Chapters 1 Introduction

Chapter 1 Flash Cards

1.1: Computer Processing

A computer system consists of hardware and software that work together to help solve problems.

A **program** is a series of instructions that the hardware executes one after another

Software consists of a program paired with the data the program uses. Software is the intangible counterpart to the physical hardware components.

Key Hardware Components

CPU: A device that executes the individual commands of a program

I/O Devices: Things like mouse, keyboard, monitor, etc that allow a human being to interact with the program

Main Memory (RAM): A storage device that holds the software and data that is used by the CPU while it is being processed. The CPU reads program instructions from the main memory, executing them one at a time until the program ends

Secondary Memory (Drives): Devices that hold software and data in a relatively permanent manner.

Software Categories

Operating System: Core software of a computer. The operating system…

- provides a user interface that allows the user to interact with the machine.
- manages computer resources such as the CPU and main memory
    - determines when programs can run
    - determines where programs are loaded to memory
    - determines how hardware devices communicate

Application: Most software that is not an operating system. Applications provide a UI (typically a GUI in most modern operating systems) to interact with a program.

The interface is an important part of the software, because to the user, it *is* the program.

Digital Computers

Two fundamental techniques are used to store and manage information: analogue and digital.

analogue: continuous, in direct proportion to source of information. Eg. alcohol thermometer, electronic signal used to represent vibrations of sound wave (voltage is proportional to original sound wave.)

digital: Basis of all modern electronic computers. breaks information into discrete pieces and represents them as numbers. Lossy, relative to analogue signals. With a high enough sampling rate, this can create a signal that is “close enough” to the original to satisfy human senses.

Binary Numbers

The **base value** of any number system indicates how many digits it has and the place of each digit in a number.

Binary: Base-2 numeric system (0 and 1). Used to store all information in a computer.

A single binary digit is called a **bit**.

Computers use binary due to the nature of the hardware. For example, HDD hard drives are magnetic, using polarized magnetic material to store binary data. Other computer memory is made up of tiny electrical circuits which switch between just two states.

Due to the base 2 nature of a binary system, any digital signal is **discrete**, ie a ‘stepped’ curve, alternating between two voltage levels.

There are exactly 2^n permutations of N bits. Therefor N bits can represent up to 2^n unique items.

For example, 2 bits have 4 permutations and so can represent 4 discrete values. 3 bits can represent 8, etc etc.

1.2 Hardware Components

computer architecture: How the hardware components of a computer are put together. Information travels between components across a group of wires called a **bus**

The CPU and main memory make up the core of a computer. Almost all other devices in a computer system are called **peripherals**

Controllers: devices that coordinate the activities of specific peripheral

I/O devices and **secondary memory devices** are considered peripherals.

Data transfer devices are another category of peripherals that allow information to be transmitted between computers.

Input/Output Devices

Input: most commonly keyboard and mouse/trackpad, but many others.

Output: most commonly monitors and printers, but many others

Some devices provide both, such as touch screens.

Main Memory

Main memory is made up of a series of small consecutive memory locations, identified with a unique number called an **address.**

Whenever data is stored in a memory location, it replaces whatever was there before.

On many computers, each memory location consists of eight bits, also called one **byte**. Often multiple consecutive bytes are needed to store larger data.

The **storage capacity** of a device is the total number of bytes it can hold. This is typically represented using larger units of measure, such as MB, GB, TB, etc

A computer with larger main memory allow larger and/or more programs to run effectively, as information does not need to be retrieved from secondary memory as often.

Main memory is usually **volatile** meaning the data stored in it will be lost if the computer loses power. Secondary memory devices are almost always non-volatile.

The **cache** is a small fast memory that stores the contents of the most frequently used main memory locations. It is used by the CPU to reduce average access time of data.

A hard disk and USB drive are both **direct access devices** (also known as random access) since the needed information can be accessed directly

Older technologies like magnetic tapes used to store data are considered **sequential access** devices

ROM (Read-Only Memory):Special direct access chip on the computer motherboard used to store software called BIOS (basic input/output system) that provides preliminary instructions needed when the computer is initially turned on.

The Central Processing Unit

The CPU interacts with the main memory to perform all fundamental processing in a computer. The CPU interprets and executes instructions in a continuous cycle.

A CPU is made up of three important components:

the **control unit** coordinates the processing steps within the CPU and the transfer of data and instructions from the main memory and the registers.

the **registers** provide a small but extremely fast cache on the CPU itself. In most CPUs, some registers are reserved. For example, the **instruction register** holds the current instruction being executed. The **program counter** holds the address of the next instruction to be executed.

the **arithmetic/logic unit** performs calculations and makes decisions.

Modern CPUs follow a fetch → decode → execute cycle:

1. **fetch** an instruction from main memory at the address stored in the program counter, and put it into the instruction register.
2. **decode** the instruction electornically to determine which operation to carry out, and increment the program counter.
3. **execute** the instruction by activating the correct circuitry to carry it out, such as loading a data value or performing arithmetics.

The CPU is located on a chip called a **microprocessor**. It also contains ROM chips and communication sockets to which device controllers can be connected.

Another important component on the microprocessor is the **system clock**, which generates an electronic pulse at regular intervals, which synchronizes the events of the CPU. The rate at which the pulses occur is called the clock speed, which is measured in GHz. For example a processor with a clock speed of 3.1 GHz pulses approximately 3.1 billion times per second.

Modern CPUs have multiple **cores** or processors, allowing them to do two or more things at once. Software must be written in a specific way to take advantage of more than one core, which is called multithreading.

1.3 Networks

Area Networks

A **network** consists of two or more computers connected together so that they can exchange information.

Each computer on a network has a **network address**, which uniquely identifies it.

point-to-point connection: When two or more physically close computers are connected physically with wires.

This is simple and cost effective, but has limitations. It is infeasible to connect many computers to one another using a separate line to/from each computer. If many computers all share one communication line, it can introduce delays, as computers need to take turns sending messages.

To reduce delays in network communication, it is possible to divide large messages into segments, called **packets**.

local area network (LAN): Network spanning short distances to connect a relatively small number of computers (within one room or building).

Convenient affordable and reliable, but range-limited.

wide area network (WAN): Connects two or more LANs, potentially over large distances. Usually one computer on each LAN is responsible for handling communication across the WAN.

