1. **Computer Processing (Chapter 1.1)**

    - A computer system is composed of **hardware** (physical components like CPU, memory, and I/O devices) and **software** (the programs and data used by the hardware).
    - Key hardware components include:
        1. **CPU (Central Processing Unit)**: Executes program commands.
        2. **Main Memory (RAM)**: Temporarily holds active programs and their data.
        3. **Secondary Memory**: Permanently stores data (e.g., hard drives, SSDs).
    - **Operating Systems** manage hardware resources and provide a user interface. **Applications** are all other software.
2. **Digital vs. Analog**

    - **Digital** systems store and process information as discrete values (bits), whereas **analog** systems store information in a continuous form.
    - Modern computers are inherently digital, using binary (base-2) to represent all data.
3. **Hardware Components (Chapter 1.2)**

    - **Main Memory** consists of addressable locations, each storing data. It is typically volatile (data lost if power is off).
    - **Cache** is a small, faster memory area used by the CPU for quick access to frequently needed data.
    - **ROM (Read-Only Memory)** stores firmware needed for boot processes.
    - **CPU Architecture**:
        1. **Control Unit** coordinates operations.
        2. **Arithmetic/Logic Unit** performs computations and logic.
        3. **Registers** store data and instructions being used immediately.
        4. **System Clock** governs the CPU's operational pace in GHz.
4. **Networks (Chapter 1.3)**

    - A **network** connects two or more computers to exchange data.
    - **LAN (Local Area Network)** covers a small area; **WAN (Wide Area Network)** can span large distances and often connects multiple LANs.
    - The **Internet** is a global WAN. Its foundational protocol is **TCP/IP**, and each machine has a unique **IP address**.
    - **The World Wide Web** runs over the Internet, relying on browsers and servers to deliver documents, typically written in **HTML**.
    - **URLs** specify the protocol, domain, and resource path for accessing web content.
5. **Java Programming Language (Chapter 1.4)**

    - Java organizes software into **classes**, containing **methods**, which contain **statements** and **expressions**.
    - Every Java application has a `main` method that serves as the entry point.
    - Java code relies on the **Java API** (standard class library) for common functionality.

- Java is compiled into **bytecode**, which the **Java Virtual Machine (JVM)** executes, making the language platform-independent.

6. **Program Development (Chapter 1.5)**

    - **Editors** or **IDEs** are used to write code, which is then **compiled** (or interpreted).
    - **Syntax**: Defines valid code structure; errors here prevent compilation.
    - **Semantics**: Defines what statements actually do during execution.
    - **Error Types**:
        1. **Compile-time errors** (usually syntax/type errors).
        2. **Run-time errors** (crashes/exceptions during execution).
        3. **Logical errors** (incorrect results despite successful execution).

7. **Object-Oriented Programming (Chapter 1.6)**

    - **Objects** encapsulate data (attributes) and behaviors (methods).
    - **Classes** are blueprints for objects; multiple objects can be instantiated from one class.
    - **Inheritance** lets a class reuse and refine the structure/behaviors of a parent class.
    - **Polymorphism** allows objects of related classes to be used interchangeably in code.

8. **Problem Solving**

    - Generally follows these steps:
        1. Understand the problem and its context.
        2. Design a solution by breaking it into manageable parts.
        3. Consider possible alternatives and refine the chosen approach.
        4. Implement the solution (i.e., write the code).
        5. Test thoroughly and fix any errors or logical flaws.

Overall, the notes introduce the foundations of computer systems (hardware, software, networking), then explain Java's structure and compilation process, and conclude with the essentials of object-oriented programming and the broader problem-solving process in software development.