

Topic 1.1: Introduction to Algorithms, Programming, and Compilers

LO: 1.1.A, 1.1.B, 1.1.C | Skill: 1.A | Scenario: Algorithms in everyday life — then your first Java program

Guided Notes — Topic 1.1

Fill in the blanks during the slide presentation. Use exact terms from the slides.

Section 1 — Algorithms and sequencing

An **algorithm** is a step-by-step process for completing a task or solving a problem. It can be written in plain **language** or drawn as a **diagram**.

Sequencing defines the order of the steps. The steps are completed **one at a time**, and changing the order can change the result.

Section 2 — From code to a running program

Code can be written in any text **editor**, but programmers often use an integrated development environment, or **IDE**, because it provides tools to write, compile, and run code in one place.

A **compiler** checks the code for certain errors. Any error the compiler detects must be **fixed** before the program can **run**.

A first Java program looks like this:

ClubWelcome.java

```
public class ClubWelcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to the Riverside Coding Club!");  
    }  
}
```

Section 3 — Three kinds of programming errors

1. **Syntax error**. A mistake where the rules of the language are not followed (for example, a missing semicolon). It is detected by the **compiler**, so the program will not run until it is fixed.

2. **Logic error**. The program runs but behaves incorrectly. It is found by **testing** the program with specific data to see if the output is what you expected.

3. **Run-time error**. A mistake that happens during **execution** and usually causes the program to stop abnormally. A run-time error caused by an unexpected condition the compiler did not catch is called an **exception**.

Quick check

A missing semicolon is caught before the program runs. A program that prints the wrong total but does not crash has a logic error.