# COMP 110-001
# Programming Basics

Yi Hong

May 14, 2015

# Announcements

- The slides of the previous two lectures have been updated on the course website
- There are seven custom textbooks available in the university book store, we can order more if needed

# Review

- Binary to decimal conversion
  - 10110
- Is Java a high-level language or a low-level language?
- How is "Write once, run everywhere" achieved in Java?
- What's output of the following Java statement?
  - System.out.println("Java is great!");

# Today

- Object-Oriented Programming (OOP)
- Intro to encapsulation, polymorphism, and inheritance
- Writing algorithms in pseudocode
- Variables, arguments, statements, and syntax

# Object-Oriented Programming

- Our world is made up of objects, e.g., people, buildings, trees, cars, cabbages
- Objects can perform actions, which affect themselves and other objects in the world
- Object-oriented programming (OOP) treats a program as a collection of objects that interact by means of actions

# OOP Terminology

- Objects: have attributes and behaviors
- Methods: each behavior is called a method
- Class: Objects of the same kind have the same type and belong to the same class
  - Objects within a class have a common set of methods and the same kinds of data
  - But each object can have its own data values
- An example: students in this class
  - Attributes: name, age, major, hobbies …
  - Actions: talk, submit assignment, play games …

# Why OOP?

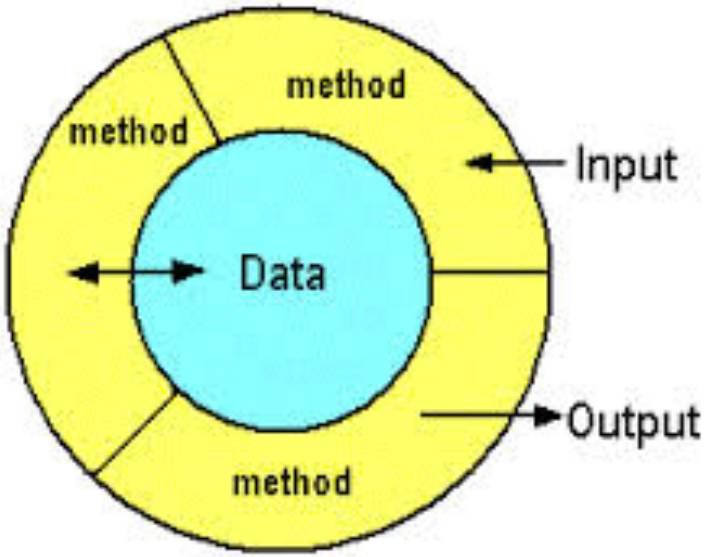
- When we build a large system, we usually divide the whole system into small pieces
- Uses objects as building blocks and “object interaction” in programming
- Each object only cares about the interactions with them
- An example: this lecture
  - Objects: instructor, students …
  - Object interaction: discuss, submit assignments …

# OOP Design Principles

- Three primary design principles
  - Encapsulation
  - Polymorphism
  - Inheritance

# Encapsulation

- Packages things up and hides details
- Provides how to use a class, but omits all the details of how it works
- Encapsulation is often called information hiding



