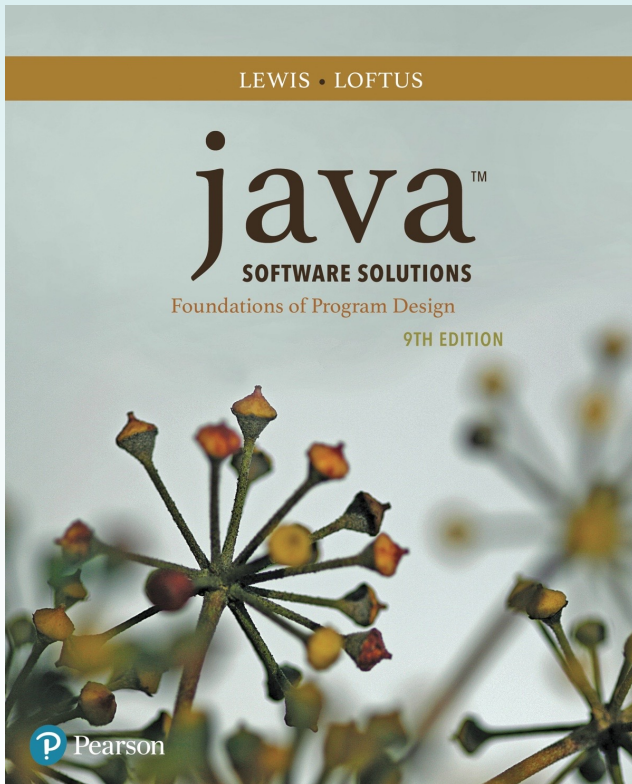


Chapter 1

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 Jungian 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 its 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

```
// comments about the class
```

```
public class MyProgram
```

```
{
```

class header



class body

Comments can be placed almost anywhere

```
}
```

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 Words

- The Java reserved words:

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

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?

