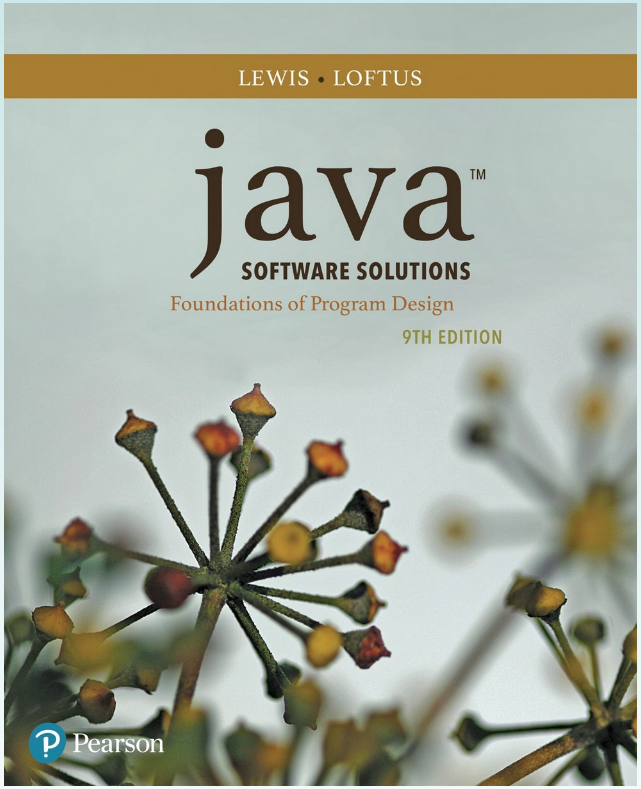
Chapter 1
Introduction



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
  - the Java programming language

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

The Java Programming Language

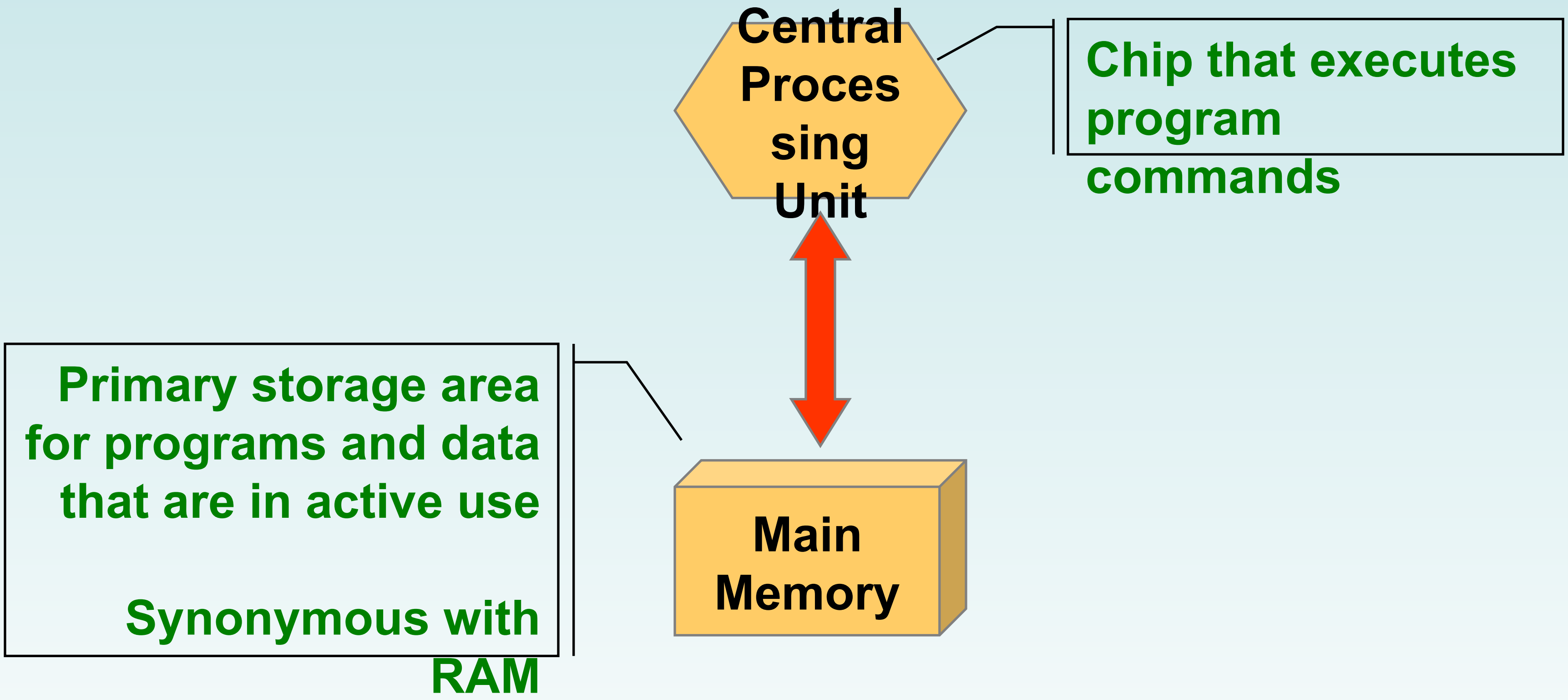
Program Development

Object-Oriented Programming

Hardware and Software

- Hardware
  - the physical, tangible parts of a computer
  - keyboard, monitor, disks, wires, chips, etc.
- Software
  - programs and data
  - a *program* is a series of instructions
- A computer requires both hardware and software
- Each is essentially useless without the other