The Internet

The **Internet** is a network of networks, connecting many small networks together. It is a WAN that spans the globe.

A **protocol** is a set of rules that governs how two things communicate. Software sends messages across the internet must conform to a protocol called TCP/IP. (Transmission Control Protocol / Internet Protocol)

Each computer connected to the internet has a uniquely identifying IP address

A computer can also be accessed using a unique internet address, for example hector.vt.edu

hector - the local name of a computer

.vt.edu - the domain name

edu - the top-level domain (TLD)

When an internet address is referenced, it gets translated to an IP address using a system called the Domain Name System (DNS). Each organization connected to the internet maintains a **domain server**: a list of all computers at the organization and their IP addressed

The World Wide Web

Software that makes sharing information across a network easy for humans.

The internet is the foundation of the web, but the web makes using the internet straightforward and enjoyable.

The web uses software called a **browser**.

A computer dedicated to providing access to Web documents is called a **web server**

Most documents on a web server are formatted using HyperText Markup Language (**HTML**)

Uniform Resource Locators

A **URL** is a unique address used by a browser to access specific documents and other information to display (a **website**).

ex: http://www.google.com

anatomy of a URL:

protocol: determines the way the browser transmits and processes information, ex. HTTP

internet address: machine on which the document is stored. “www” is a typical reference to a Web server.

file name or resource of interest: directs to a specific resource. If not specified, the home page is usually provided.

1.4 The Java Programming Language

Software is written in a **programming language**: a set of rules that dictate how the words and symbols can be arranged to form valid **program statements**.

In Java:

- A program is made of one or more classes.
- A class contains one or more method
- A method contains **expressions** and **statements**

expressions evaluates to a value

statements do not evaluate to a value

A method or class declaration is **not** a statement

The Java language is accompanied by the Java API, also known as the standard class library, which provides important functionality.

There are three main streams of development for Java:

- **Standard Edition**
- **Enterprise Edition**, which includes libraries to support large scale development
- **Micro Edition**, which is for developing software for mobile

A Java Program

```
//

// Demonstrates the basic structure of a Java application.

//********************************************************************

public class Lincoln

{

//-----------------------------------------------------

--------

// Prints a presidential quote.

//-----------------------------------------------------

--------

public static void main(String[] args)

{

System.out.println("A quote by Abraham Lincoln:");
```

```
System.out.println("Whatever you are, be a good one.");

}

}
```

Comments

First few lines of the program are comments.

Comments should be used only when needed to explain the purpose of the program/code

Using comments is a minor failure. Its better to just have the code self-explanatory

Comments must be maintained. It is possible that comments could become out-of-date, providing wrong information, which is actively harmful.

```
// this is a comment


/*

this is also a comment

*/


/** this is a javadoc comment */
```

Class Definition

The rest of the code is a **class definition**.

The **class** is called Lincoln. It runs from the first to last curly brace.

The class has one method called **main**

Every Java program must have one main method, which executes when the program is run.

A *main* method header always includes “public static void”, which are all Java keywords that define the properties of the method.

Identifiers

Identifiers are the ‘words’ of a program.

The identifiers in the Lincoln program are:

class, Lincoln, public, static, void, main, String, args, System

These fall into three categories:

- words that we make up when writing a program (Lincoln and args)
- words that another programmer chose (String , System , out , println , and main)
- words that are reserved for special purposes in the language (class , public , static , and void)

Identifier Rules

A java identifier can use letters, digits, and the _ and $ characters.

‘letters’ and ‘digits’ are flexible terms that encompasses many human languages, not just english alphanumeric.

Identifiers cannot begin with a digit.

Java has reserved words, which also cannot be used as identifiers

Java also has contextual keywords which can *sometimes* be used as an identifier (but you probably shouldn’t)

These were keywords added later and thus were not made technically reserved to avoid breaking old code

Java is case sensitive (Blame C for starting this convention.)

Conventions

- Classes: title case. ex. *Lincon*
- constants: upper case. ex. MAXIMUM

Identifiers can be any length. Avoid overly verbose identifiers, and meaningless ones, such as arbitrary single letters.

White Space

spaces, blank lines, and tabs are called white space.

White space is ignored by the java compiler. Its just for look and feel.

Good use of white space enhances readability, such as by using consistent indentation.

1.5 Program Development

Programming languages can be categorized into 4 groups:

- machine language
- assembly language

- high-level languages
- fourth-generation languages.

Machine language is written in binary and is essentially not human readable.

Regardless of what language it is written in, a program must be translated into machine language to run on a computer.

Each type of CPU has its own specific machine language.

Machine language is simple, with each statement doing a single operation.

Assembly language is a human readable expression of machine language. It retains the quality of being composed of simple statements that perform a single operation.

Machine language and assembly are both low level languages.

High-level languages like Java are expressed in english like phrases, and allow much more complex logic to be expressed in a single statement. Most languages used today are in this category.

fourth-generation languages feature even more abstraction and automation than high-level languages. Instead of writing code to accomplish goals, simply state the goals and the code is generated. SQL is an example of this.

Editors, Compilers, and Interpreters

Editors are software used to write programs. They have features that make programmers lives easier, like line number and colour coding.

Complex editors called **integrated development environments (IDEs)** provide additional advanced tools, such as compiling Java code in the background before the code is saved or run.

Compilers translate source code into machine language. Java has a special machine language called bytecode.

Java bytecode is nice because it is system neutral. It runs on any system with a Java Virtual Machine, regardless of OS.

Java is therefor considered architecture-neutral

Bytecode is slower than languages that are natively compiled, but it is pretty close.

Interpreters combine the tasks of compiling and execution, essentially compiling as they go. Some languages, like Python, are interpreted languages, meaning they do not require compiling before they are run. Interpreted languages are generally slower than compiled languages.

Syntax and Semantics

All programming languages have their own unique **syntax** - essentially the grammar of the language. Using incorrect syntax produces code that will not run or compile.

The **semantics** of a statement define the *meaning* of a statement - what will happen when a statement is executed. This is rarely ambiguous in programming languages. There should be only one correct interpretation of a statement.

A program will always do what we tell it to do. but maybe not what we *meant* to tell it to do.

