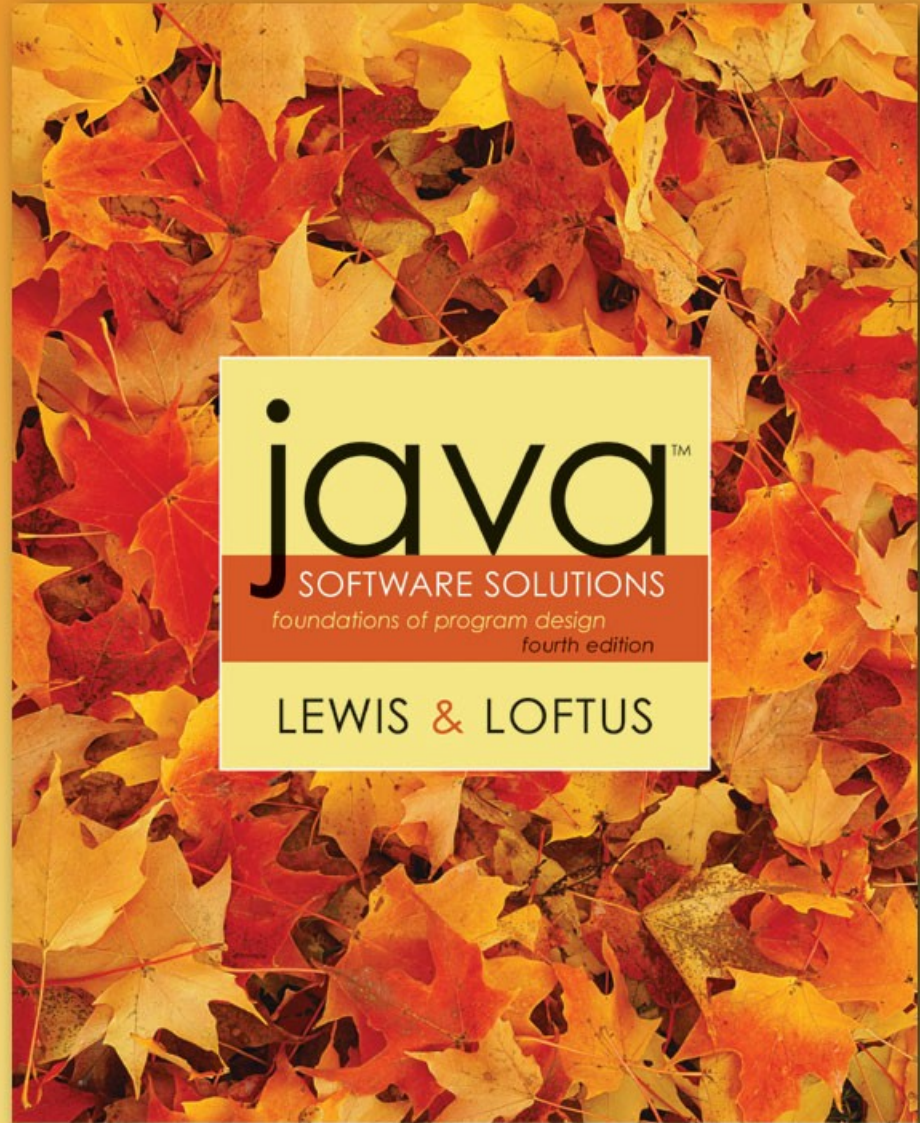


# Chapter 1

## Introduction



# 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**



# Introduction

- **We first need to explore the fundamentals of computer processing**
- **Chapter 1 focuses 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**