CPU and Main Memory



Central Processing Unit

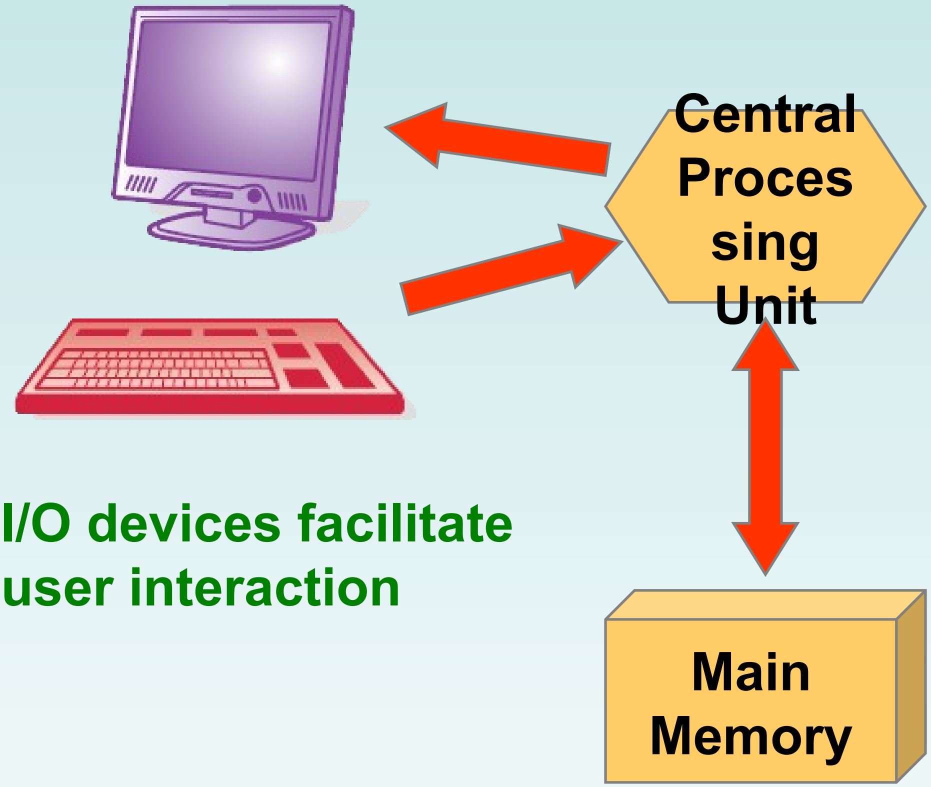
Chip that executes program commands

Main Memory

Primary storage area for programs and data that are in active use

Synonymous with RAM

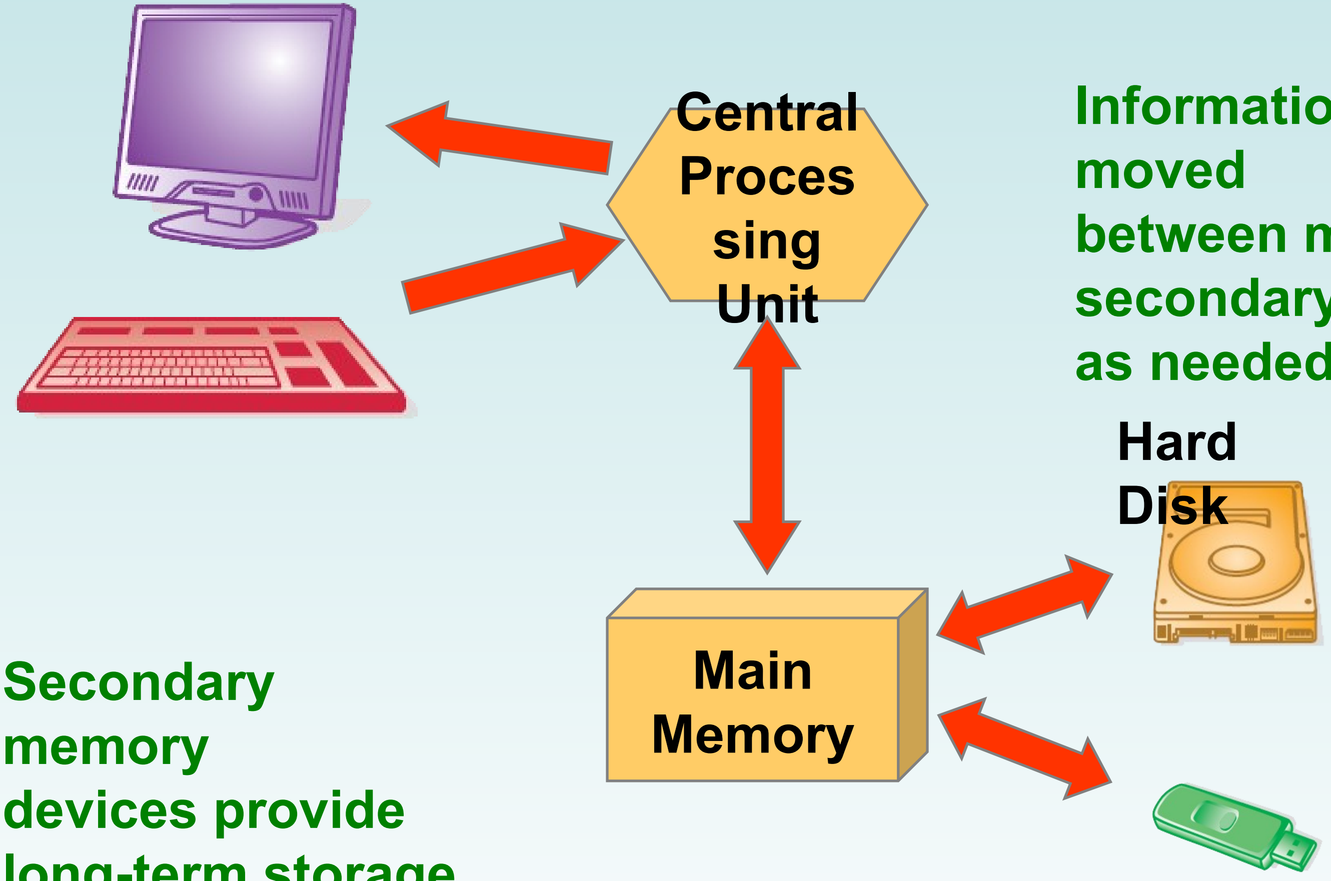
Input / Output Devices



I/O devices facilitate user interaction

Monitor
screen
Keyboard
Mouse
Touch screen

Secondary Memory Devices



Central Processing Unit

Main Memory

Hard Disk

USB Flash Drive

Information is moved between main and secondary memory as needed

Secondary memory devices provide long-term storage

Software Categories

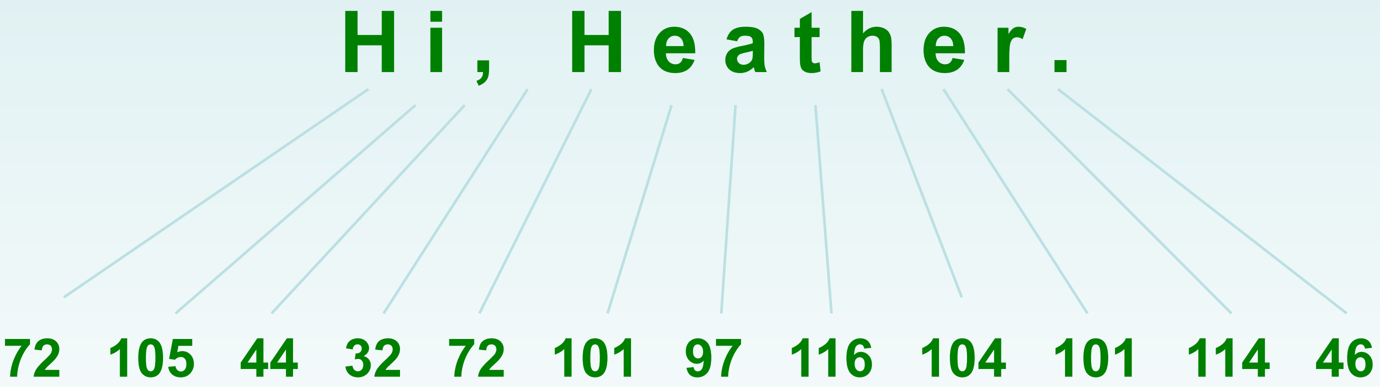
- Operating System
  - controls all machine activities
  - provides the user interface to the computer
  - manages resources such as the CPU and memory
  - Windows, Mac OS, Unix, Linux,
- Application program
  - generic term for any other kind of software
  - word processors, missile control systems, games
- Most operating systems and application programs have a *graphical user interface* (GUI)

Digital Information

- Computers store all information digitally:
  - numbers
  - text
  - graphics and images
  - audio
  - video
  - program instructions
- In some way, all information is *digitized* - broken down into pieces and represented as numbers

Representing Text Digitally