Errors

There are generally three broad categories of computer errors:

- compile-time errors
- run-time errors
- logical errors

Compile-time errors are usually the result of incorrect syntax, but could also result in semantic problems, such as using incompatible types.

Run-time errors also known as “crashing” ****is the result of a problem during execution, such as dividing by zero. Often, these result in an **exception:** predictable error types, which can be caught and dealt with in a well written program.

Logical errors are the result of code which executes without complaint, but produces the wrong result. These are the easiest to ignore and often the hardest to fix.

1.6 Object-Oriented Programming

Java is an **object-oriented programming language**, meaning all code is contained within classes, which are instantiated as **objects** at runtime.

For example: in real life, a car is an object. The car has attributes, such as weight and colour, and methods, such as drive and brake.

In addition to objects, Java also has **primitive objects:** core building blocks such as numbers and characters.

Classes are the blueprints for objects. When we write a program, we never actually write an object directly. We write classes, which define the properties and behaviours an object will have. During runtime, the objects are “instantiated” using the class as a template.

Multiple objects created from the same class often have different states from one another.

An object should be **encapsulated**, meaning it protects and manages its own information.

Classes can inherit and build on or modify the attributes and methods of other classes. this is called **inheritance**.

In Java a class can have only one parent. This constraint can be worked around using interfaces.

polymorphism builds on the idea of inheritance: essentially it is the idea that an object inheriting from a parent class can be used in place of the parent class in a program.

Problem Solving

The purpose of writing a program is to solve a problem.

In general, problem solving consists of multiple steps:

1. Understanding the problem.
2. Designing a solution.
3. Considering alternatives to the solution and refining the solution.
4. Implementing the solution.
5. Testing the solution and fixing any problems that exist.

These steps are non-linear. They overlap and interact.

Understanding the problem is essential to avoid misguided solutions. Each problem has a **domain** - the necessary information needed to understand the problem.

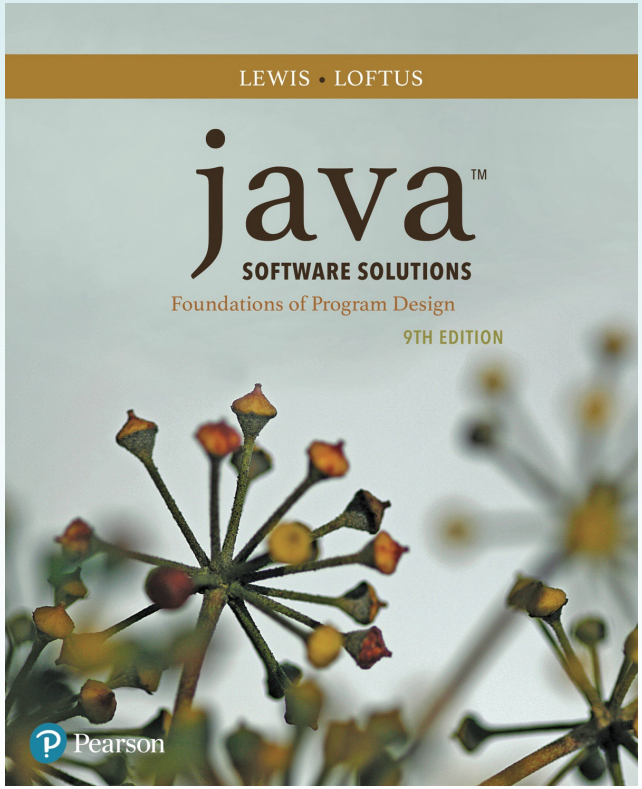
Once a problem is understood, the best way to design a solution is to break it down into manageable pieces.

Avoid temptation to immediately commit to the first solution you think of. Often alternatives can be found that are superior, but the earlier they are considered, the more easily they can be integrated.

Implementing the solution is first-and-foremost designing the solution. The actual writing of code is secondary to this.

Throughout implementation, software testing should be used to catch as many errors as possible.

Chapter 1
Introduction



Java Software Solutions
Foundations of Program Design
9th Edition

John Lewis
William Loftus

Focus of the Course

- Object-Oriented Software Development
  - problem solving
  - program design, implementation, and testing
  - object-oriented concepts
    - classes
    - objects
    - encapsulation
    - inheritance
    - polymorphism
  - graphical user interfaces
  - the Java programming language

Instructor

- Bruce Link
- Ph.D. Computer Science
- 5 years in US research laboratory
- 18 years at MacDonald Dettwiler
  - Project engineer, project manager, engineering manager
- Option Head Information Systems, BCIT
- Contact Methods
  - 412-7508, SW2/365
  - bruce_link@bcit.ca
- Course content:
  - learn.bcit.ca Comp1510 course (Learning Hub or D2L)

Course Outline: *Read It*

- Evaluation methods
- Required texts
- Course marking guidelines and details
  - quizzes, assignments, midterm, final
- Lectures
- Labs
- Policies and procedures

Expectations

- Polite and civilized behaviour
  - Respect for classmates and instructors
  - One person at a time talking
  - No cell phones
  - No latecomers
  - Feel free to remind your classmates
- Lots of hard work and learning
  - Take responsibility for *your* learning
  - Read the assigned reading before lecture
  - Look at the assignments when they are posted
- Fun and Magic
  - Programming is creating something from nothing

How Much Effort to Put Into This Course?

- This is the foundation for the other programming courses in CST
  - More Java, C, C++, C#
- *All* second year options have programming content
- Comp 1510 is a prerequisite to three second term courses
  - Comp 2510 C, Comp 2526 Java II, Comp 2721 Architecture
  - You will be blocked from these without this course
- Be sure to master this course!!!

How Much Effort to Put Into This Course?

