Programming 1

Lecture 1 – Getting started
Introduction to Java

Faculty of Information Technology
Hanoi University

Course objectives

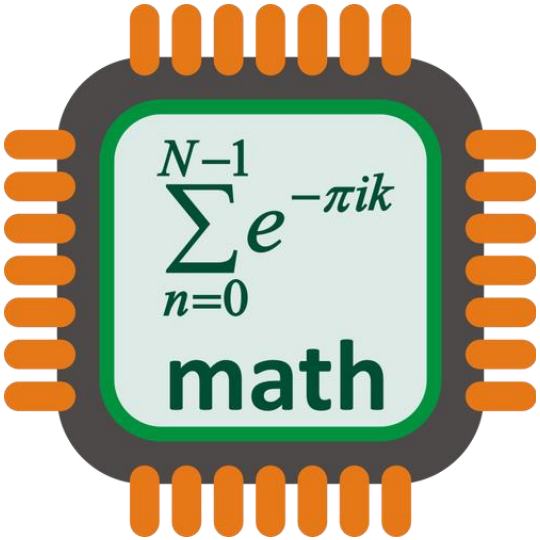
- Interact a lot
- Improve English communications
- Know how computer programs are made
- Learn basic programming concepts
- Solve simple problems with Java
- Transfer ideas/requirements into programs

Why programming?

- Programming is the core of Information Technology and Computer Science
- Create websites, apps, games, robots…
- It is the art of logic
- Programmers are needed in the industry

What does a computers do?

- Performs calculations
  - Billions of calculations per second!
- Remembers results
  - Gigabytes, Terabytes of storage!
- Computers only do
  - What you tell them
  - They're fast, but not creative



Simple computer program

- Linear Equation Solver: you enter a, b and the program shows you the solution for the equation: ax + b = 0.
- You: start the program
- Program: “Please enter A: ”
- You: type 2 and press Enter
- Program: “Okay, I got it. Please enter B: ”
- You: type 1 and press Enter

  (something happened inside the computer)
- Program: “The solution is X = -0.5”

Linear Equation Solver Program

- Start the program
- Type 2 and press Enter
- Type 1 and press Enter

- Please enter A:
- Okay, I got it. Please enter B:

(something happened inside)

- The solution is X = -0.5

What a program really is?

- Something that **interacts** with an user.
- Like a ping-pong game.
- A computer program:
  - Gets **input** from user
  - Do something with the given **input**
  - Show the **result**
  - (Optional) Repeats the above procedure
  - Terminates

Linear Equation Solver

Solve **ax + b = 0**

- Show text "Please enter number A: "
- Wait for user to enter a number
- Store this number in variable A
- Show text: "Please enter number B: "
- Wait for user to enter a number, Store in B
- Evaluate: A ≠ 0?
  - If true:
    - Calculate: X = -B/A
    - Show text: "The solution is X = "
    - Show X's value
  - If false:
    - Evaluate: B = 0?
      - If true:
        - » Show text: "The equation has infinite number of solutions"
      - If false:
        - » Show text: "The equation has no solution"

What is programming?

- Programming is
  - Writing codes to do what would be manually done otherwise.
  - Telling a computer **exactly** what to do.
  - Listing the little **steps** to achieve a goal.
- Programming is not
  - Making something out of thin air.
  - Making software to do what we don’t know how.

Cooking: Spring Rolls

- Step 1: Mince raw meat, onions, carrots, wood ears and mix together.
- Step 2: Crack a few eggs and add to the mixture.
- Step 3: Wrap the mixture in spring roll sheets.
- Step 4: Add vegetable oil to the frying pan.
- Step 5: Fry the spring rolls.

Cooking: Spring Rolls (messed-up)

- Step 1: Fry the spring rolls.
- Step 2: Add vegetable oil to a frying pan.
- Step 3: Mince raw meat, onions, carrots, wood ears and mix together.
- Step 4: Wrap the mixture in spring roll sheets.
- Step 5: Crack a few eggs and add to the mixture.