- For example, every character is stored as a number, including spaces, digits, and punctuation
- Corresponding upper and lower case letters are separate characters



Hi, Heather.

72 105 44 32 72 101 97 116 104 101 114 46

Binary Numbers

- Once information has been digitized, it is represented and stored in memory using the *binary number system*
- A single binary digit (0 or 1) is called a *bit*
- Devices that store and move information are cheaper and more reliable if they have to represent only two states
- A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)
- Permutations of bits are used to store values

Bit Permutations

1 bit	2 bits	3 bits	4 bits	
0	00	000	0000	1000
1	01	001	0001	1001
	10	010	0010	1010
	11	011	0011	1011
		100	0100	1100
		101	0101	1101
		110	0110	1110
		111	0111	1111

Each additional bit doubles the number of possible permutations

Bit Permutations

- Each permutation can represent a particular item
- There are 2^N permutations of N bits
- Therefore, N bits are needed to represent 2^N unique items

How many items can be represented by

1 bit ?	**$2^1 = 2$ items**
2 bits ?	**$2^2 = 4$ items**
3 bits ?	**$2^3 = 8$ items**
4 bits ?	**$2^4 = 16$ items**
5 bits ?	**$2^5 = 32$ items**

Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

Five bits wouldn't be enough, because 2^5 is 32.

Six bits would give us 64 permutations, and some wouldn't be used.