- Expect learning *may* be hard
  - You are rewiring your brain with new habits
  - If you have prior knowledge you may need to unlearn it
- Choose to make the commitment to put in the required effort
  - Burdens freely chosen are lighter
- Programming is learned by doing
  - You will have times of frustration before finding solutions to problems
  - This is normal – never give up
  - You are also learning to work hard

Train and Tame your Mind

- Software development is a mental activity
  - Need sound mind in sound body
- General mental development can help you succeed
- Mindfulness training and meditation
  - BCIT mindfulness workshops and practice
  - Sam Harris' Waking Up app
  - Help with eating, smoking, anxiety
- Self knowledge
  - Where you are coming from; where you are going
  - Your Big-5 personality
  - Your Jungean personality

Meditation Apps

- Not strictly necessary but may be helpful at first
- These seem to be based in authentic practice
  - Waking Up with Sam Harris
    - Totally secular but based on deep Tibetan meditation
  - Headspace by Andy Puddicombe
    - Popular for beginners, too cute for my taste
  - Ten Percent Happier by Dan Harris
    - Lots of authentic meditation teachers
  - Insight Timer App
    - Large free library of guided meditations
    - Good for experienced meditators

Introduction

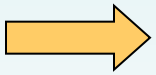
- We start with the fundamentals of computer processing
- Chapter 1 focuses on:
  - components of a computer
  - how computers store and manipulate information
  - computer networks
  - the Internet and the World Wide Web
  - programming and programming languages
  - an introduction to Java
  - an overview of object-oriented concepts

Outline

Computer Processing

Hardware Components

Networks



The Java Programming Language

Program Development

Object-Oriented Programming

Java

- The Java programming language was created by Sun Microsystems, Inc.
- It was introduced in 1995 and it's popularity has grown quickly since
- A *programming language* specifies the words and symbols that we can use to write a program
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*

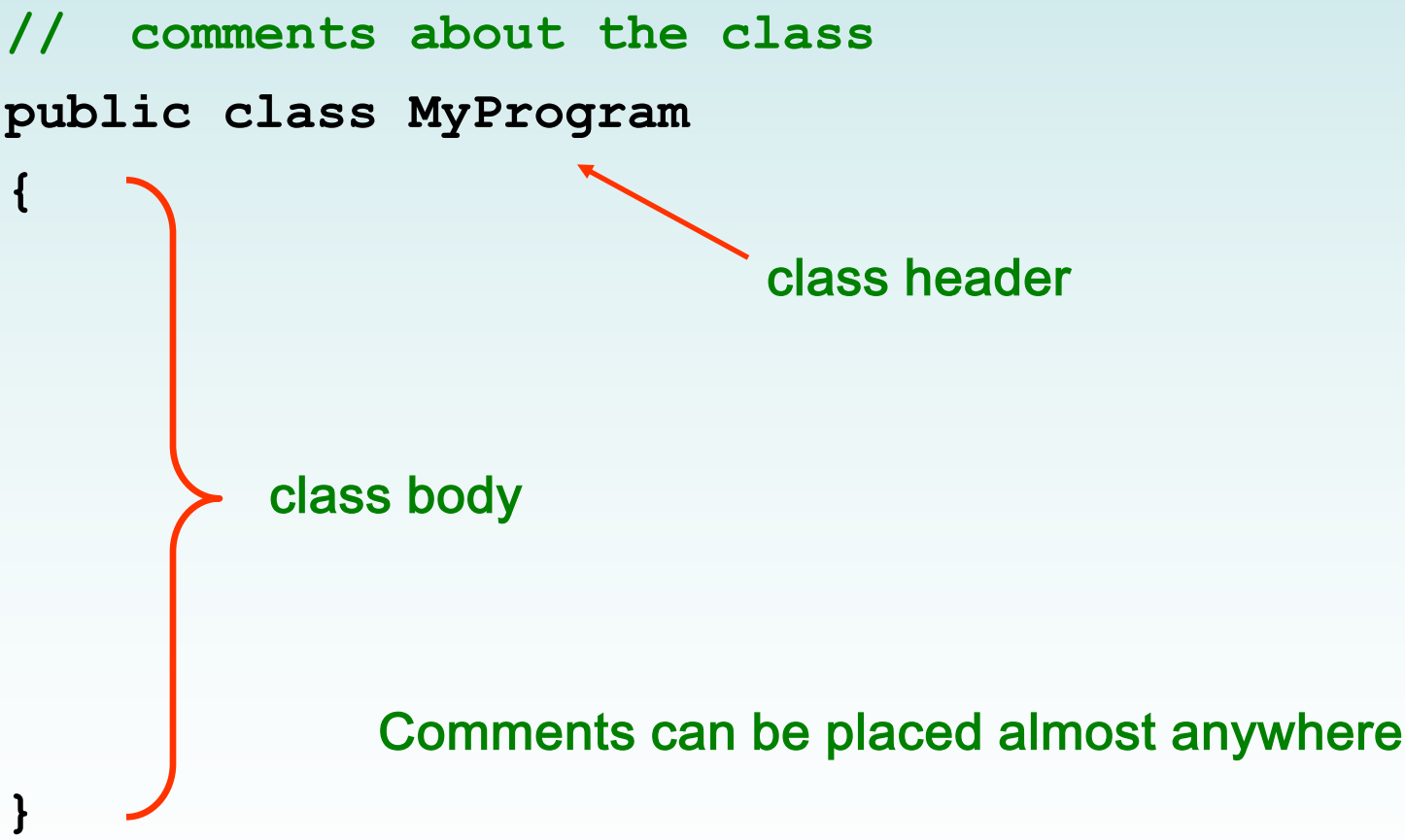
Java Versions

- Versions are important – pay attention
- Java is very active with a new version coming every 6 months with new features
  - The new one replaces the old one which becomes unsupported
- Periodically, a version is chosen for long term support (LTS)
  - We are using Java 21, the latest LTS version
  - Do not install other versions for now

Java Program Structure

- In the Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements*
- These terms will be explored in detail throughout the course
- A Java application always contains a method called `main`
- See `Lincoln.java`

Java Program Structure



Java Program Structure

```
//  comments about the class
public class MyProgram
{
   //  comments about the method
   public static void main (String[] args)
   {

   }

}
```

method body

method header

Comments

- Comments should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Java comments can take three forms:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks        */