# Outline



**Computer Processing**

**Hardware Components**

**Networks**

**The Java Programming Language**

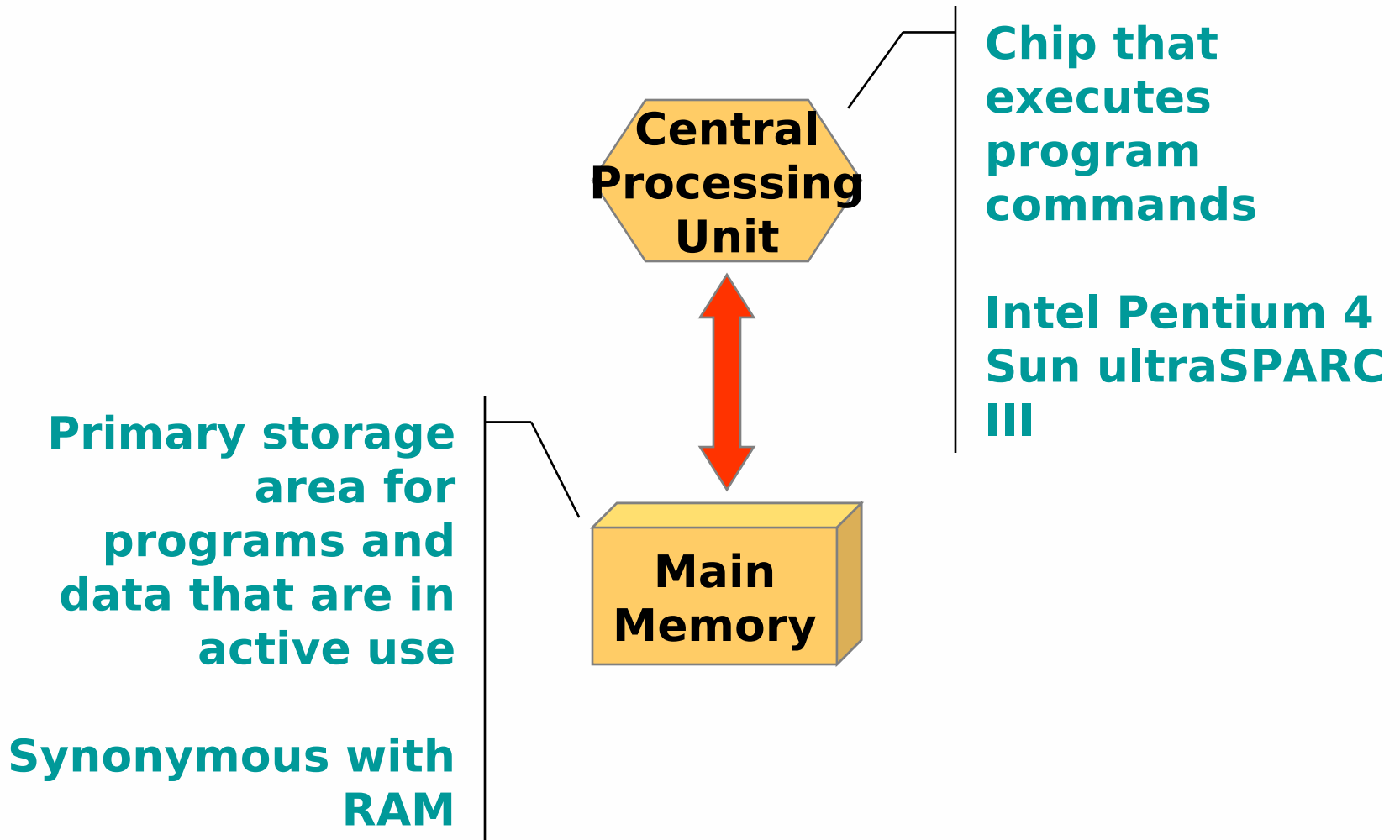
**Program Development**

**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



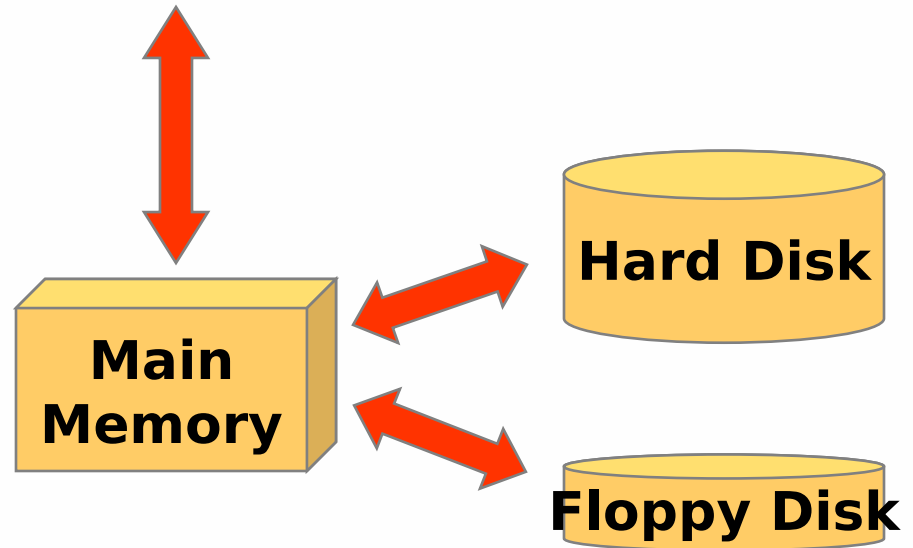
# Secondary Memory Devices

Secondary memory devices provide long-term storage

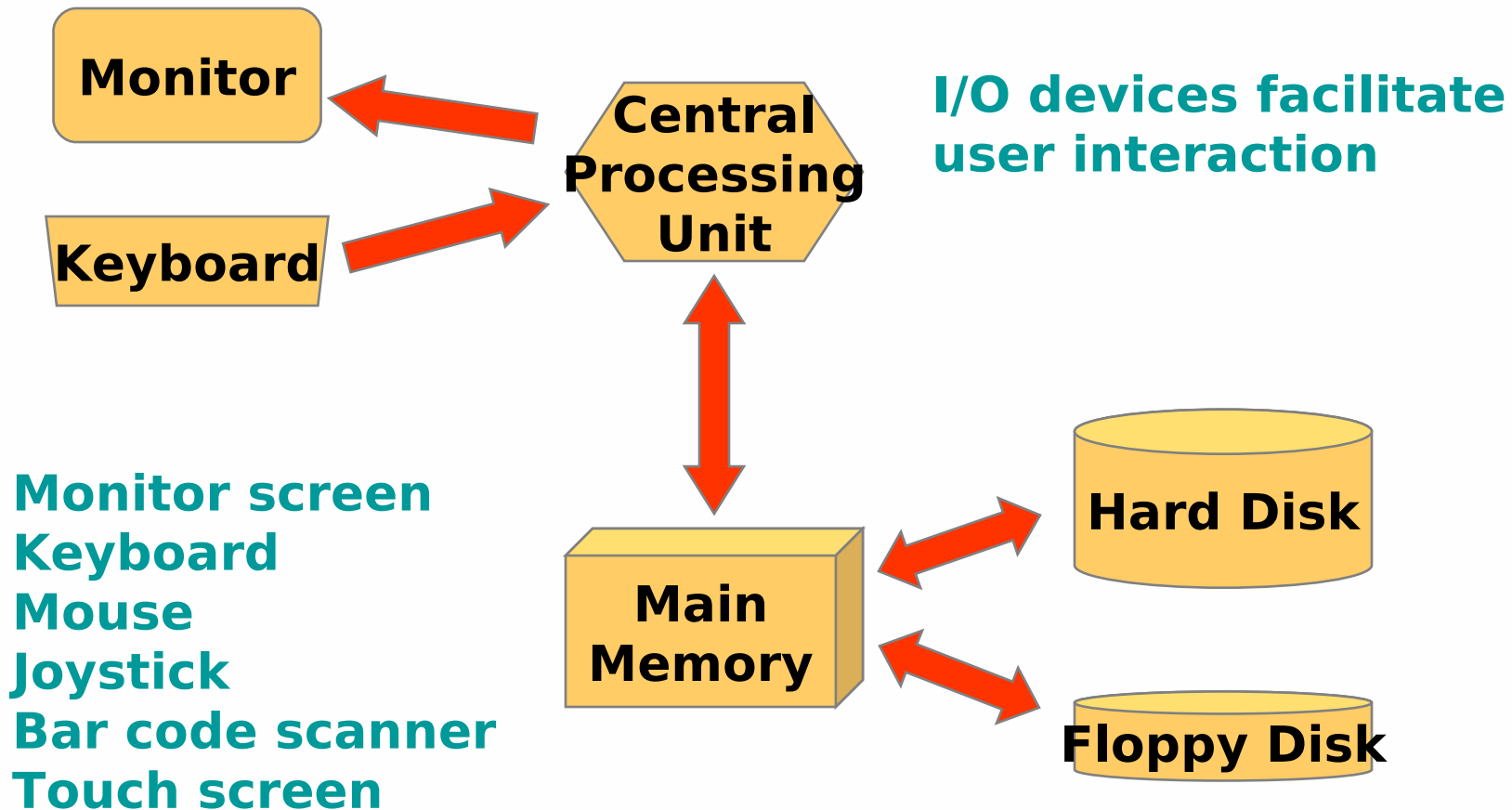
Central Processing Unit

Information is moved between main memory and secondary memory as needed

Hard disks  
Floppy disks  
ZIP disks  
Writable CDs  
Writable DVDs  
Tapes



# Input / Output Devices



# Software Categories

- **Operating System**
  - controls all machine activities
  - provides the user interface to the computer
  - manages resources such as the CPU and memory
  - Windows XP, Unix, Linux, Mac OS
- **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)**

# Analog vs. Digital

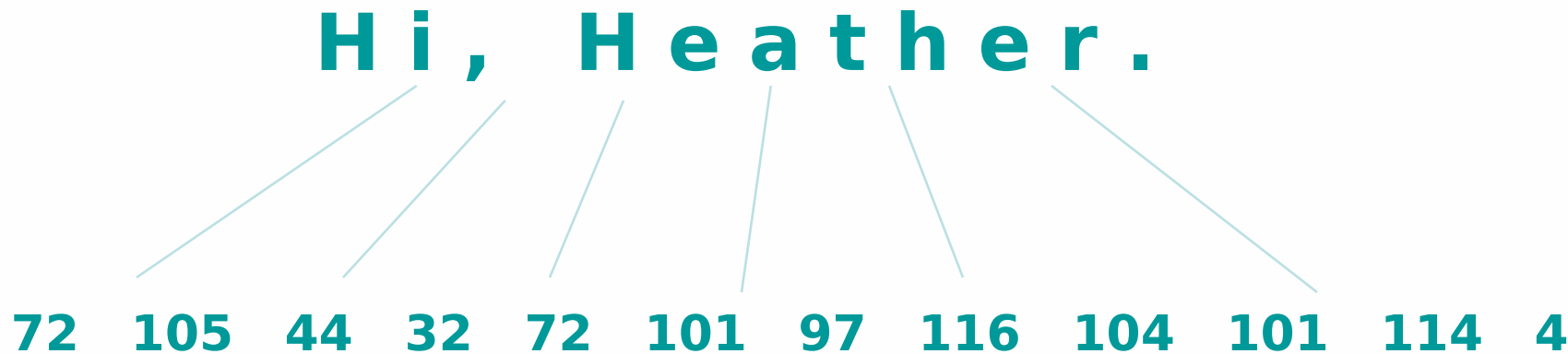
- **There are two basic ways to store and manage data:**
- ***Analog***
  - **continuous, in direct proportion to the data represented**
  - **music on a record album - a needle rides on ridges in the grooves that are directly proportional to the voltages sent to the speaker**