000000	Alabama
000001	Alaska
000010	Arizona
000011	Arkansas
000100	California
000101	Colorado

etc.

Outline

Computer Processing

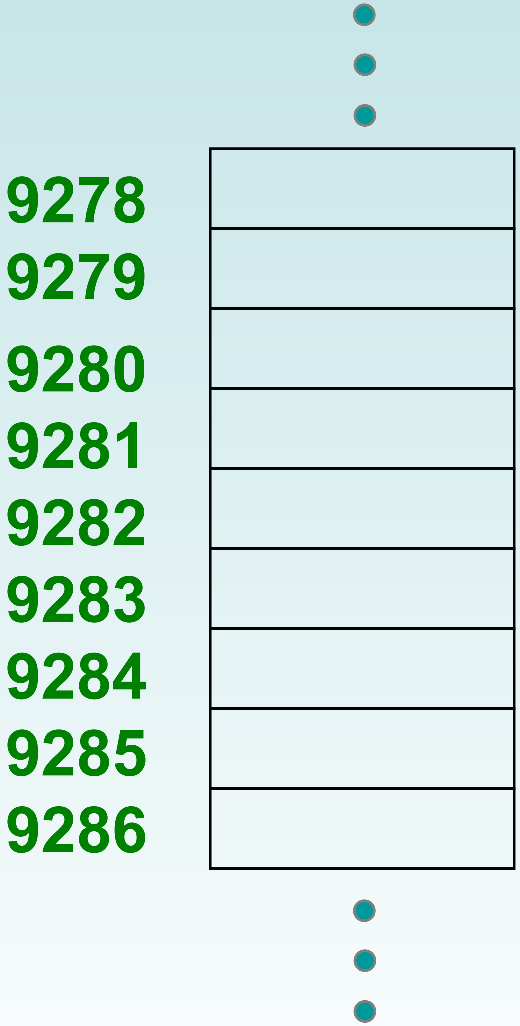
Hardware Components

The Java Programming Language

Program Development

Object-Oriented Programming

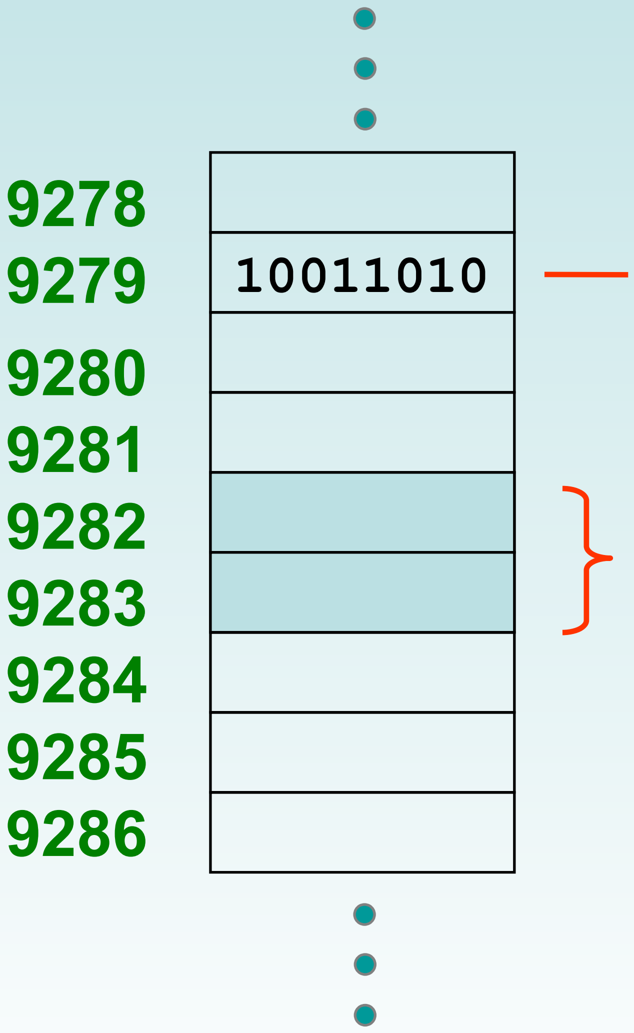
Memory



Main memory is divided into many memory locations (or *cells*)

Each memory cell has a numeric *address*, which uniquely identifies it

Storing Information



9278	
9279	10011010
9280	
9281	
9282	
9283	
9284	
9285	
9286	

Each memory cell stores a set number of bits (usually 8 bits, or one *byte*)

Large values are stored in consecutive memory locations

Storage Capacity

- Every memory device has a *storage capacity*, indicating the number of bytes it can hold
- Capacities are expressed in various units:

Unit	Symbol	Number of Bytes
kilobyte	KB	2^{10} = 1024
megabyte	MB	2^{20} (over one million)
gigabyte	GB	2^{30} (over one billion)
terabyte	TB	2^{40} (over one trillion)
petabyte	PB	2^{50} (a whole bunch)