/** this is a javadoc comment   */
```

Identifiers

- *Identifiers* are the "words" in a program
- A Java identifier can be made up of letters, digits, the underscore character (_), and the dollar sign
- Identifiers cannot begin with a digit
- Java is *case sensitive*: `Total`, `total`, and `TOTAL` are different identifiers
- By convention, programmers use different case styles for different types of identifiers, such as
  - *title case* for class names - `Lincoln`
  - *upper case* for constants - `MAXIMUM`

Identifiers

- Sometimes the programmer chooses the identifier (such as `Lincoln`)
- Sometimes we are using another programmer's code, so we use the identifiers that they chose (such as `println`)
- Some things that look like identifiers are called *reserved words*
  - These are used for the structure of the program
- Reserved words cannot be used as identifiers

Reserved Keywords

- The Java reserved words:

`abstract`	`else`	`interface`	`switch`
`assert`	`enum`	`long`	`synchronized`
`boolean`	`extends`	`native`	`this`
`break`	`false`	`new`	`throw`
`byte`	`final`	`null`	`throws`
`case`	`finally`	`package`	`transient`
`catch`	`float`	`private`	`true`
`char`	`for`	`protected`	`try`
`class`	`goto`	`public`	`void`
`const`	`if`	`return`	`volatile`
`continue`	`implements`	`short`	`while`
`default`	`import`	`static`	`_ (underscore)`
`do`	`instanceof`	`strictfp`	
`double`	`int`	`super`	

Contextual Keywords

- exports
- module
- non-sealed
- open
- opens
- permits
- provides
- record
- required
- sealed
- to
- transitive
- uses
- var
- when
- with
- yield

Quick Check

Which of the following are valid Java identifiers?