Algorithm

- An algorithm is the solution to a specific problem
- An algorithm consists of
  - A set of simple **steps**
  - A **flow of control** that specifies when each step is executed
  - A means of determining **when to stop**

➢ We use programming language to describe algorithms.

What is Java?

- A programming language
- Appeared in 1995.
- Invented by James Gosling (born 1955)
- Applications: Symbian apps, Android apps, web servers, websites, scientific apps, video games…
- Write once, runs anywhere

Java™

Why Java?

- Huge community
  - Lots of documents and tutorials.
- A neat, reasonably fast, practical language
  - Logical, easy to learn
- Multi-purpose, cross-platform
- It’s a typical Object-Oriented Programming language
- It is quite new
  - JavaScript (1996), Python (1991)
  - Perl (1987), C++ (1985), PHP (1995)

JDK, JRE, JVM

Class activity:

- What do JDK, JRE and JVM stand for?
- Describe JDK, JRE, JVM
  - What kind of software is each of them?
  - How do we use them?
- What is the latest Java version?
- Which Java versions are LTS?

Which JDK should I use?

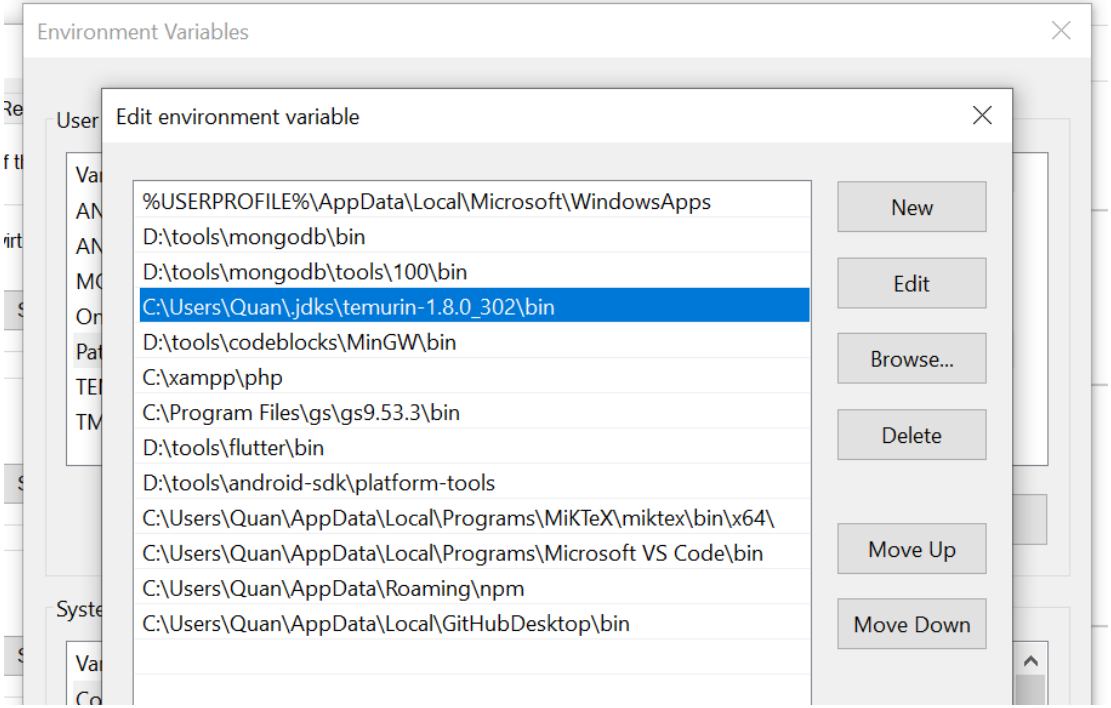
- Oracle JDK (a.k.a Java SE, paid)
- OpenJDK
  - Java SE’s community version
- Amazon Corretto
- Azul Zulu (paid support)
- AdoptOpenJDK
  - HotSpot (recommended)
  - OpenJ9

Which Java version to use?