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

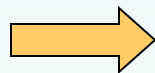
Outline

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

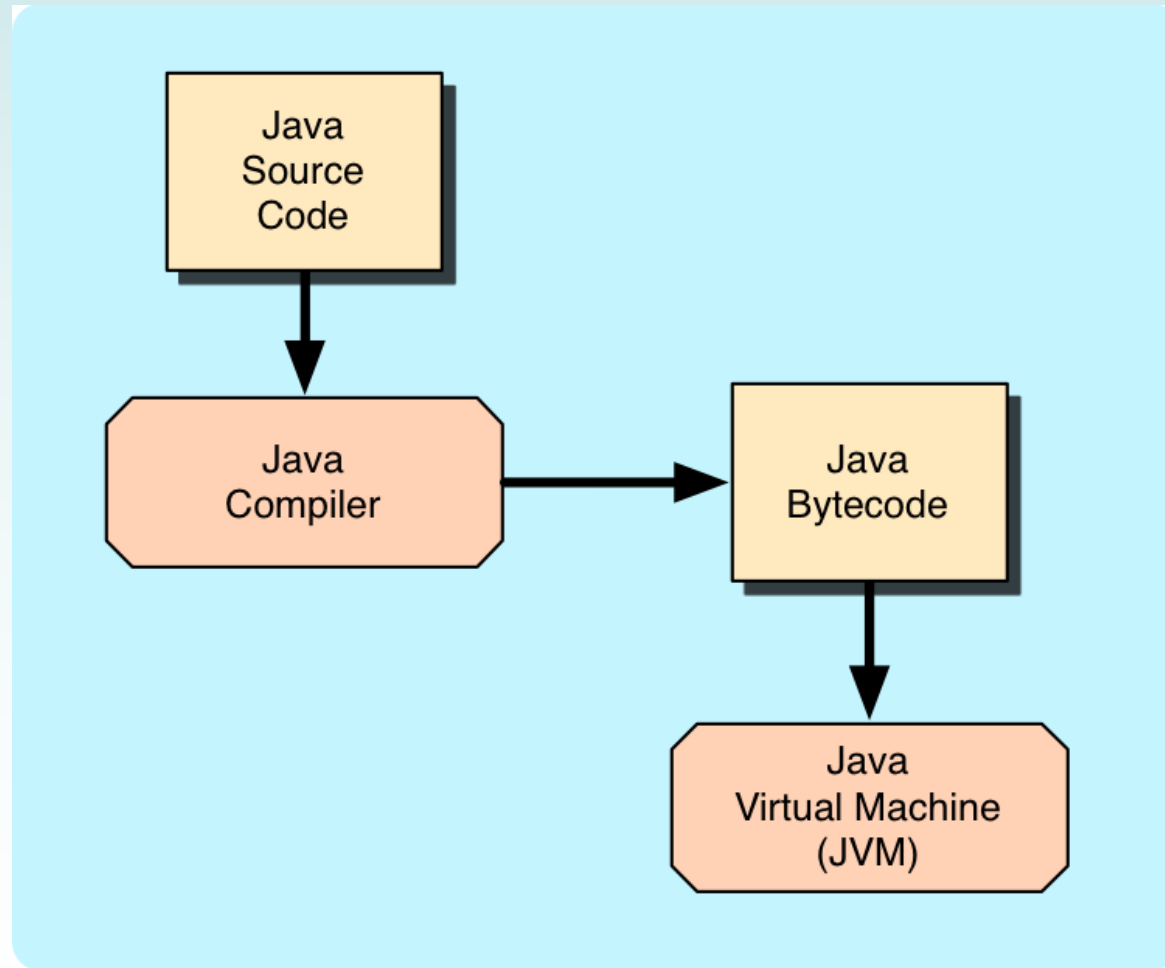
Programming Languages

- 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

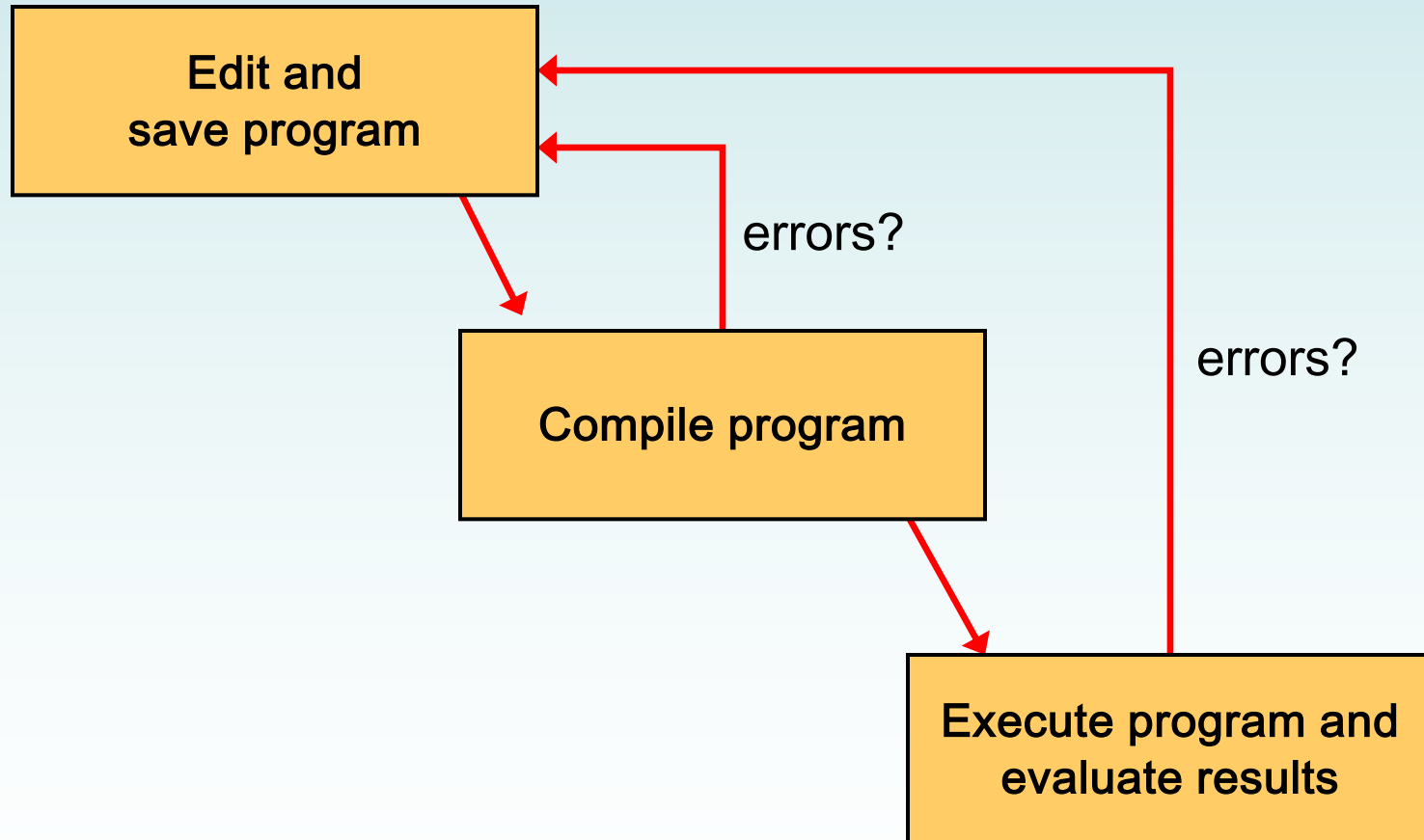