- ***Digital***
  - **the information is broken down into pieces, and each piece is represented separately**
  - **music on a compact disc - the disc stores numbers representing specific voltage levels sampled at specific times**

# Digital Information

- **Computers store all information digitally:**
  - numbers
  - text
  - graphics and images
  - video
  - audio
  - 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



# Binary Numbers

- **Once information is 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

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

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

# Outline

**Computer Processing**



**Hardware Components**

**Networks**

**The Java Programming Language**

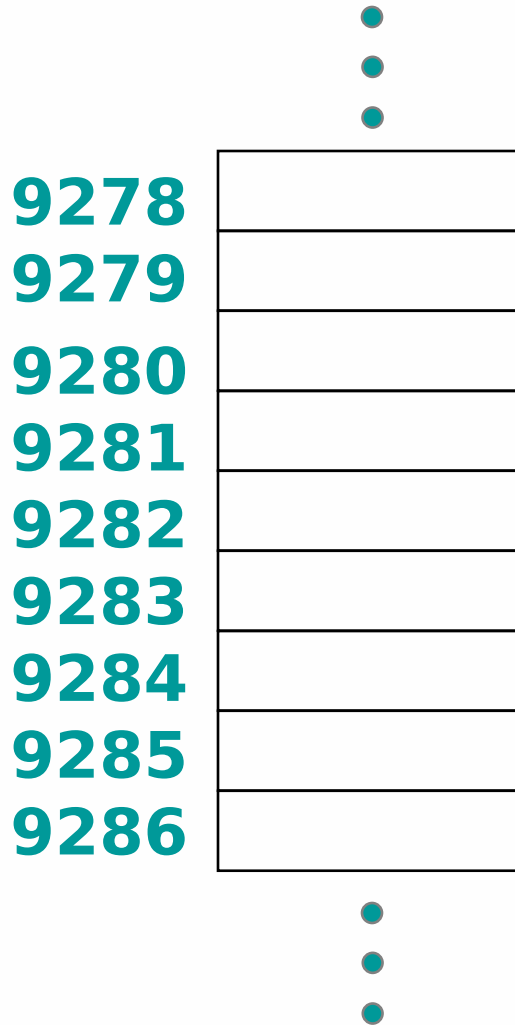
**Program Development**

**Object-Oriented Programming**

# A Computer Specification

- **Consider the following specification for a personal computer:**
  - **2.8 GHz Pentium 4 Processor**
  - **512 MB RAM**
  - **80 GB Hard Disk**
  - **48x CD-RW / DVD-ROM Combo Drive**
  - **17" Video Display with 1280 x 1024 resolution**
  - **56 Kb/s Modem**
- **What does it all mean?**