```
grade
quizGrade
NetworkConnection
frame2
3rdTestScore
MAXIMUM
MIN_CAPACITY
student#
Shelves1&2
```

Quick Check

Which of the following are valid Java identifiers?

`grade`	Valid
`quizGrade`	Valid
`NetworkConnection`	Valid
`frame2`	Valid
`3rdTestScore`	Invalid – cannot begin with a digit
`MAXIMUM`	Valid
`MIN_CAPACITY`	Valid
`student#`	Invalid – cannot contain the '#' character
`Shelves1&2`	Invalid – cannot contain the '&' character

White Space

- Spaces, blank lines, and tabs are called *white space*
- White space is used to separate words and symbols in a program
- Extra white space is ignored
- A valid Java program can be formatted many ways
- Programs should be formatted to enhance readability, using consistent indentation
- See `Lincoln2.java` and `Lincoln3.java`

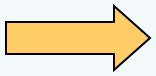
Outline

Computer Processing

Hardware Components

Networks

The Java Programming Language



Program Development

Object-Oriented Programming

Program Development

- The mechanics of developing a program include several activities:
  - writing the program in a specific programming language (such as Java)
  - translating the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process

Language Levels

- There are four programming language levels:
  - machine language
  - assembly language
  - high-level language
  - fourth-generation language
- Each type of CPU has its own specific *machine language*
- The other levels were created to make it easier for a human being to read and write programs

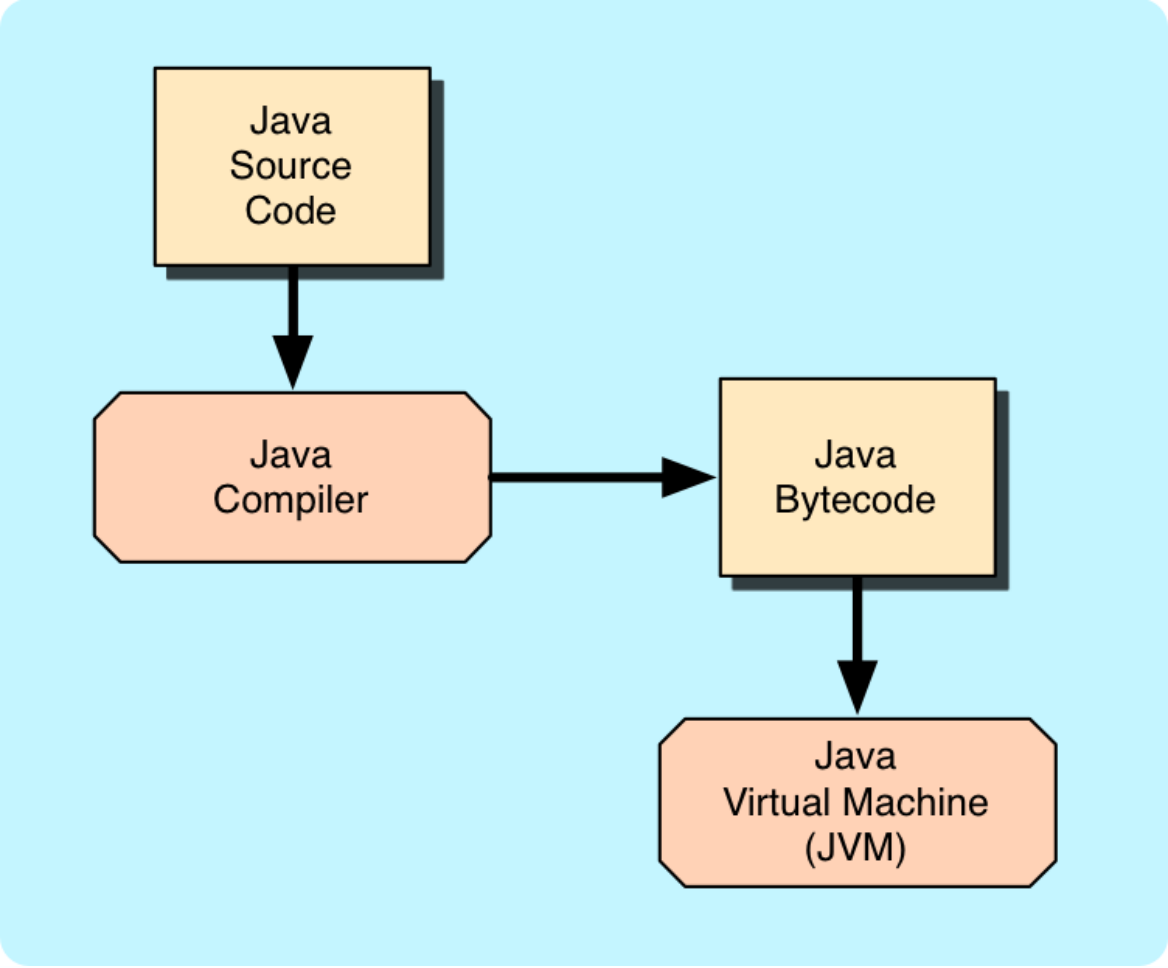
Programming Languages

- Each type of CPU executes only a particular *machine language*
- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates *source code* into a specific target language
- Sometimes, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

Java Translation

- The Java compiler translates Java source code into a special representation called *bytecode*
- Java bytecode is not the machine language for any traditional CPU
- Bytecode is executed by the *Java Virtual Machine* (JVM)
- Therefore Java bytecode is not tied to any particular machine
- Java is considered to be *architecture-neutral*

Java Translation



Development Environments

- There are many programs that support the development of Java software, including:
  - Java Development Kit (JDK)
  - Eclipse
  - NetBeans
  - BlueJ
  - jGRASP
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

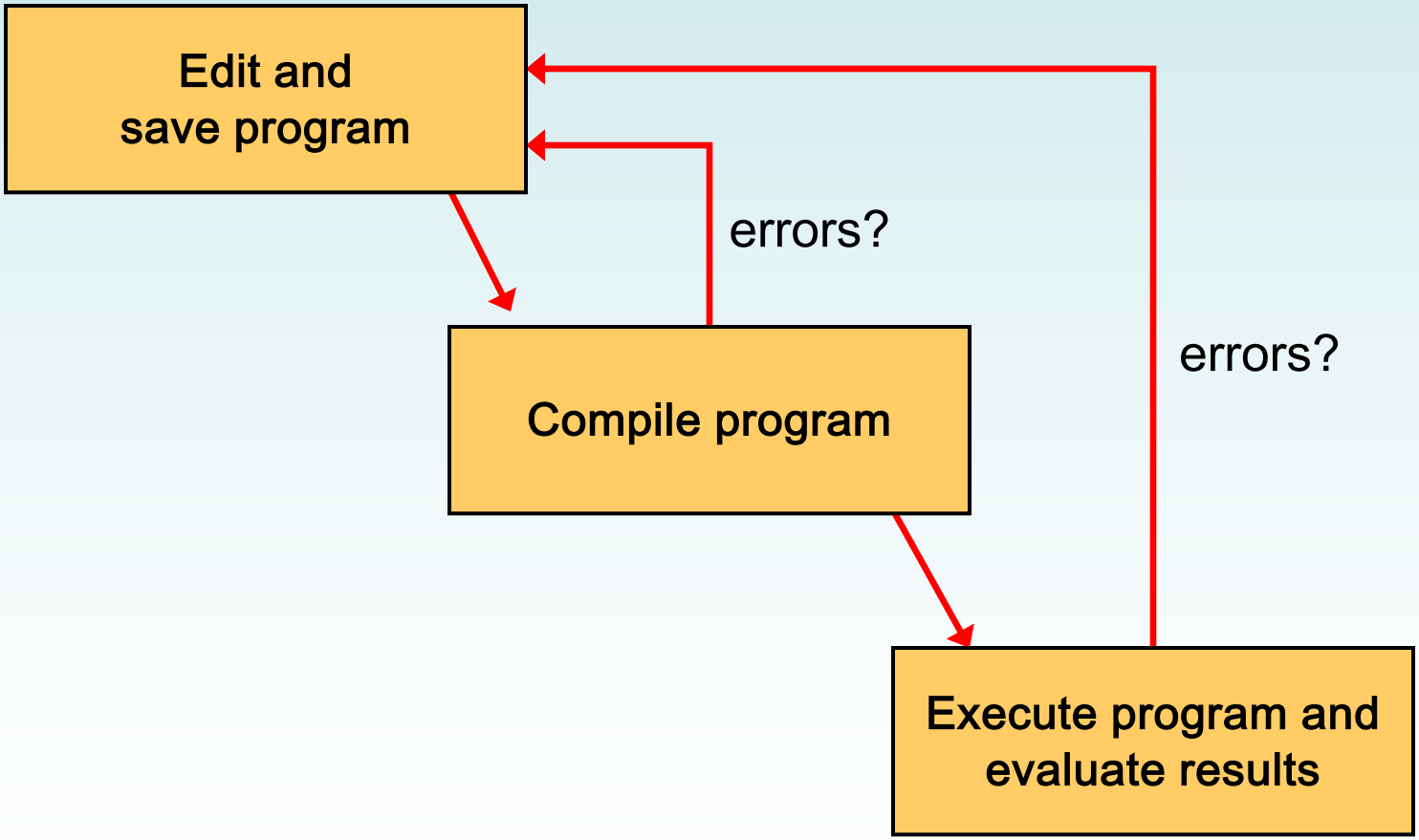
Syntax and Semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do

Errors

- A program can have three types of errors
