value.

|  |
| --- |
| Sample output:    Increased Brightness: 60%  Decreased Brightness: 40% |

### Part 2: Control Structures

#### Task 2.1: Using if-else for Light Control

Implement logic that turns the light on or off using an if-else statement.

**Instructions:**

* Define a boolean variable isLightOn and initialize it to false.
* Use an if-else structure to simulate turning the light on or off based on user input.

|  |
| --- |
| Sample output:    Light is now ON |

#### Task 2.2: Using switch-case for Light Modes

Simulate different lighting modes (e.g., Normal, Reading, Night) using the switch-case statement.

**Instructions:**

* Define an int variable mode to represent the lighting mode (1 for Normal, 2 for Reading, 3 for Night).
* Use a switch-case to print the current lighting mode.

|  |
| --- |
| Sample output:    Normal mode |