Programming Basics

Cheng, Wei  COMP110-001  May 14, 2014
Review of Previous Lecture

• Binary to Decimal Conversion
  – 10110
  – 1001001

• If I use 4 bytes to represent an integer (both positive & negative), what is the range?
  – Hint: use 1 bit for +/- sign

• How is “Write once, run everywhere” achieved in Java?

• What is System.out in Java?
Today’s contents

• Object Oriented Programming

• Algorithm & Pseudocode

• Variables
  – Integer & String
Object Oriented Programming (OOP)

• OOP (with OOD) is probably the most important concept in today’s software engineering

• What is OO?

• Think of our real-life world. It’s just interactions of objects (including abstract objects)
  – Borrow a book from library
  – This lecture
Object Oriented Programming (OOP)

• Identify the objects and the interactions
• For now, we only consider the objects at high level
  – Borrow a book from library
    • Member, Book, Library
  – This lecture
    • Lecturer, Students, Lecture Contents
  – Browse Internet
    • User, Browser, Websites
Object Oriented Programming (OOP)

• When we want to build large system, we would like to break the whole system into small pieces that are easy to maintain.

• Since the “object interaction” view is a natural representation of our world, we could just use objects as building blocks in programming.

• What are the important factors of an object? Let’s say a student in our class:
  – Internal attributes: Name, Age, Major, Hobbies, Knowledge in head...
  – Actions:
    • Actions that only affect internal attributes: grow older, do exercise, think about a problem and get answer in head, drink water*
    • Interaction with other objects: talk to other students, submit assignment to Prof. X, play videogame with Y...
Object Oriented Programming (OOP)

• Let’s say a student in our class:
  – Internal attributes: Name, Age, Major, Hobbies, Knowledge in head...
  – Actions:
    • Actions that only affect internal attributes: grow older, do exercise, think about a problem and get answer in head, drink water*
    • Interaction with other objects: talk to other students, submit assignment to Prof. X, play videogame with Y...

• What does it mean to another object?
  – To a professor?
  – To a friend?

• Other objects only care about the specific interaction(s) with them
• This greatly simplifies the design and reduces maintenance effort.
Object Oriented Programming (OOP)

• Formally, we consider each object to be defined by:
  – Attributes
    • the values that define the state of an object
  – Actions (behaviors in textbook)
    • We call them **methods** in Java
    • Can update state (change internal attribute values)
    • Can represent interaction with other object
    • Or both
Object Oriented Programming (OOP)

• Wait, we want the design to be more elegant:

• For objects of the same type:
  – aren’t they sharing the same set of attributes (not values) or methods?

  – Student example
  – Cars (Ferrari or Beetle):
    • Fuel level, current speed, owner, plate number, year, model...
    • Accelerate, brake, lock/unlock ....
Object Oriented Programming (OOP)

- Objects of the same type should have the same set of attributes and methods:
  - We define a high-level abstract concept called Class.
  - Class defines the set of attributes and methods.
  - Each Object is an instance of a Class
  - Objects under the same Class have the same set of attributes / methods.
Object Oriented Programming (OOP)

- Class – corresponds to type/kind concept in real world
- Classes are organized in a hierarchy naturally
- The example in the textbook:
Object Oriented Programming (OOP)

• A Class at higher level is more general
• A Class at lower level is more specific – and it inherits all the characteristics (attributes & methods) of parent classes in the hierarchy.

• Super-class & sub-class

Diagram:
- **Vehicle**
  - **Automobile** (Having 4 wheels)
  - **Motorcycle** (Having 2 wheels)
  - **Bus** (Having at least 4 wheels)
    - **FamilyCar** (Having 8 wheels)
    - **SportsCar**
    - **SchoolBus**
    - **LuxuryBus** (Having 16 wheels)
Object Oriented Programming (OOP)

- Let’s gather all concepts
- Class, Super-class & sub-class, Object (instance of class)
Object Oriented Programming (OOP)

• Let’s go back to Java:
• A program is a collection of objects that interact by methods.
  – Determine the types/kinds of objects we are going to work with.
    • Define Classes or use existing Classes (import …)
    – Create objects (instances of Classes) as needed
    – Invoke objects method as needed

import java.util.*;

Scanner s = new Scanner(System.in);
String name = s.next();
Algorithm

• A sequence of instructions for solving a problem
  – Expressed completely and precisely

• Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.

• In this course, we mainly consider sequence of instructions inside a method (one action)
Pseudocode

• Combination of code and English used to express an algorithm **before** writing algorithm into code

• For 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. Can be updated. Think a variable as a container.

• **Statements** - the smallest standalone element of a programming language (don’t worry about its tricky definition)

• **Syntax** - grammar rules for a language
Variables

• Defining variables:
  
  type variable_name;
  
  int i;
  
  type variable_name = initial_value;
  
  int i = 1;
  
  type variable_name1, variable_name2, .... ;
  
  int i,j,k;

• int : built-in integer type
• Note the syntax.
Using int Variables

```c
int i;
```

- Once a variable has been declared, you do not have to repeat the type in using it.

- Assignment:
  ```c
  i = 2;
  i = i + 1;
  j = 3;
  i = j * i;
  ```

- You can apply all numerical operations with int:
  ```c
  +, -, *, / ....
  ```
  But the result is restricted to int
Using String Variables

• Declare String variables:

```java
String FamilyName;
String FirstName = "Wei";
```

Assignment:

```java
FamilyName = "Cheng";
```

For now, the only important operation on `String` you need to remember is concatenation (by `+`).

```java
String FullName = FirstName + " " + FamilyName;
```

• The concatenation can combine a string with an integer. The result will be a string.

• What are the values of the expressions below?
  
  "1" + 2  
  1 + 2  
  "1" + "2"
Using Variables

• The left side and right side of an assignment must be of the same type.

```java
int i;
i = "1" + 2; // this will not run

String t;
t = 1 + 2; // this will not run
```

• Variables can be used in expression or passed into methods directly. Recall our First & Second Java Program in previous lecture

```java
String name;
...
System.out.println( name + "\n, welcome to COMP 110" );
```