- The compiler will find syntax errors and other basic problems (*compile-time errors*)
  - If compile-time errors exist, an executable version of the program is not created
- A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
- A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)

Basic Program Development



Outline

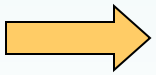
Computer Processing

Hardware Components

Networks

The Java Programming Language

Program Development



Object-Oriented Programming

Problem Solving

- The purpose of writing a program is to solve a problem
- Solving a problem consists of multiple activities:
  - Understand the problem
  - Design a solution
  - Consider alternatives and refine the solution
  - Implement the solution
  - Test the solution
- These activities are not purely linear – they overlap and interact

Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An *object-oriented approach* lends itself to this kind of solution decomposition
- We will dissect our solutions into pieces called objects and classes

Object-Oriented Programming

- Java is an object-oriented programming language
- As the term implies, an object is a fundamental entity in a Java program
- Objects can be used effectively to represent real-world entities
- For instance, an object might represent a particular employee in a company
- Each employee object handles the processing and data management related to that employee

Objects

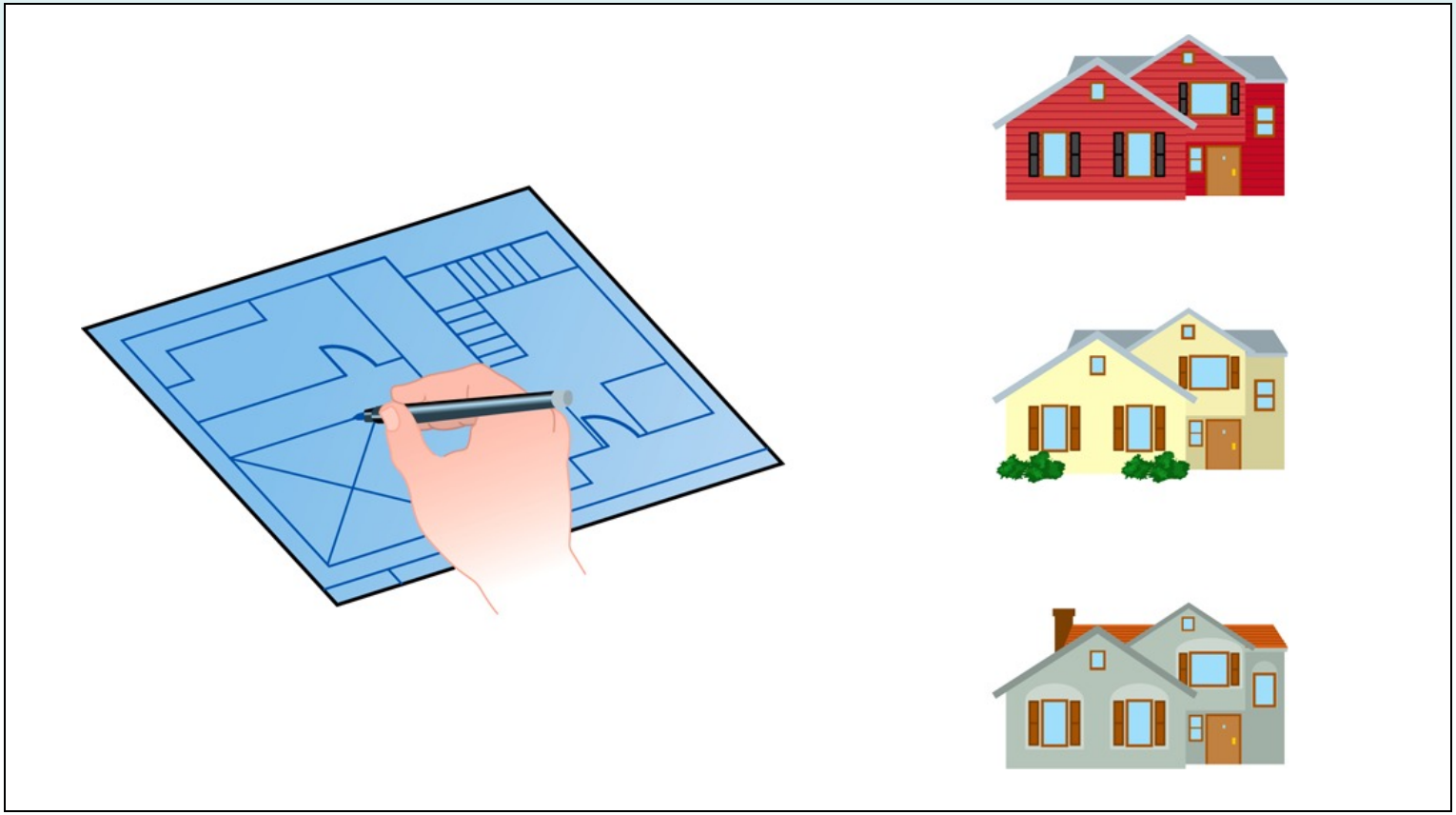
- An object has:
  - *state* - descriptive characteristics
  - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

Classes

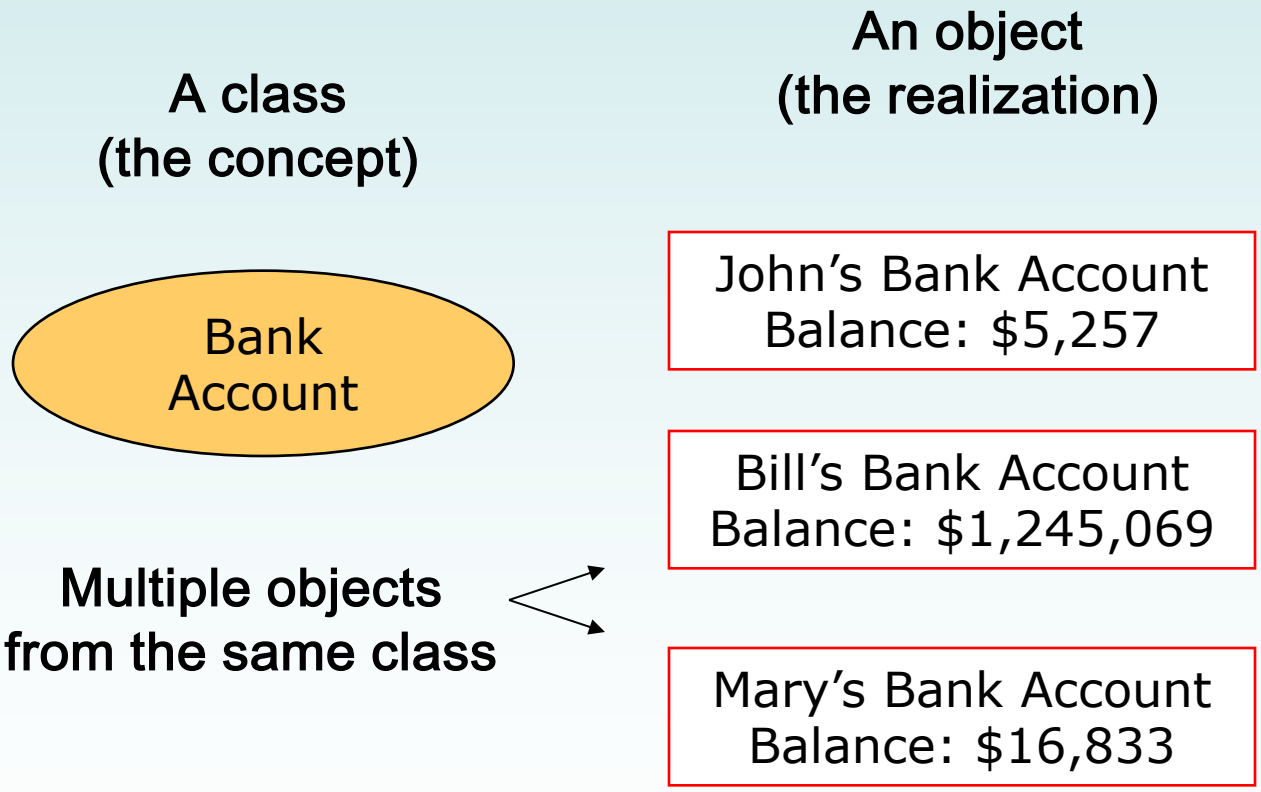
- An object is defined by *a class*
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment of that concept
- Multiple objects can be created from the same class

Class = Blueprint

- One blueprint to create several similar, but different, houses:

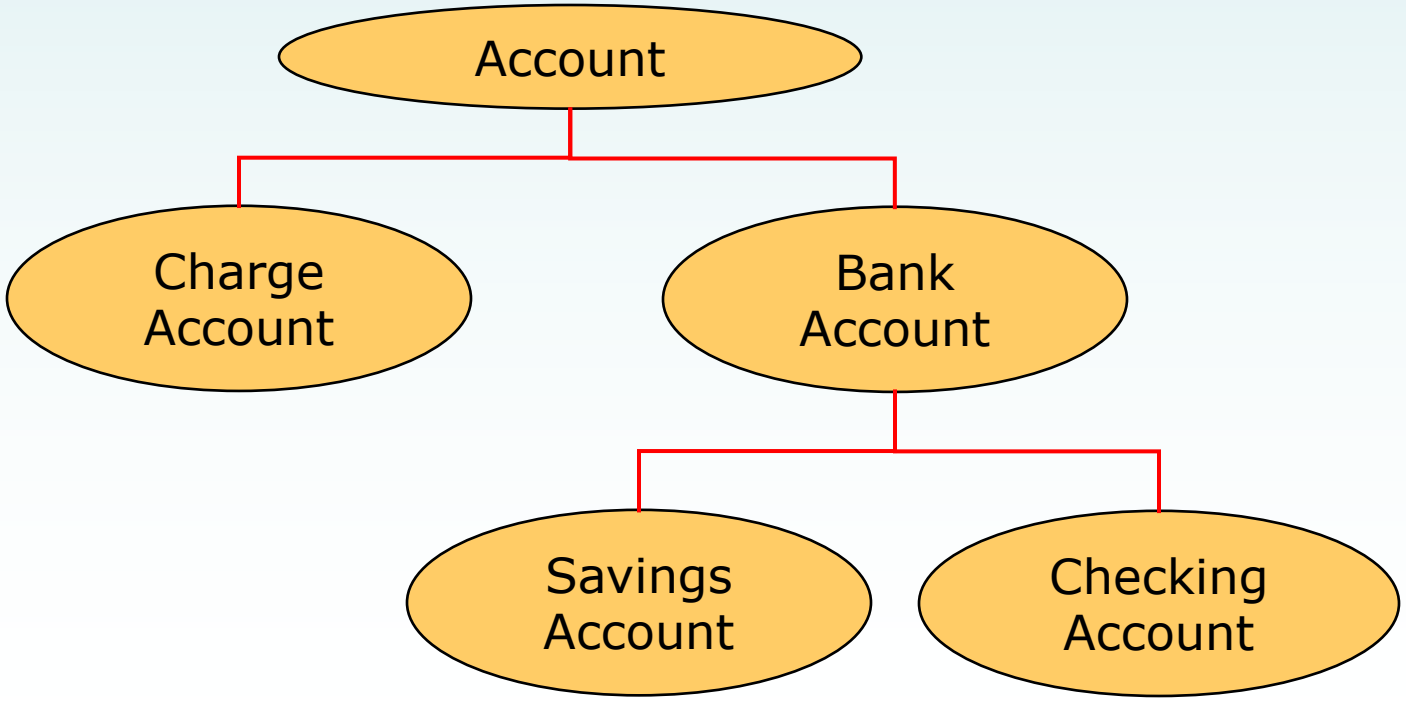


Objects and Classes



Inheritance

- One class can be used to derive another via *inheritance*
- Classes can be organized into hierarchies



Summary

- Chapter 1 focused on:
  - components of a computer
  - how those components interact
  - how computers store and manipulate information
  - computer networks
  - the Internet and the World Wide Web
  - programming and programming languages
  - an introduction to Java
  - an overview of object-oriented concepts

- What is a program?

  A series of instructions that the hardware executes one after another.

- What is software?

  Software consists of a program paired with the data the program uses. It is the intangible counterpart to physical hardware components.