# An Example

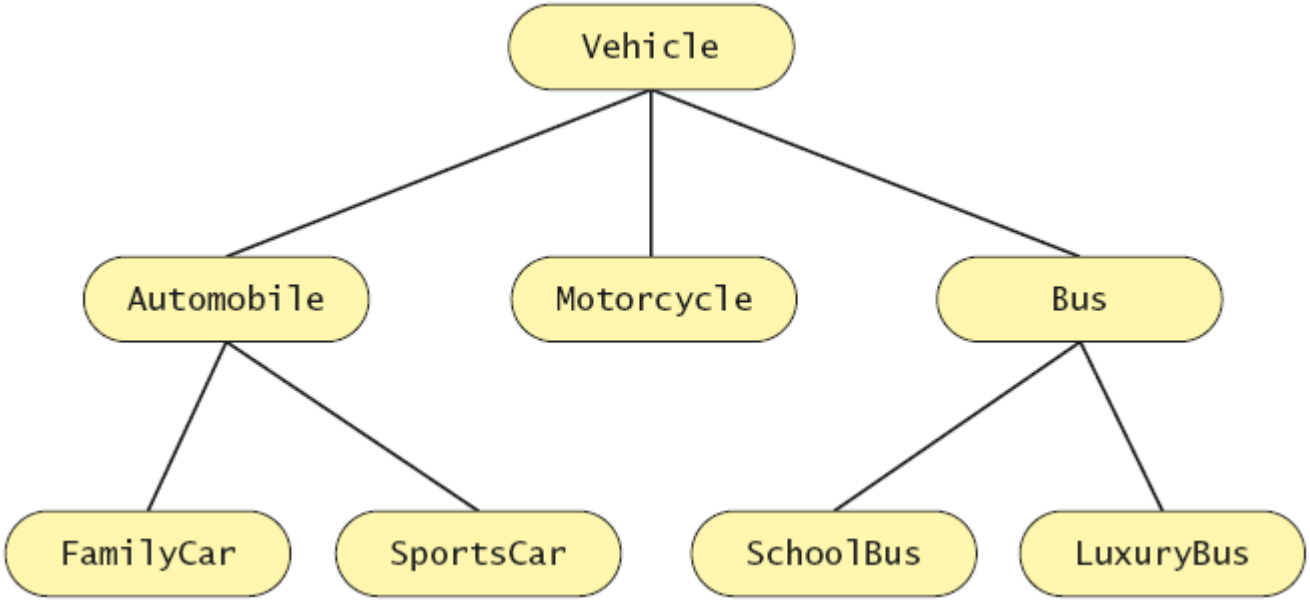
- An automobile consists of several parts and is capable of doing many useful things
  - Awareness of the accelerator pedal, the brake pedal, and the steering wheel is important to the driver
  - Hiding information: awareness of the fuel injectors, the automatic braking control system, the power steering pump is *NOT* important to the driver.

# Polymorphism

- "many forms"
- Allows the same program instruction to mean different things in different contexts
  - Example: "Go play your favorite sport"
    - To one person, it means to play baseball
    - To another person, it means to play soccer
- In programming, polymorphism means that one method name, used as an instruction, can cause different actions
- More details in Chapter 8

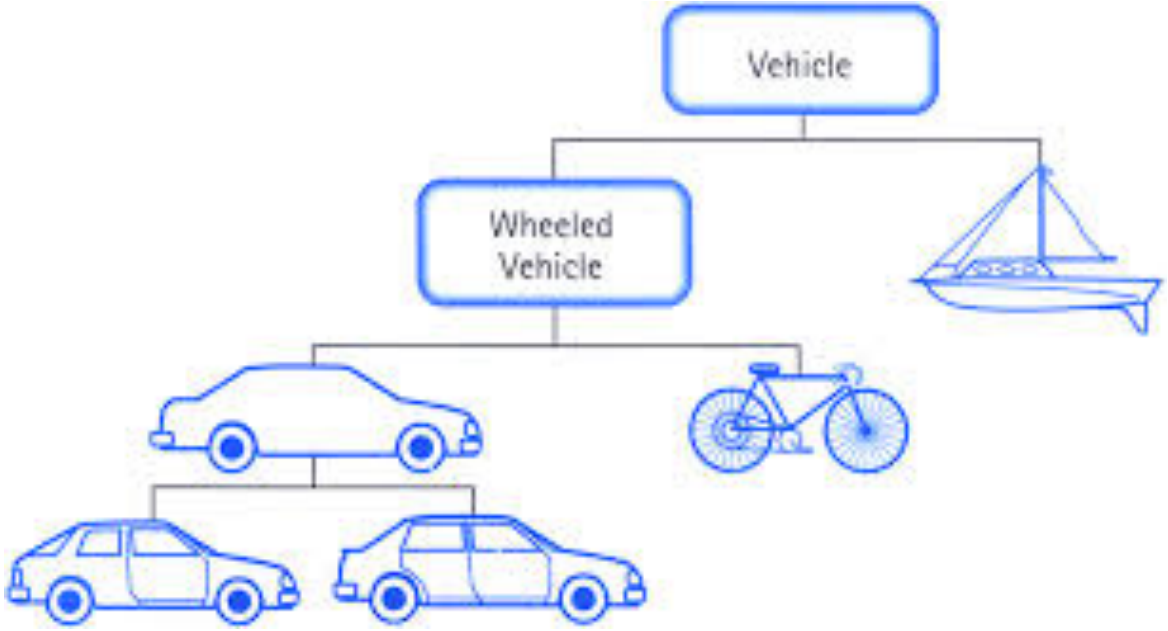
# Inheritance

- A way of organizing related classes
- At each level, the classifications become more specialized