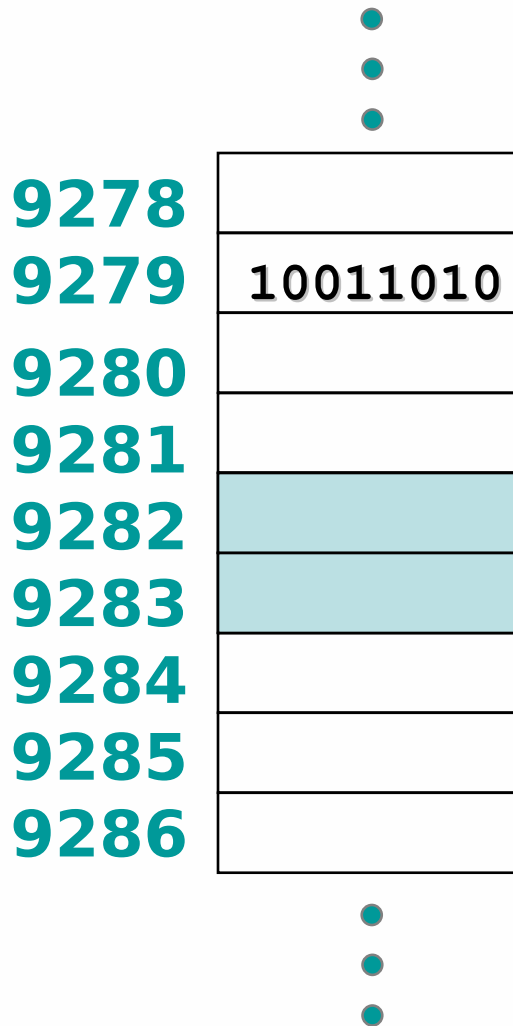
# Memory



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

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

# Storing Information



— 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:

<u>Unit</u>	<u>Symbol</u>	<u>Number of Bytes</u>
kilobyte	KB	$2^{10} = 1024$
megabyte	MB	$2^{20}$ (over 1 million)
gigabyte	GB	$2^{30}$ (over 1 billion)
terabyte	TB	$2^{40}$ (over 1 trillion)

# Memory

- **Main memory is *volatile* - stored information is lost if the electric power is removed**
- **Secondary memory devices are *nonvolatile***
- **Main memory and disks are *direct access* devices - information can be reached directly**
- **The terms *direct access* and *random access* often are used interchangeably**
- **A magnetic tape is a *sequential access* device since its data is arranged in a linear order - you must get by the intervening data in order to access other information**

# RAM vs. ROM

- **RAM** - Random Access Memory (direct access)
- **ROM** - Read-Only Memory
- The terms RAM and main memory are basically interchangeable
- ROM could be a set of memory chips, or a separate device, such as a CD ROM
- Both RAM and ROM are random (direct) access devices!
- RAM probably should be called Read-Write Memory

# Compact Discs

- **A CD-ROM is portable read-only memory**
- **A microscopic pit on a CD represents a binary 1 and a smooth area represents a binary 0**
- **A low-intensity laser reflects strongly from a smooth area and weakly from a pit**
- **A CD-Recordable (CD-R) drive can be used to write information to a CD once**
- **A CD-Rewritable (CD-RW) can be erased and reused**
- **The speed of a CD drive indicates how fast (max) it can read and write information to a CD**

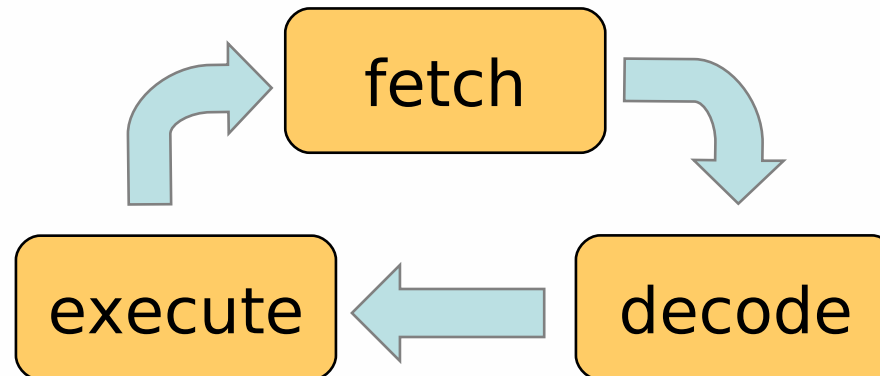
# DVDs

- **A DVD is the same size as a CD, but can store much more information**
- **The format of a DVD stores more bits per square inch**
- **A CD can store 650 MB, while a standard DVD can store 4.7 GB**
  - **A double sided DVD can store 9.4 GB**
  - **Other advanced techniques can bring the capacity up to 17.0 GB**
- **Like CDs, there are DVD-R and DVD-RW discs**

# The Central Processing Unit

- A CPU is on a chip called a *microprocessor*
- It continuously follows the *fetch-decode-execute cycle*:

Retrieve an instruction from main memory

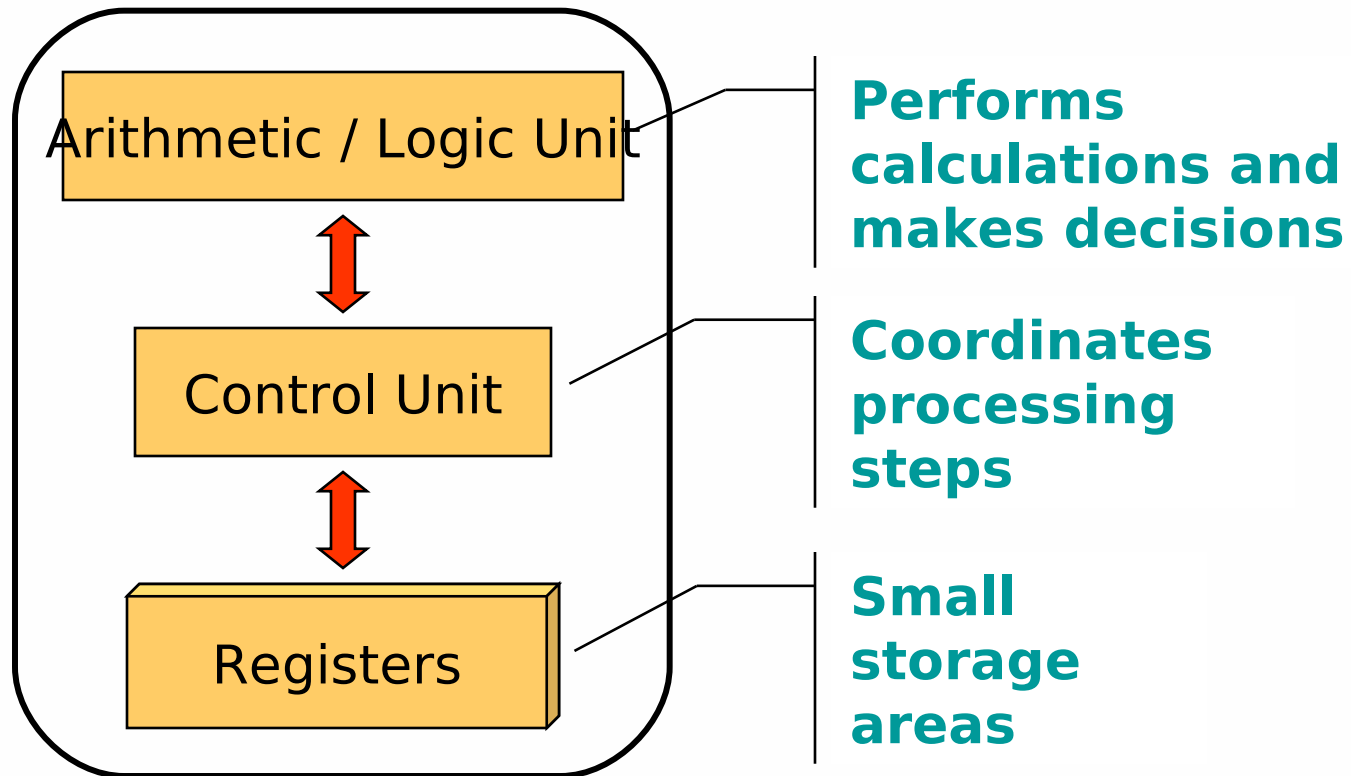


Carry out the instruction

Determine what the instruction is

# The Central Processing Unit

- **The CPU contains:**



# The Central Processing Unit

- The speed of a CPU is controlled by the *system clock*
- The system clock generates an electronic pulse at regular intervals
- The pulses coordinate the activities of the CPU
- The speed is usually measured in *gigahertz* (GHz)

# Monitor

- **The size of a monitor (17") is measured diagonally, like a television screen**
- **Most monitors these days have *multimedia* capabilities: text, graphics, video, etc.**
- **A monitor has a certain maximum *resolution* , indicating the number of picture elements, called *pixels*, that it can display (such as 1280 by 1024)**
- **High resolution (more pixels) produces sharper pictures**

# Modem

- ***Data transfer devices* allow information to be sent and received between computers**
- **Many computers include a modulator-demodulator or *modem*, which allows information to be moved across a telephone line**
- **A data transfer device has a maximum *data transfer rate***
- **A modem, for instance, may have a data transfer rate of *56,000 bits per second (bps)***

# Outline

**Computer Processing**

**Hardware Components**



**Networks**

**The Java Programming Language**

**Program Development**

**Object-Oriented Programming**

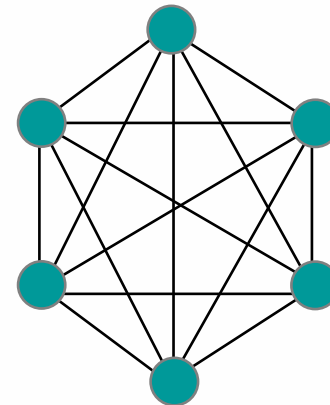
# Networks

- **A *network* is two or more computers that are connected so that data and resources can be shared**
- **Most computers are connected to some kind of network**
- **Each computer has its own *network address*, which uniquely identifies it among the others**
- **A *file server* is a network computer dedicated to storing programs and data that are shared among network users**

# Network Connections

- Each computer in a network could be directly connected to every other computer in the network
- These are called *point-to-point* connections

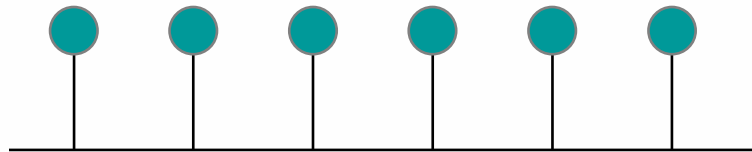
Adding a computer requires a new communication line for each computer already in the network



This technique is not practical for more than a few close machines

# Network Connections

- Most networks share a single communication line
- Adding a new computer to the network is relatively easy

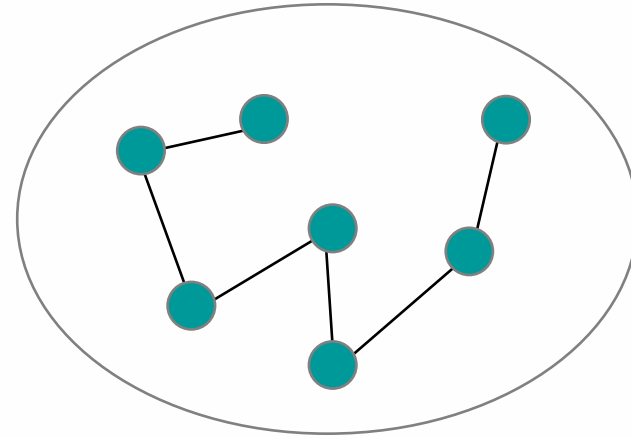


Network traffic must take turns using the line, which introduces delays

Often information is broken down in parts, called *packets*, which are sent to the receiving machine and then reassembled