Outline

Computer Processing

Hardware Components

⇨ The Java Programming Language

Program Development

Object-Oriented Programming

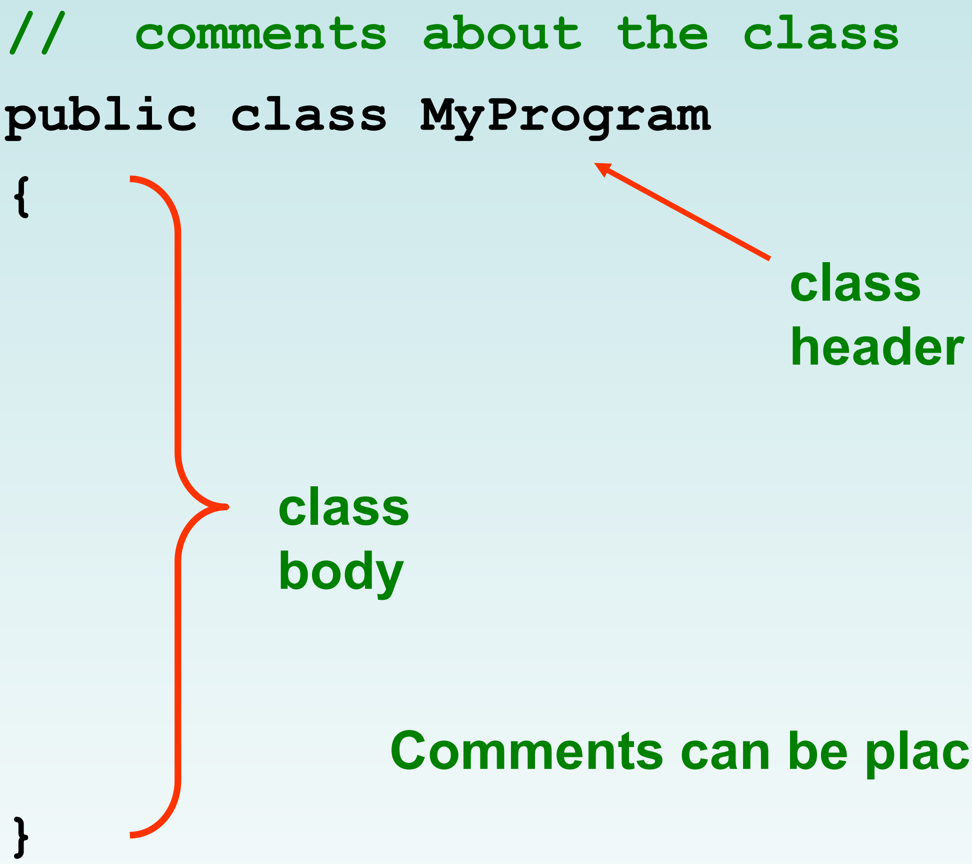
Java

- 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 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

Copyright © 2017 Pearson Education, Inc.

```java
//********************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  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.");
   }
}
```