- LTS versions (recommended)
  - Java 8
  - Java 11
  - Java 17
- Latest version
  - Java 18
  - Java 19 (coming soon)

Prepare for Java programming

- Install IntelliJ IDEA Community Edition
- Create the first Java project in IntelliJ IDEA
- Download JDK for the project
- Add *<JDK_path/bin >* to the **Path** environment variable



How to test JDK?

- My Computer -> Properties -> Advanced System Settings -> Environment Variables.
- Make sure JDK's bin directory is in the user variable or system variable named "Path"
- Open CMD, type **javac**

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\fairy25>javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
```

Hello World program

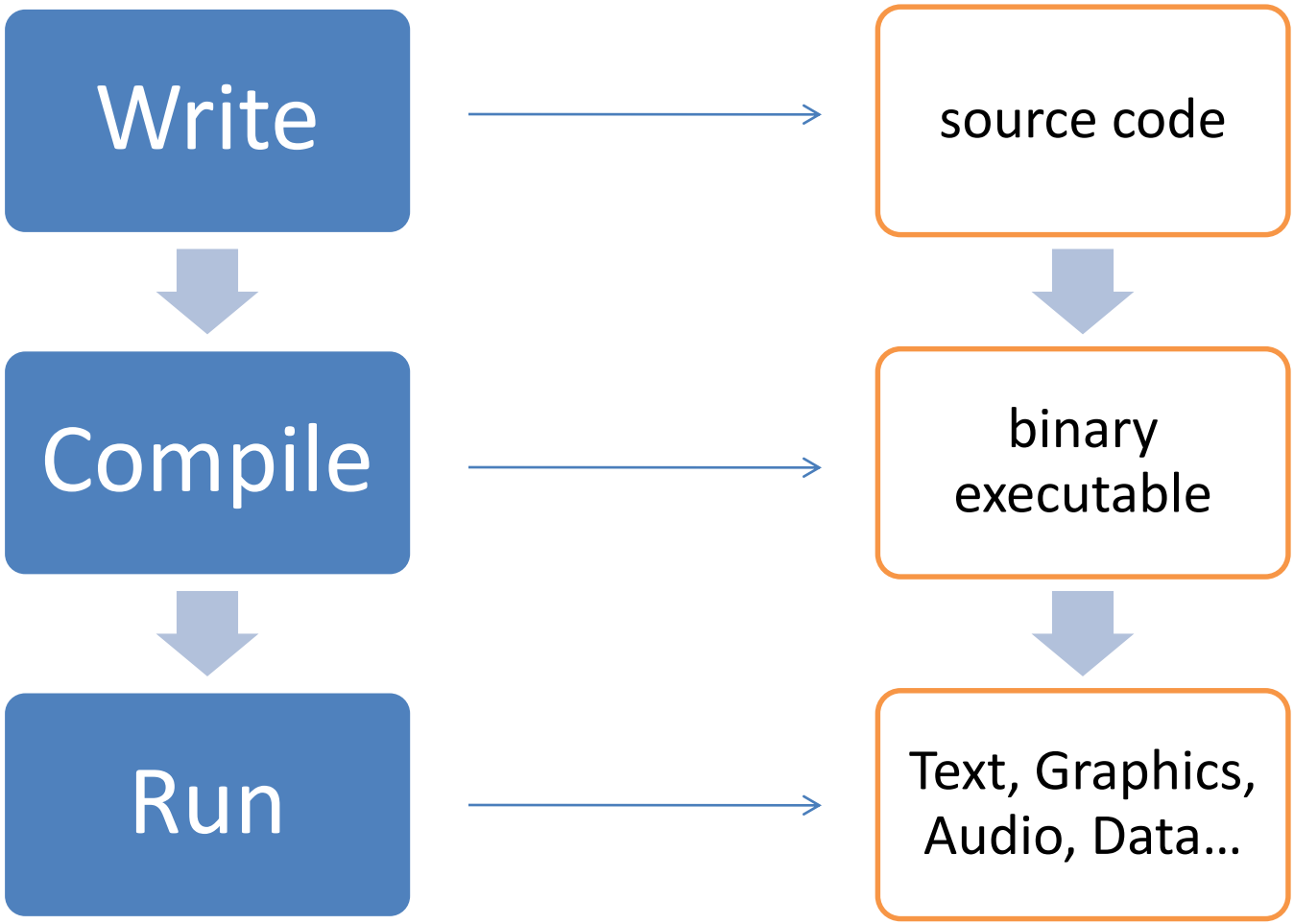
Source code of HelloWorld.java

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

Output:

```
Hello World
```

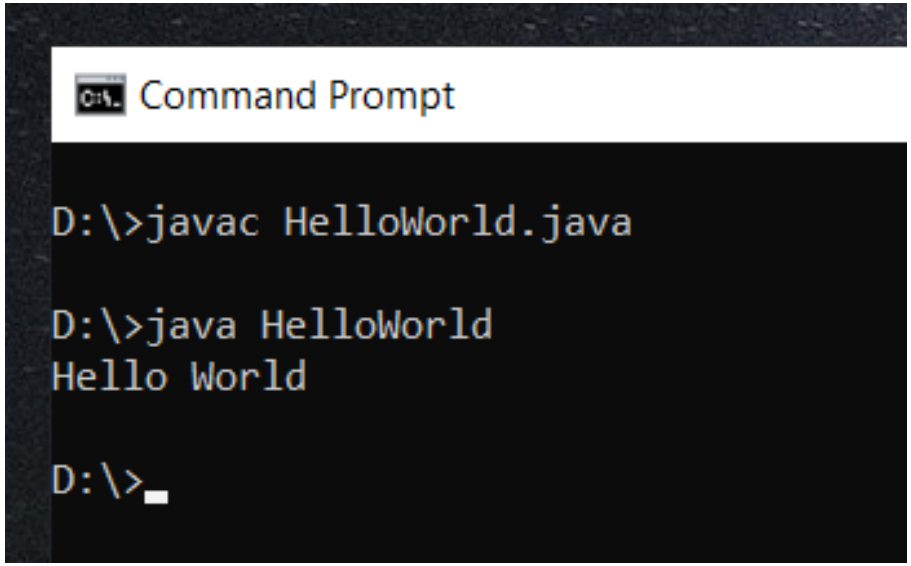
The steps of writing a program



What can go wrong?

- Syntax errors
  - Codes do not follow language rules
  - Cannot compile
- Semantic errors
  - Codes follow language rules but don’t make sense
  - Can compile but gives error when run
- Different meaning than what is intended
  - Not following algorithm
  - Program gives unexpected answer

Compile & Run with CMD



- After running **javac** command, a file named **HelloWorld.class** is generated in the same folder as **HelloWorld.java**
- By running **java <ClassName>**, the output of the program is displayed in the CMD console.

Hello World program analysis (1)

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

- `HelloWorld` is the **class name** as well as **program name**. The **file name** must be the same as **class name**, plus the **.java** extension. `(HelloWorld.java)`
- The `HelloWorld` class is **public**, it is **visible** to other classes.
- `main` is a **method**'s name. The name `main` is special, it is the method that **starts** a program.
- The **string** `"Hello World"` is surrounded by "**double quotes**"

Hello World program analysis (2)

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

- `main` method is always **`public static void`**. The name **`args`** of the argument can be changed.
- **System.out** means the **standard output**, which is the **screen** or more specifically, the **console**.

→ See https://en.wikipedia.org/wiki/System_console

Hello World program behavior

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

- When the **`HelloWorld`** program is run, its **`main`** method is **called** first of all.
- When a method is called, all of its statements **execute**. Here, there is 01 **statement** in the **`main`** method.
- Let's ignore the **`String[] args`** part.

Notes on Java syntax

- If there is a `{`, there should be a `}` to close it.
  (curly brackets, open brace, close brace)
- The same applies to `(…)` and `[…]`
  (parentheses, square brackets)
- What is opened earlier should be closed later.
  (one thing wraps around another thing)