# Local-Area Networks

***A Local-Area Network (LAN) covers a small distance and a small number of computers***

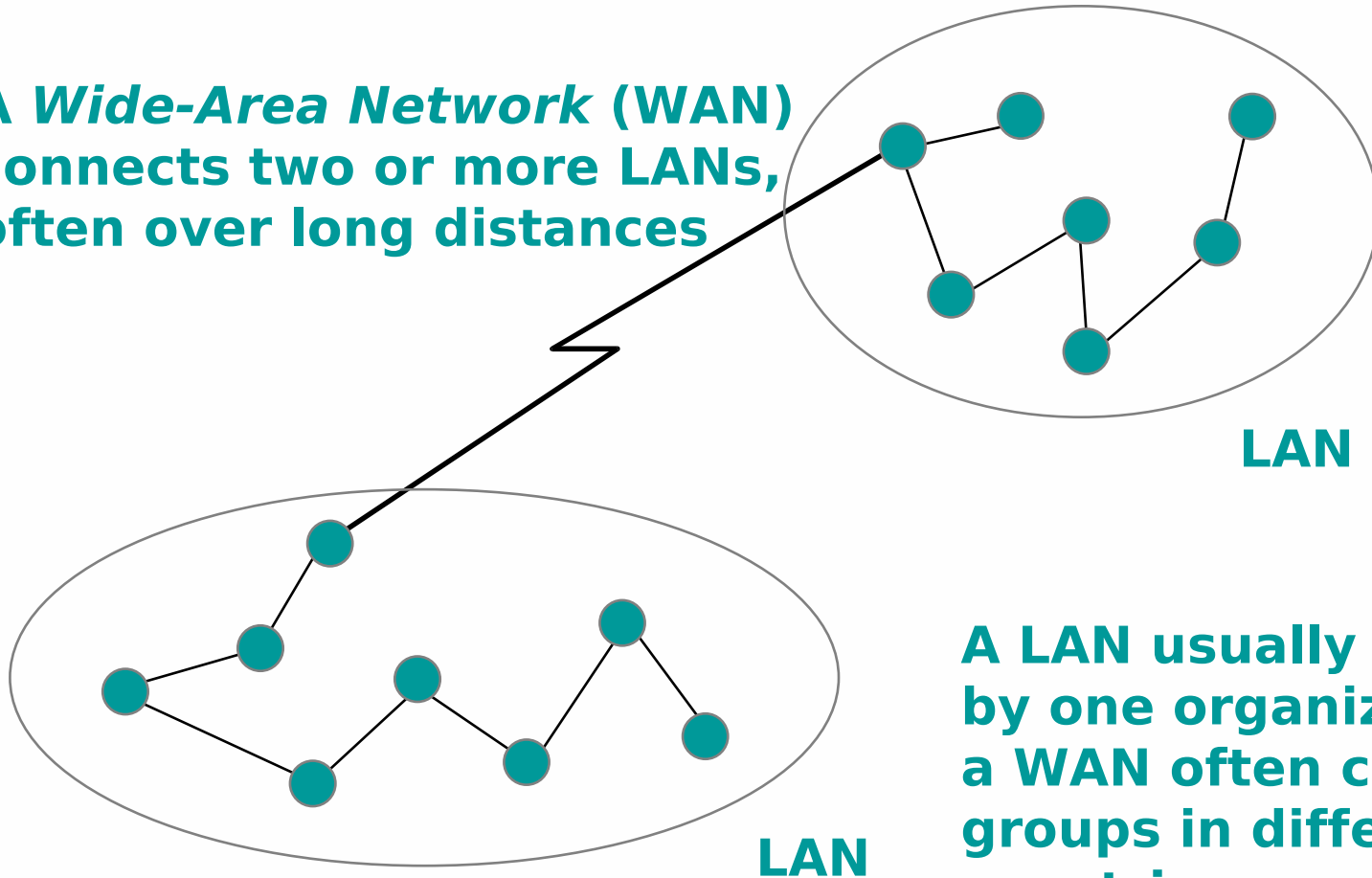


**LAN**

**A LAN often connects the machines in a single room or building**

# Wide-Area Networks

**A Wide-Area Network (WAN) connects two or more LANs, often over long distances**



**A LAN usually is owned by one organization, but a WAN often connects groups in different countries**

# The Internet

- The *Internet* is a WAN which spans the entire planet
- The word Internet comes from the term *internetworking*
- It started as a United States government project, sponsored by the Advanced Research Projects Agency (ARPA) - originally it was called the ARPANET
- The Internet grew quickly throughout the 1980s and 90s

# TCP/IP

- A protocol is a set of rules that determine how things communicate with each other
- The software which manages Internet communication follows a suite of protocols called *TCP/IP*
- The *Internet Protocol* (IP) determines the format of the information as it is transferred
- The *Transmission Control Protocol* (TCP) dictates how messages are reassembled and handles lost information

# IP and Internet Addresses

- Each computer on the Internet has a unique *IP address*, such as:

204.192.116.2

- Most computers also have a unique Internet name, which also is referred to as an *Internet address*:

spencer.villanova.edu

kant.gestalt-llc.com

- The first part indicates a particular computer (spencer)
- The rest is the *domain name*, indicating the organization (villanova.edu)

# Domain Names

- The last part of a domain name, called a *top-level domain* (TLD), indicates the type of organization:

**edu** - educational institution  
**com** - commercial entity  
**org** - non-profit organization  
**net** - network-based organization

**Sometimes the suffix indicates the country:**

**uk** - United Kingdom  
**au** - Australia  
**ca** - Canada  
**se** - Sweden

**New TLDs have recently been added:**

**biz, info, tv, name**

# Domain Names

- A domain name can have several parts
- Unique domain names mean that multiple sites can have individual computers with the same local name
- When used, an Internet address is translated to an IP address by software called the *Domain Name System* (DNS)
- There is no one-to-one correspondence between the sections of an IP address and the sections of an Internet address