Output

```
A quote by Abraham Lincoln:
Whatever you are, be a good one.
```

```
//*********                                           ***********
//  Lincol
//
//  Demons
//********************************************************************

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

- 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 identifer(such as `Lincoln`)
- Sometimes we are using another programmer's code, so we use the identifiers that he or she chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
- A reserved word cannot be used in any other way

Reserved Words

- The Java reserved words:

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

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

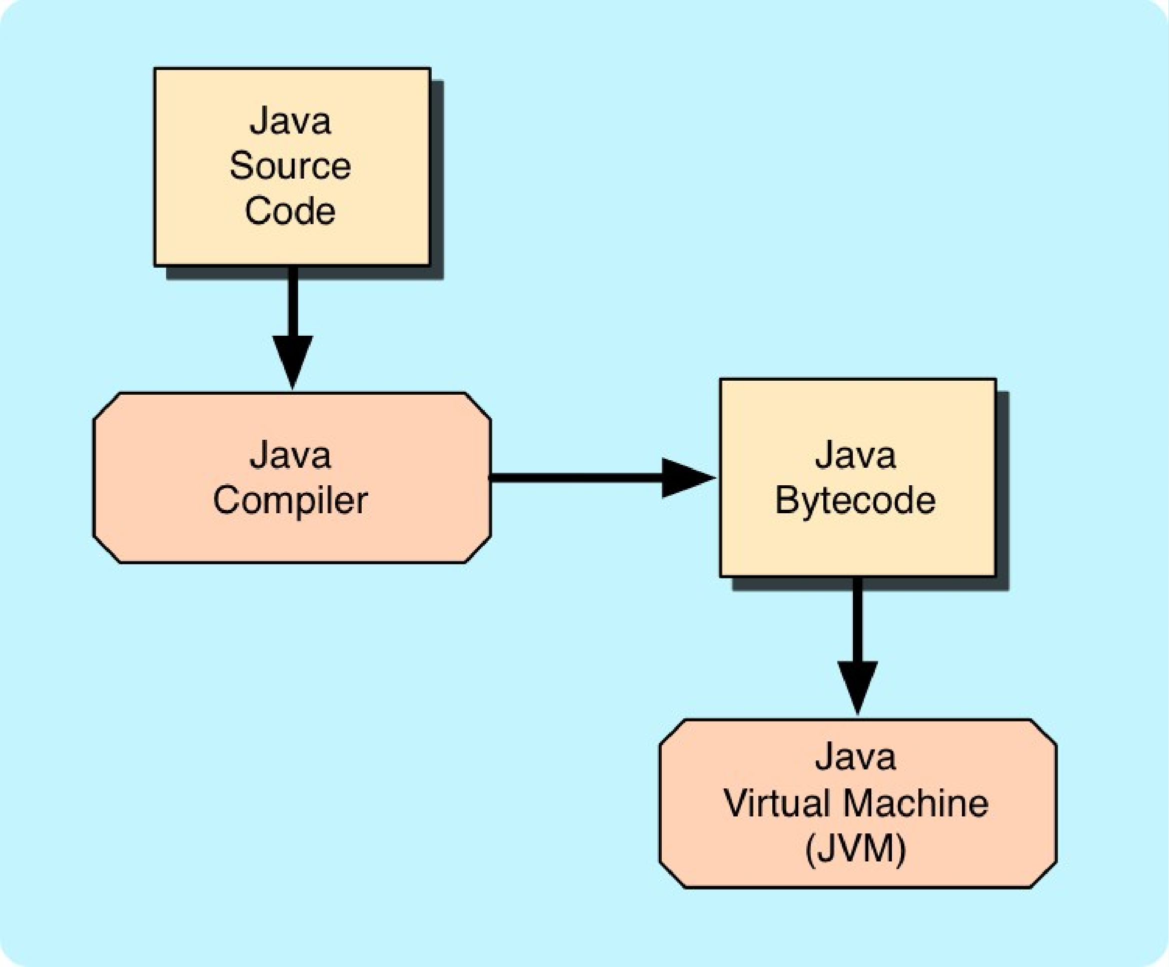
Programming Languages

- 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

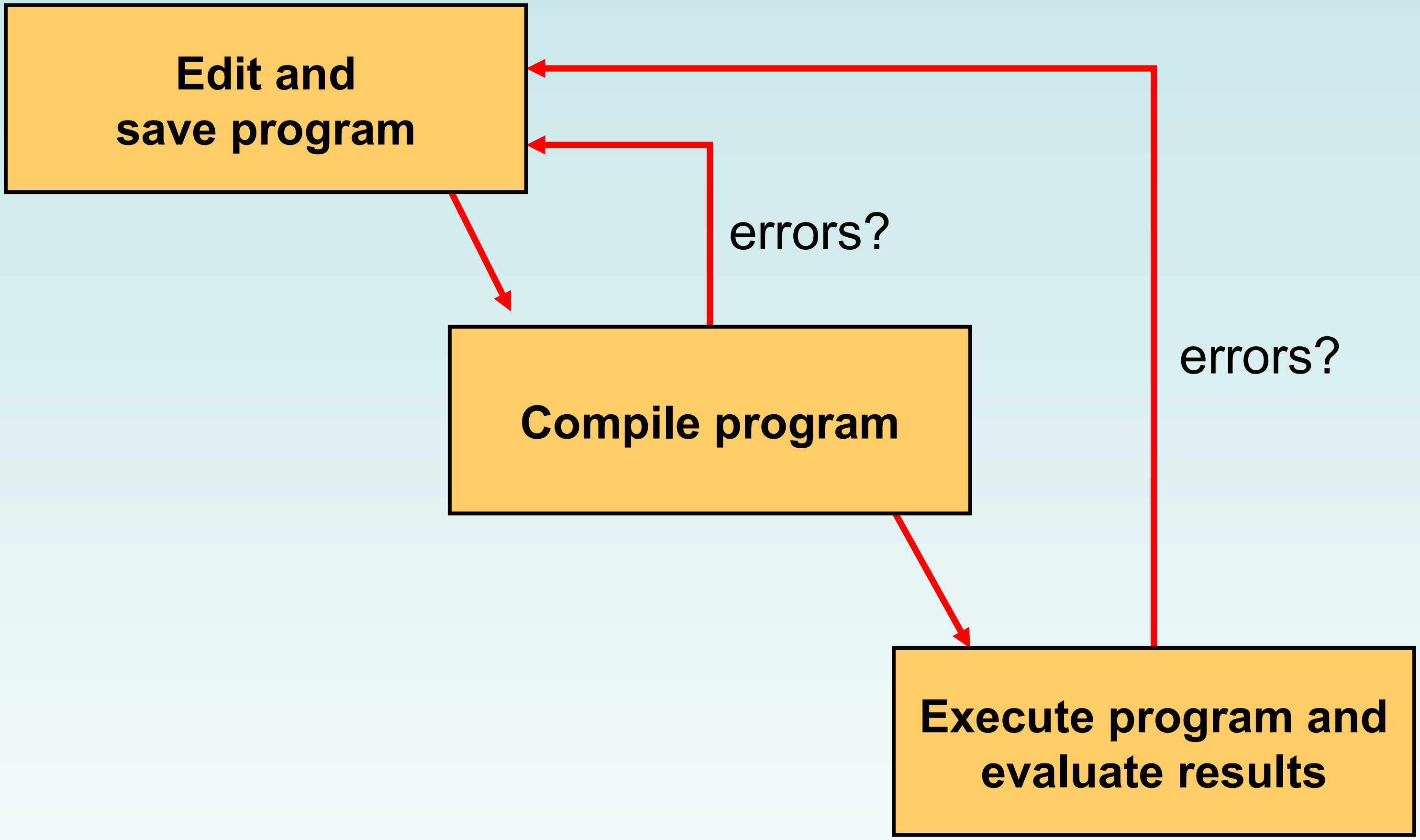
Syntax and Semantics

- 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



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

Outline

Computer Processing

Hardware Components

Networks

The Java Programming Language

Program Development

Object-Oriented Programming

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

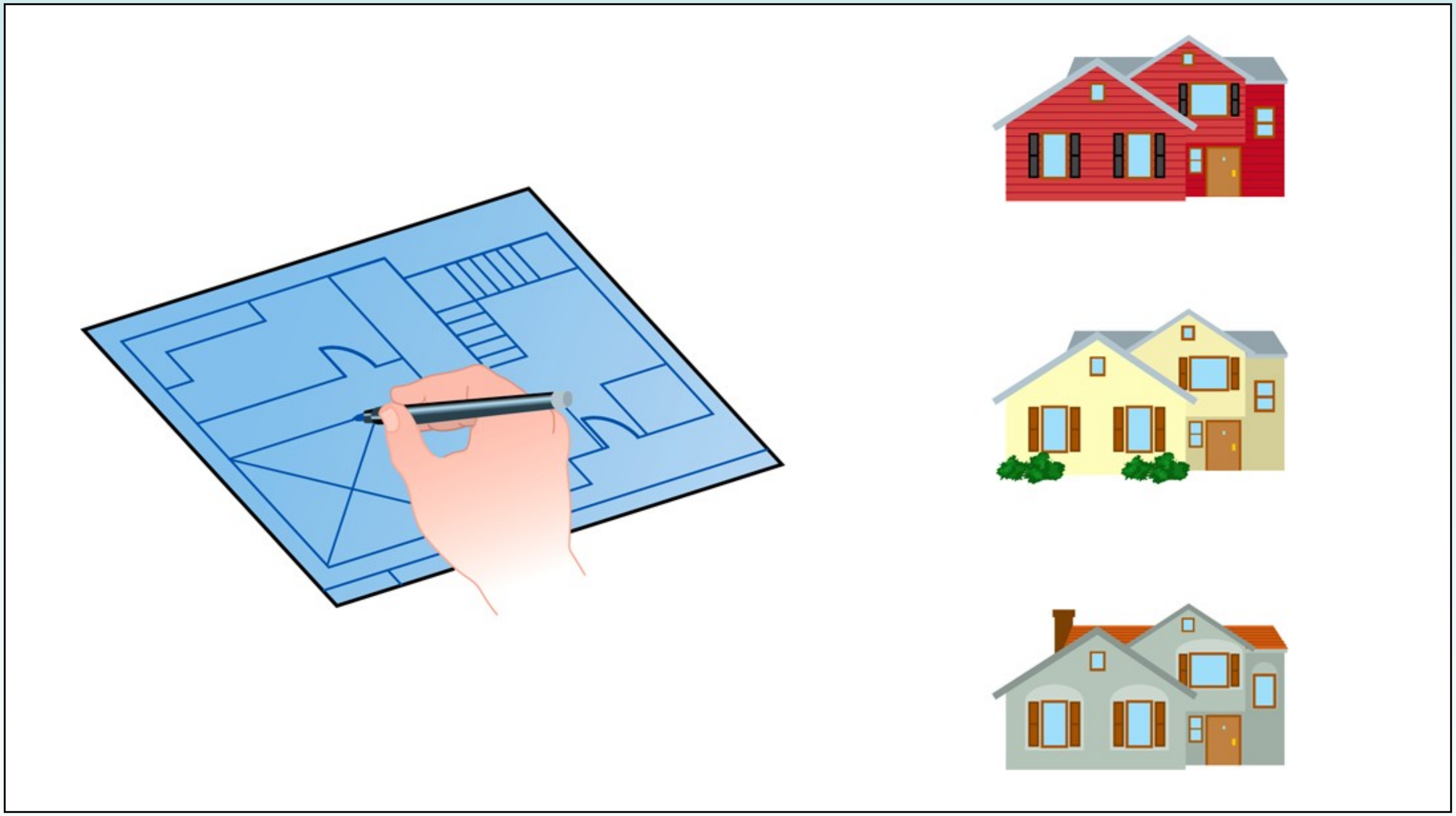
- An object has:
  - *state* - descriptive characteristics