Syntax and Semantics

- 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

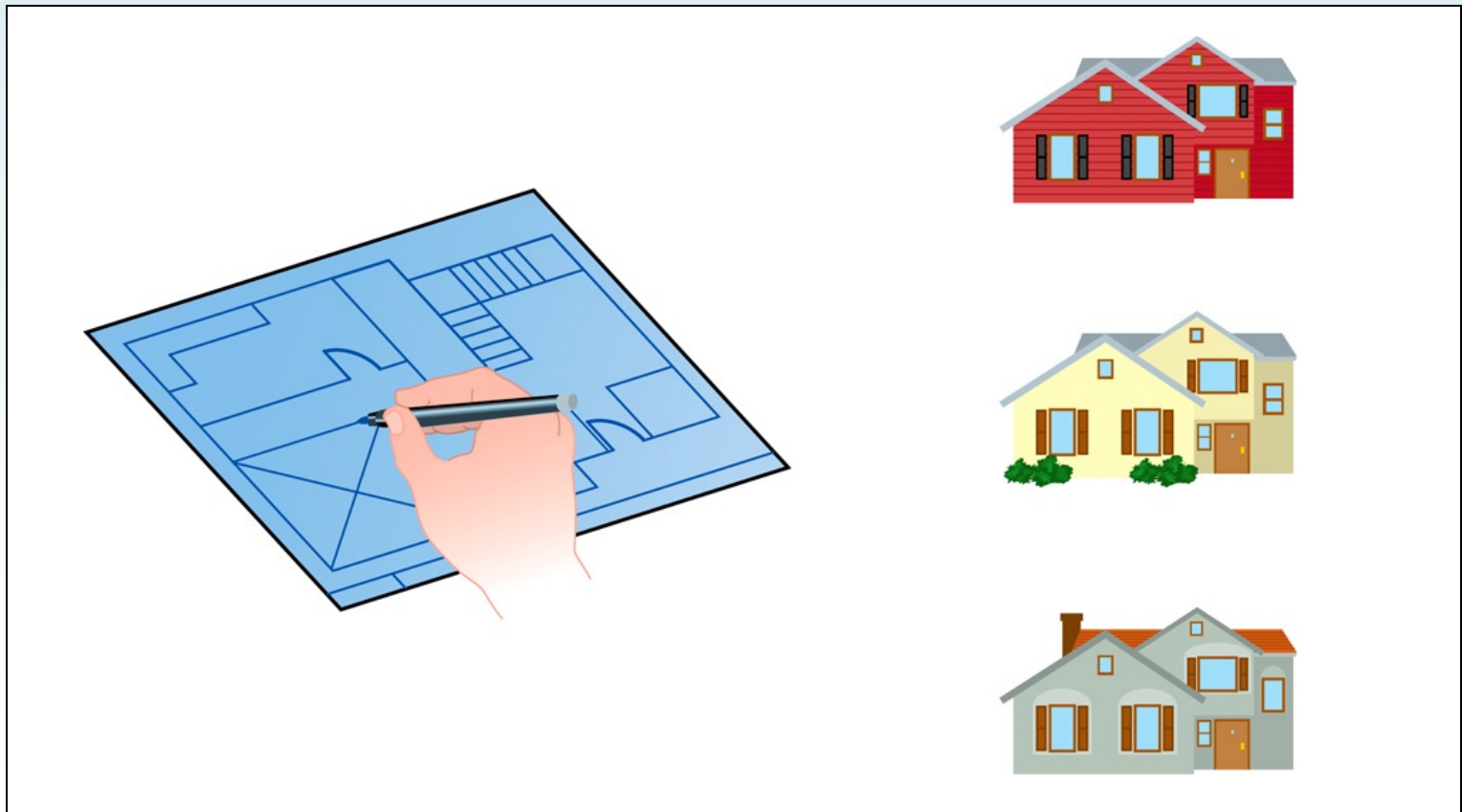
- An object has:
 - *state* - descriptive characteristics
 - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
-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