```
{ codes { other codes } }
```

Structure of the first program

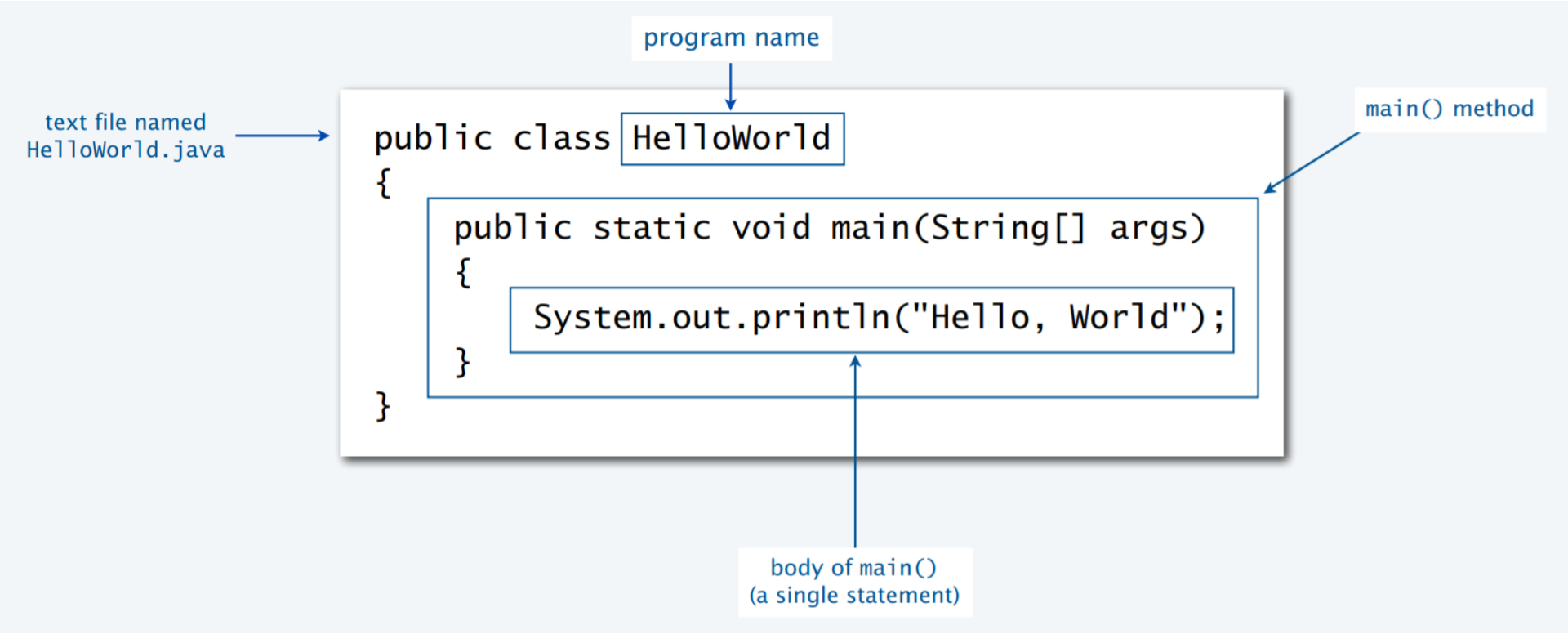


Image Credit: **R. Sedgewick**

Three versions of the same program.

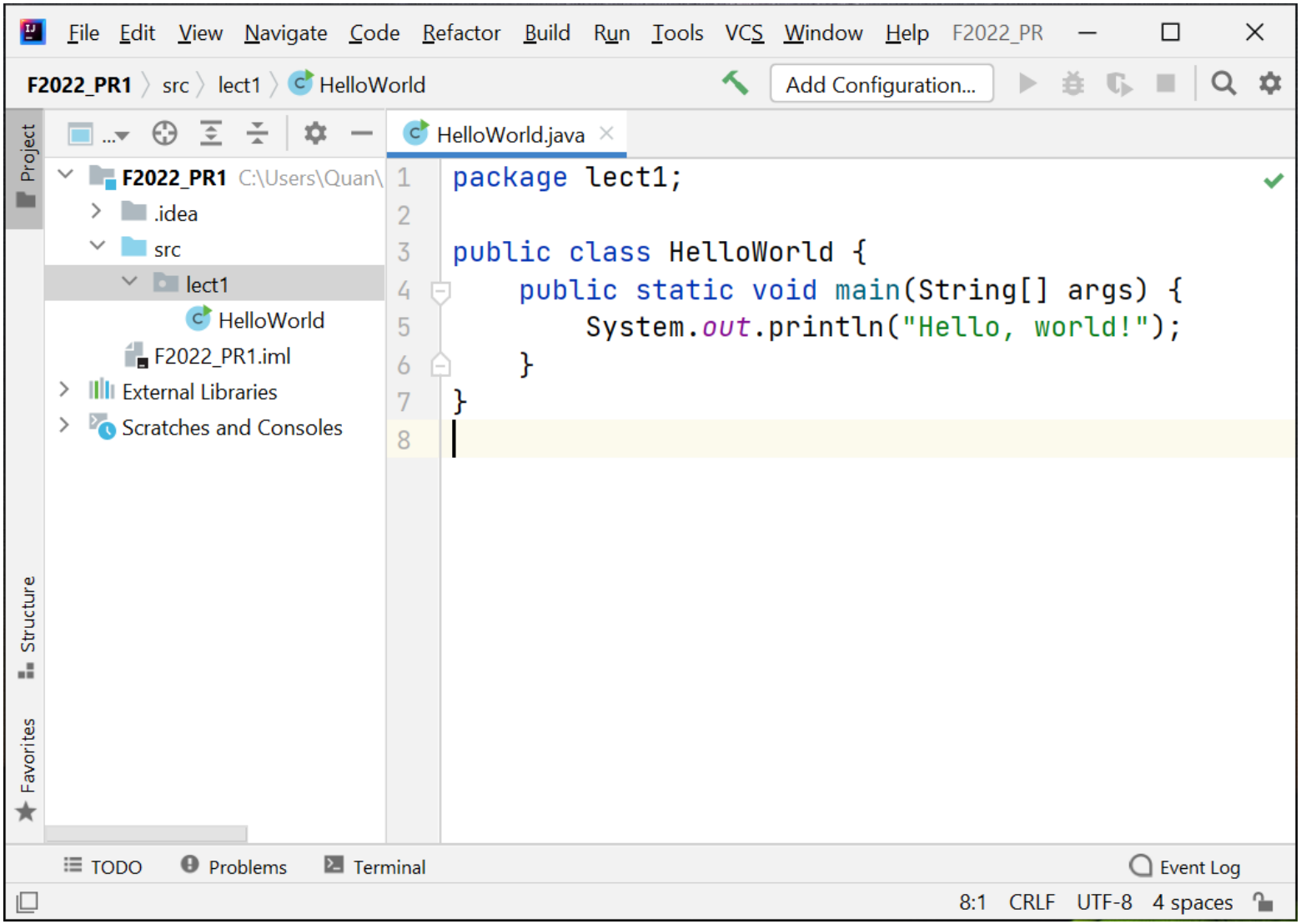
```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World");
    }
}
```

```
/*************************************************************************
 *  Compilation:  javac HelloWorld.java
 *  Execution:    java HelloWorld
 *
 *  Prints "Hello, World". By tradition, this is everyone's first program.
 *
 *  % java HelloWorld
 *  Hello, World
 *
 *************************************************************************/

public class HelloWorld {

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

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

Fonts, color, comments, and extra space are not relevant in Java

Live instructions: IntelliJ Idea



Demo exercise 1

- Write a program to calculate the sum/product/quotient of two integers and display the result.

Demo exercise 2

- Write a program to display the following shape:

```
*
* *
* * *
* * * *
* * * * *
```

Reading Resources

- R. Sedgewick - Introduction to Programming in Java
  - Online: https://introcs.cs.princeton.edu/java/home/
- Oracle's Java Tutorials:
  - https://docs.oracle.com/javase/tutorial/
- W3Schools Java Tutorial:
  - https://www.w3schools.com/java/default.asp