  - *behaviors* - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals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

Classes

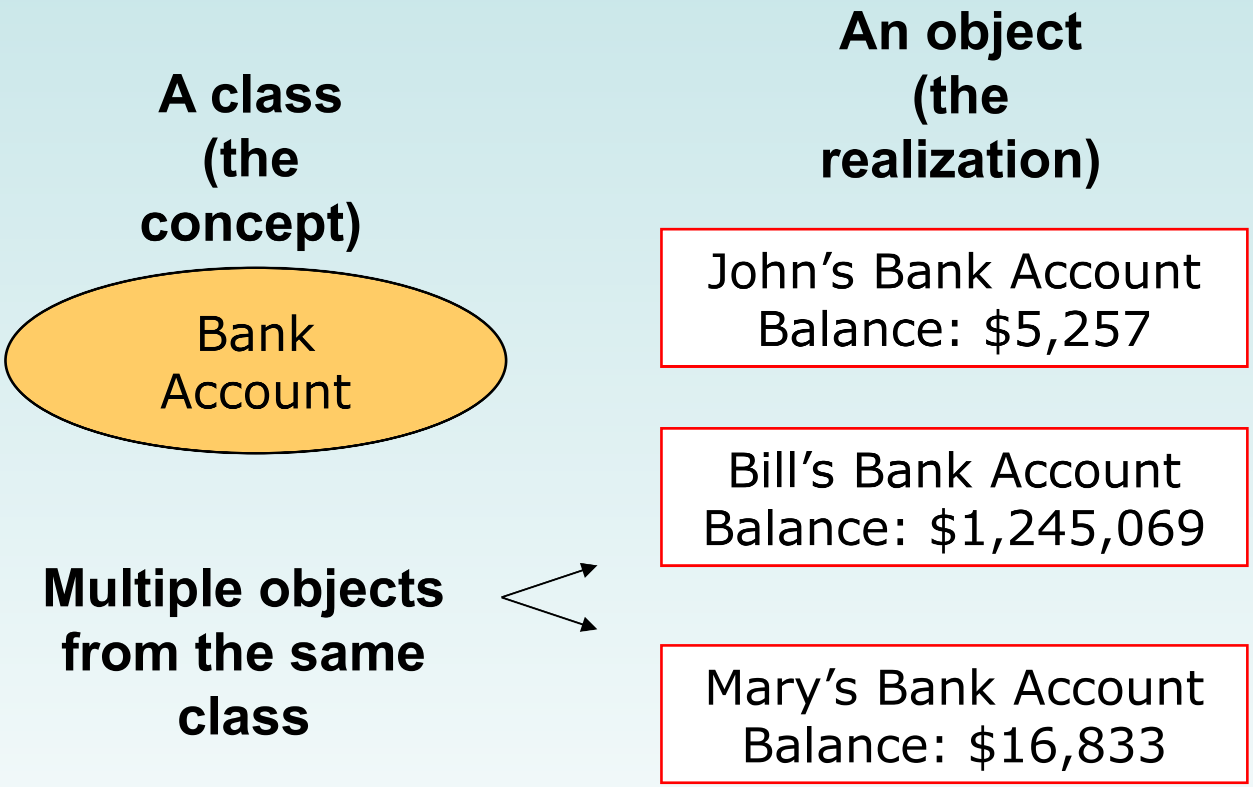
- An object is defined by a *class*
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment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

Bank Account

An object (the realization)

John’s Bank Account
Balance: $5,257

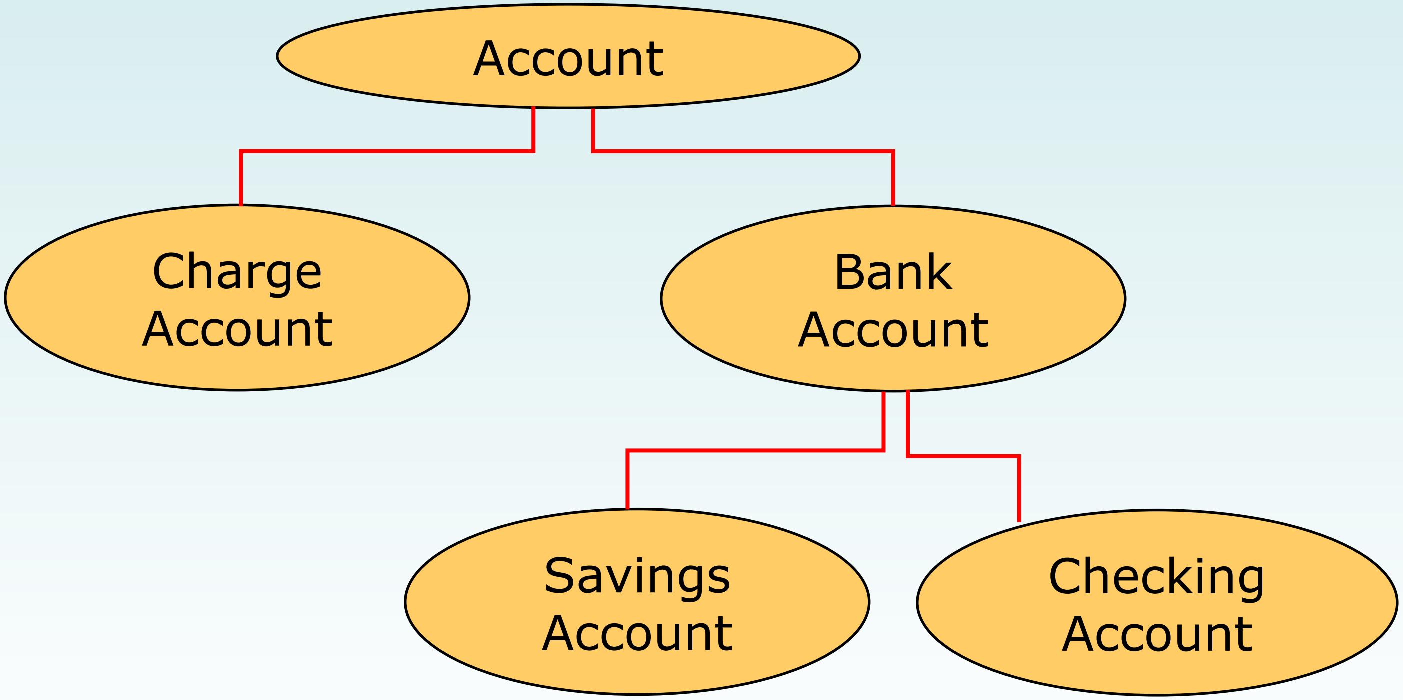
Bill’s Bank Account
Balance: $1,245,069

Mary’s Bank Account
Balance: $16,833

Multiple objects from the same class

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
  - programming and programming languages
  - an introduction to Java
  - an overview of object-oriented concepts