- What are the key hardware components of a computer?

  - CPU: Executes individual program commands
  - I/O Devices: Allow human interaction (mouse, keyboard, etc.)
  - Main Memory (RAM): Temporary storage for software and data being processed
  - Secondary Memory (Drives): Permanent storage devices
- What are the main software categories?

  - Operating System: Core software that manages resources and provides user interface
  - Applications: All other software that provides specific functionality to users
- What is the difference between analog and digital?

  - Analog: Continuous, in direct proportion to source of information
  - Digital: Breaks information into discrete pieces and represents them as numbers
- What is binary and why do computers use it?

  Binary is a base-2 numeric system using only 0 and 1. Computers use binary due to hardware limitations - components can only reliably distinguish between two states (on/off, magnetic polarization, etc.).

- What is computer architecture?

  How the hardware components of a computer are organized and connected. Information travels between components across a bus (group of wires).

- What is the difference between volatile and non-volatile memory?

  - Volatile memory (like RAM) loses data when power is lost
  - Non-volatile memory (like hard drives) retains data even without power
- What are the three main components of a CPU?

  - Control Unit: Coordinates processing steps and data transfer
  - Registers: Small but extremely fast memory cache on the CPU
  - Arithmetic/Logic Unit: Performs calculations and makes decisions

- What is the CPU fetch-decode-execute cycle?

    - Fetch: Get instruction from memory
    - Decode: Determine what operation to perform
    - Execute: Carry out the instruction
- What is a network and what are its types?

    A network connects two or more computers to exchange information. Types:

    - LAN (Local Area Network): Short distance, small number of computers
    - WAN (Wide Area Network): Connects multiple LANs over large distances
- What is the Internet vs. the World Wide Web?

    - Internet: Global network of connected computers using TCP/IP protocol
    - World Wide Web: System of linked documents accessed via browsers that makes using the internet easy for humans
- What is object-oriented programming?

    A programming paradigm where:

    - All code is contained within classes
    - Classes are blueprints for objects
    - Objects have state (attributes) and behavior (methods)
    - Objects should be encapsulated (protect and manage their own information)