# The World Wide Web

- The *World Wide Web* allows many different types of information to be accessed using a common interface
- A *browser* is a program which accesses and presents information
  - text, graphics, video, sound, audio, executable programs
- A Web document usually contains *links* to other Web documents, creating a *hypermedia* environment
- The term Web comes from the fact that information is not organized in a linear fashion

# The World Wide Web

- Web documents are often defined using the *HyperText Markup Language* (HTML)
- Information on the Web is found using a *Uniform Resource Locator* (URL):

`http://www.lycos.com`

`http://www.villanova.edu/webinfo/domains.html`

`ftp://java.sun.com/applets/animation.zip`

- A URL indicates a protocol (http), a domain, and possibly specific documents

# Outline

**Computer Processing**

**Hardware Components**

**Networks**



**The Java Programming Language**

**Program Development**

**Object-Oriented Programming**

# Java

- **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***
- **The Java programming language was created by Sun Microsystems, Inc.**
- **It was introduced in 1995 and it's popularity has grown quickly since**

# 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` (page 28)

# 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 in a program are called *inline documentation*
- They 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 a programmer uses in a program
- An 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 we choose identifiers ourselves when writing a program (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:

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

# 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` (page 34)
- See `Lincoln3.java` (page 35)

# 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**

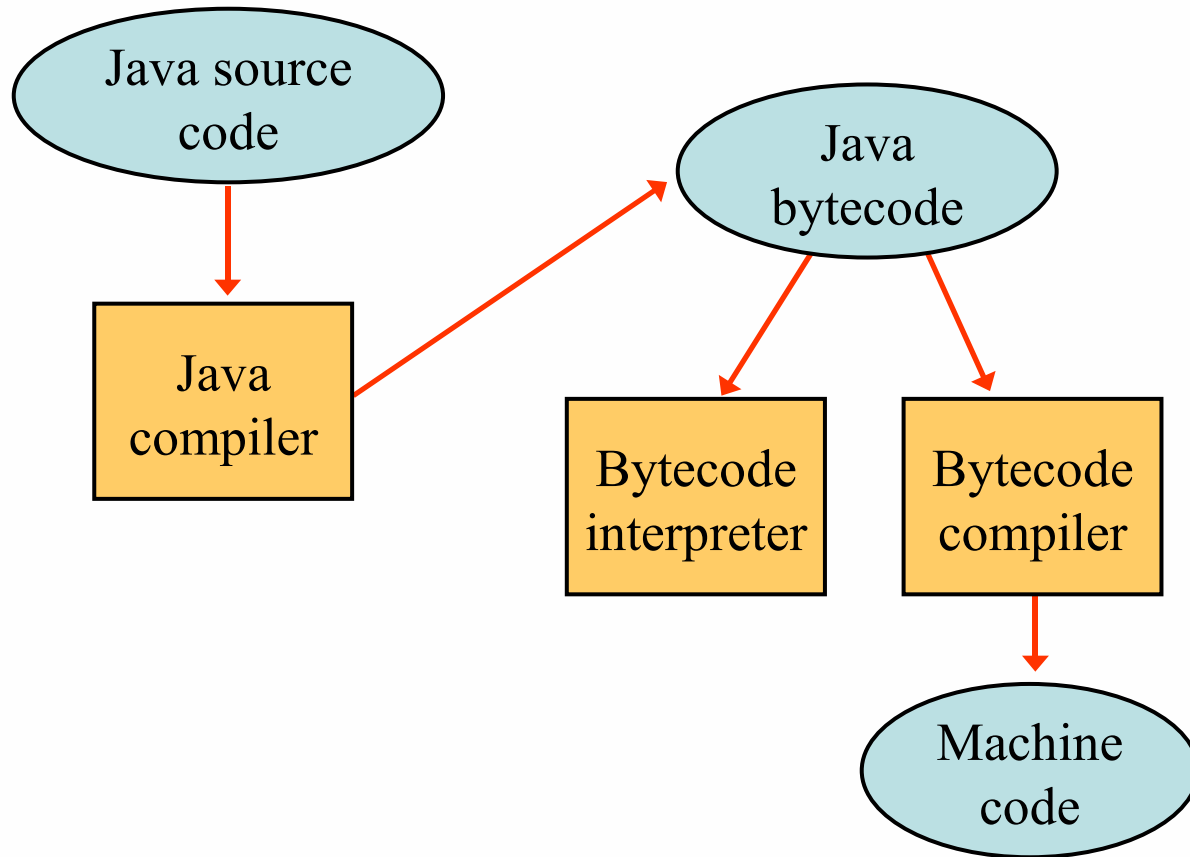
# Programming Languages

- 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
- Often, 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
- Another software tool, called an *interpreter*, translates bytecode into machine language and executes it
- Therefore the Java compiler 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:**
  - **Sun Java Development Kit (JDK)**
  - **Sun NetBeans**
  - **IBM Eclipse**
  - **Borland JBuilder**
  - **MetroWerks CodeWarrior**
  - **Monash BlueJ**
- **Though the details of these environments differ, the basic compilation and execution process is essentially the same**