# More About Inheritance

- A class at higher level is more general
- A class at lower level is more specific, and it inherits all the characteristics of classes above it in the hierarchy



# Algorithm

- A set of instructions for solving a problem
  - The directions must be expressed completely and precisely
- By designing methods, programmers provide actions for objects to perform
- In this course, we mainly consider a sequence of instructions in a method (one action)

# Pseudocode

- Before translating algorithm into code
- A mixture of English and Java
- Write each part of an algorithm in whatever language is easiest for you
- An example:

```
For each student A in the class:
    If A's score > 60   Print A + "has passed"
    Else Print A + "has Failed"
```

# Vocabulary

- Variables: store a piece of data. Think a variable as a container
- Statements: instructions, the smallest standalone element of a programming languages
- Syntax: grammar rules for a language
- Arguments: information that methods need to carry out its action

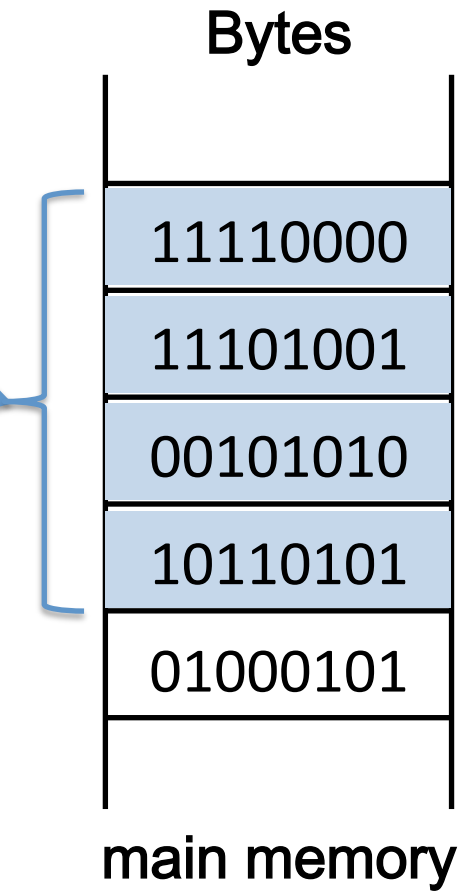
# Variables

```
int age;
age = 16;
```

- A variable is a program component used to store or represent data
- The data currently in a variable is its value
- Name of variable is an identifier
- Can change value throughout program
- Choose meaningful variable names!!!

# Variables and Memory

- A variable corresponds to a location in memory
  - Variable: int age
  - Use this cell to store the value of the variable “age”
  - Prevent this cell from being used by other variables later



# Variable Declarations

- Syntax
  - Type variable_1, variable_2, … ;
- Examples
  - int count, score, myInt;
  - double totalCost, ratio;
  - char letter, answer;

We will discuss the data types in Java in the next lecture

# How to Use Variables

- Declare a variable
  - int age;
- Assign a value to the variable
  - age = 16;
- Change the value of the variable
  - age = age + 1;

# How to Name an Identifier

- Letters, digits (0-9), the underscore character (_)
- The symbol $ is also allowed, but it is reserved for special purposes, so you should not use $ in a Java name
- First character *cannot* be a digit
- No spaces
- Java is case sensitive

Legal identifiers: pinkFloyd, b3Atlas, eyeColor

Illegal identifiers: michael.bolton, kenny-G, 1CP

# Keywords

- Reserved words with predefined meanings
- You cannot name your variables using keywords
- All Java keywords are entirely in lowercase
- We will learn them as we go along
  - E.g.: if, else, return, new, public, class, static, void …

# Next class

- Primitive type
- Class type