Chapter 1 introduces fundamental computer concepts, covering hardware (CPU, memory, I/O), software types, and networking basics.

It distinguishes digital vs. analog systems and explains the Java language's platform independence via the JVM.

We learn about syntax vs. semantics and common error types in program development.

Java's object-oriented approach uses classes, objects, inheritance, and polymorphism.

A straightforward problem-solving process concludes the chapter.

Chapter 2 focuses on handling data and expressions in Java:

1. **Character Strings**
    - Defined by the `String` class and written in **double quotes**.
    - The `+` operator concatenates strings or appends numbers to strings.
2. **Variables and Assignment**
    - A **variable** is a named memory location declared with a type (e.g., `int total;`).
    - **Initialization** can set an initial value (e.g., `int sum = 0;`).
    - Use `static final` for **constants** (unchanging values).
3. **Primitive Data Types**
    - Java provides **eight** primitive types: `byte`, `short`, `int`, `long`, `float`, `double`, `char`, and `boolean`.
    - **Integers** differ by size (e.g., `int` vs. `long`); **floating-point** types are `float` and `double`.
    - **char** stores a single character (using Unicode).
    - **boolean** holds `true` or `false`.
4. **Expressions and Operators**
    - Arithmetic operators: `+`, `-`, `*`, `/`, `%`.
    - **Integer division** truncates fractional parts.
    - Operator **precedence** affects evaluation order (multiplication/division before addition/subtraction).
    - The **assignment operator** (`=`) stores the evaluated result into a variable.
    - **Increment/decrement** (`++`, `--`) and **compound assignment** (e.g., `+=`) provide shortcuts.
5. **Data Conversion**
    - **Widening** (safe) and **narrowing** (may lose data) conversions.
    - **Assignment conversion** and **promotion** happen automatically when compatible.

    - **Casting** (e.g., `(float)`) explicitly converts data types.
6. **Interactive Programs**

    - Use `Scanner` (from `java.util`) to read input (e.g., `nextLine`, `nextInt`).
    - `System.in` represents keyboard input.

Overall, Chapter 2 covers how to represent and manipulate data using variables, operations, and user input in Java.