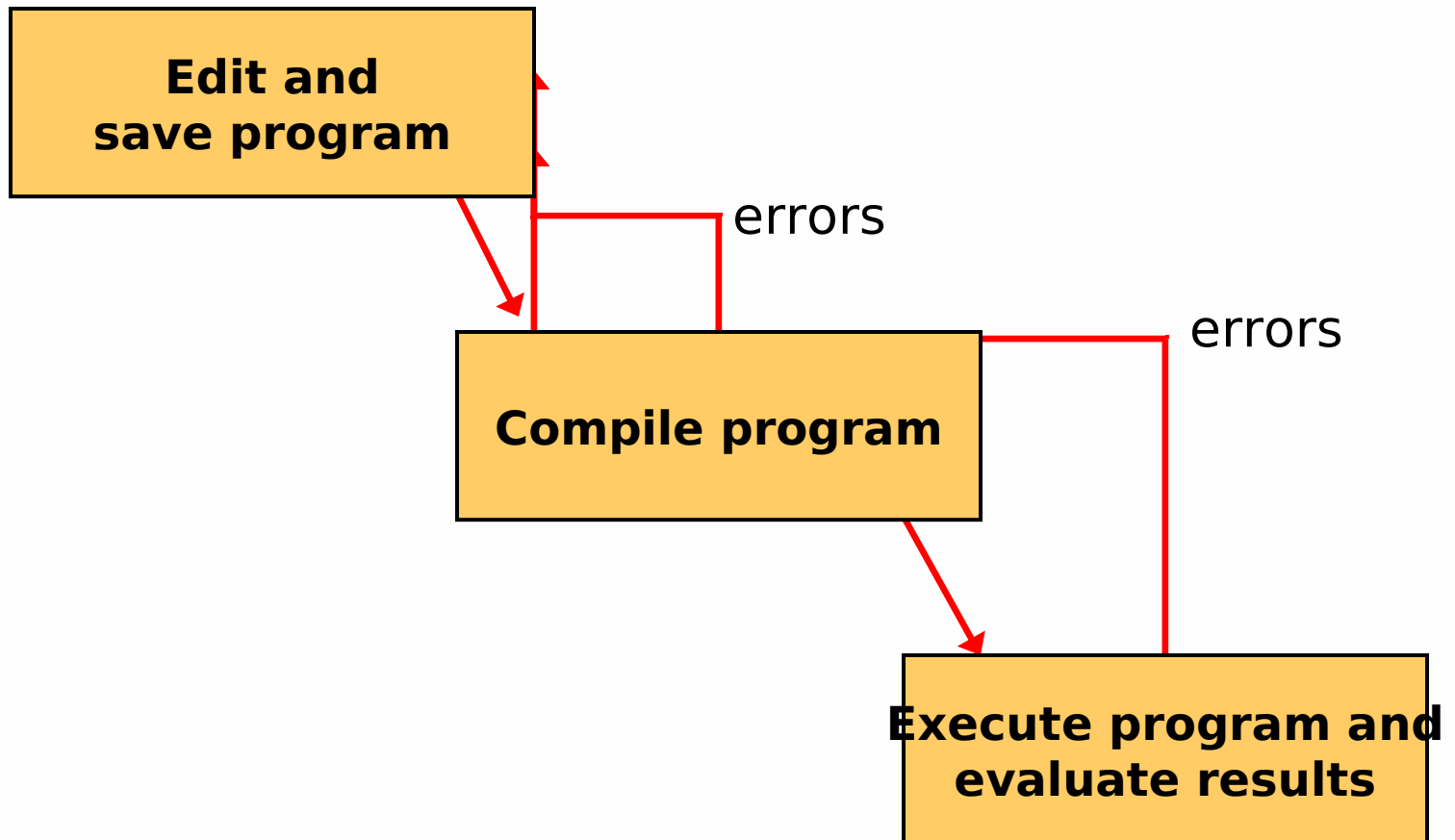
# Syntax and Semantics

- 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 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**

# Objects and Classes

**A class  
(the concept)**

Bank  
Account



**Multiple objects  
from the same class**



**An object  
(the realization)**

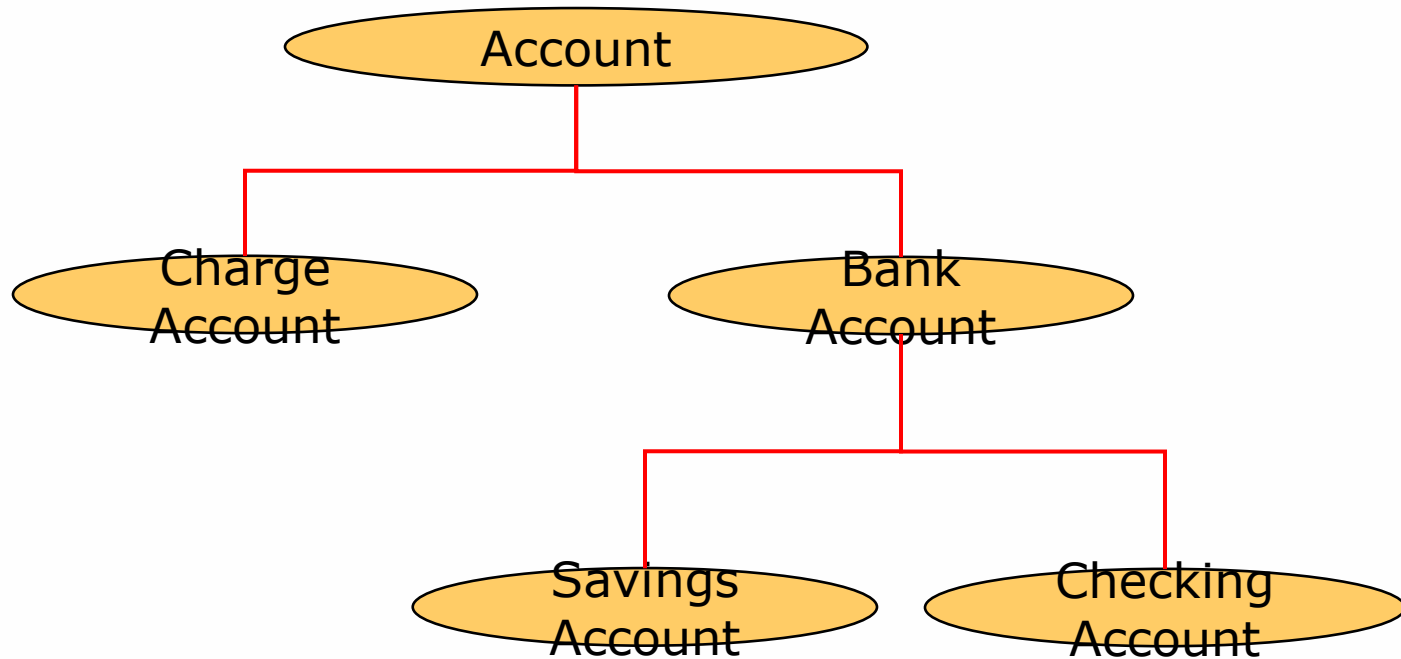
John's Bank Account  
Balance: \$5,257

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**