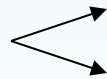


Objects and Classes

**A class
(the concept)**



**Multiple objects
from the same class**



**An object
(the realization)**

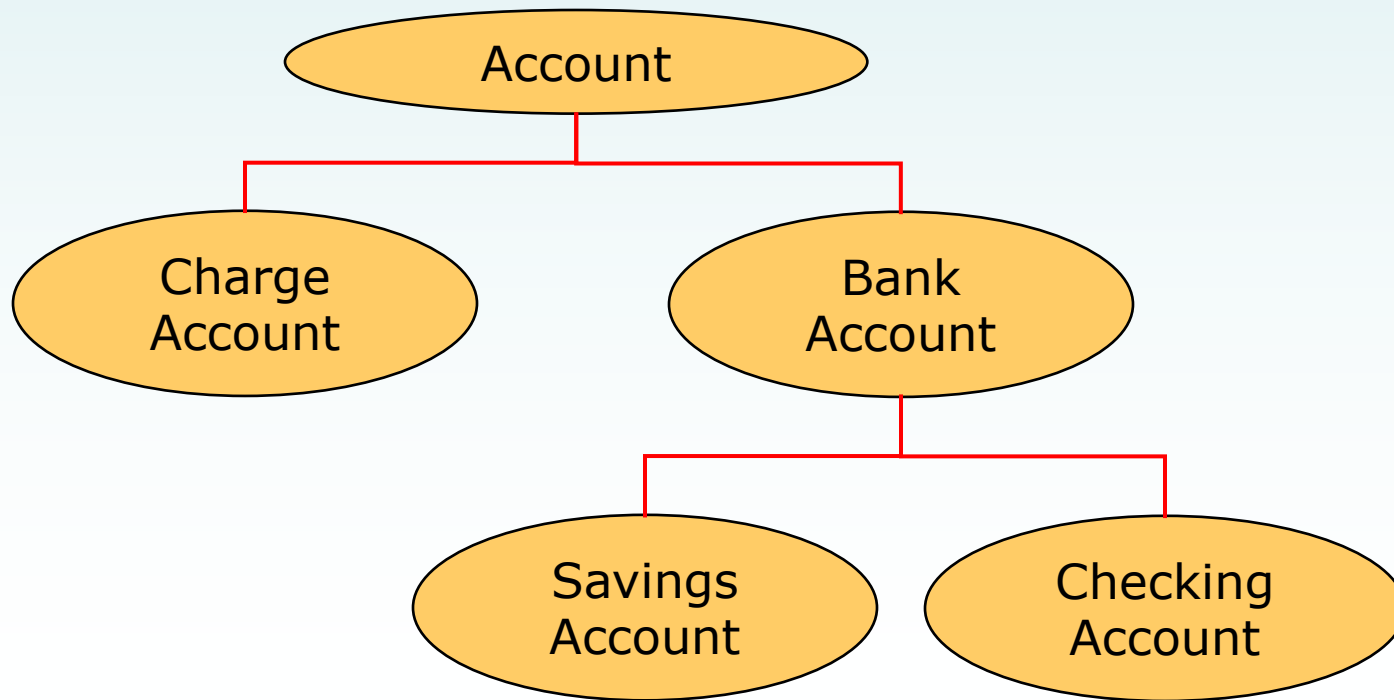
John's Bank Account
Balance: \$5,257

Bill's Bank Account
Balance: \$1,245,069

Mary's Bank Account
Balance: \$16,833

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