Tutorial 2:
Advanced Object Oriented Concepts

Xiangnan He
3rd Sep. 2013

xiangnan@comp.nus.edu.sg
Sharing Experience

• How to learn JAVA well?
  – 1. More practice!!
    a) Implement the codes of your lectures & tutorials & labs (write the codes by yourself, rather than just copy-paste)
    b) Know how does the program run.
  – 2. Be clear about the meaning of each line.
    a) For any questions, you can refer the book, search the web, discuss with others, ask the teacher/TA etc.

• To conclude:
  – 1. Get the program run;
  – 2. Know the running steps of the program;
  – 3. More practice to be familiar with the API and the codes.
Lecture Review Outline

• Class VS. Object
• Static variables VS. non-Static variables
• Method overloading
• Type system of Java:
  – Primitive types and Wrapper classes
  – Recap the function call: Pass-by-value
• OO Design Principles:
  – Abstraction, Information Hiding, Coherence, Coupling…
Class VS. Objects

• Class denotes a **new data type (like a template)**
  – No actual instances are allocated
• Object is the **instance/variable** of a class
• Example:

```java
//Create a class named House
public class House {
    double size;
    public House(double size){
        this.size = size;
    }
    public void printSize(){
        System.out.println("Size is "+size);
    }
    public static void main(String[] args) {
        House h; //Declare a variable h of type House
        h = new House(100); //Allocate memory space of h
        h.printSize(); //Call the member function of h
    }
}
```
Static variables VS. Non-Static variables

• **Static variable belongs to class**
  – All objects of the class share the same variable

• **Non-Static variable belongs to object**
  – One object has one specific variable

```java
public class Singaporean {
    public static String nationality = "Singapore";
    public String name;
    public Singaporean(String name) {
        this.name = name;
    }

    public static void main(String[] args) {
        Singaporean David = new Singaporean("David");
        Singaporean Tommy = new Singaporean("Tommy");
        System.out.println("My name is "+David.name+", nationality is "+David.nationality);
        System.out.println("My name is "+Tommy.name+", nationality is "+Tommy.nationality);
    }
}
```
Test of Static VS. Non-Static variables

```java
public class Test {
    static int value1 = 1;
    int value2 = 1;
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Test obj1 = new Test();
        Test obj2 = new Test();
        Test obj3 = new Test();

        System.out.print("obj1: ");
        System.out.print(obj1.value1++ + "\t");
        System.out.println(obj1.value2++);

        System.out.print("obj2: ");
        System.out.print(obj2.value1++ + "\t");
        System.out.println(obj2.value2++);

        System.out.print("obj3: ");
        System.out.print(obj3.value1++ + "\t");
        System.out.println(obj3.value2++);
    }
}
```

What’s the output?

- obj1: 1 1
- obj2: 2 1
- obj3: 3 1

More explanation using the memory allocation mechanisms.
Example of Method Overloading

• Overloading methods:
  – 2 or more methods **within the same** class (**i.e.**, have the same action scope) **have the same name**, but **different parameters**;
  – When the method name is called, which one is used depends on the parameters.
  – Example:

```java
public static void f(int a, int b) {
    System.out.println(a + b);
}

public static void f(double a, double b) {
    System.out.println(a - b);
}
```

  – What are the outputs of the following codes?

```java
f(3, 6);
f(3.0, 6.0);
f(3, 6.0);
```
Type System of Java

- There are only 2 categories of types in Java:
  - (a) 8 primitive data types: `byte, short, int, long, float, double, boolean, and char`.
  - (b) Other than the 8 primitive data types (including Array, Wrapper class, Self-defined types, etc) in Java are in **object** form.
- Each primitive data type corresponds to a wrapper class

<table>
<thead>
<tr>
<th>Primitive Data Type</th>
<th>Wrapper Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Byte</td>
</tr>
<tr>
<td>short</td>
<td>Short</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>long</td>
<td>Long</td>
</tr>
<tr>
<td>float</td>
<td>Float</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
</tr>
</tbody>
</table>
Function call

- All function call in Java are passing-by-value
  - For variables of the primitive data types, it just copies the value of the variable;
    - i.e., two variables points to different memory space.
  - For variables of other types, it copies the object reference (i.e., the address) of the variable.
    - i.e., two variables points to the same memory space.
public class MyInteger {
    int value;
    public MyInteger(int value) {
        this.value = value;
    }
}

public static void swap(int a, int b) {
    int temp = a;  a = b;  b = temp;
}

public static void swap(Integer a, Integer b) {
    Integer temp = a;  a = b;  b = temp;
}

public static void swap(MyInteger a, MyInteger b) {
    int t = a.value;
    a.value = b.value;
    b.value = t;
}

public static void main(String[] args) {
    int a = 1;  int b = 2;
    swap(a, b);
    System.out.println("a = " + a +", b = " + b);

    Integer aa = 1;  Integer bb = 2;
    swap(aa, bb);
    System.out.println("aa = " + aa +", bb = " + bb);

    MyInteger aaa = new MyInteger(1);
    MyInteger bbb = new MyInteger(2);
    swap(aaa, bbb);
    System.out.println("aaa = " + aaa.value +", bbb = " + bbb.value);
}

Problems @ Javadoc
<terminated> Test [Java App]
a = 1, b = 2
aa = 1, bb = 2
aaa = 2, bb = 1
Questions Discussion

• Q(a): UML
• Q(b): Data Encapsulation
• Q(c): Complete class Custom
• Q(d): Modify class CafeSystem to include the method findOrderID()
• Q(e): Modify the class CafeSystem to include an option to change a custom drink
Some basics about UML (Unified Modeling Language):
1. Graphic modeling techniques to show the design of OO systems.
2. Each node typically denotes a class.
3. Within each node are the variables/methods of the class: 
   [+/-] variable/method_name:type
4. Edge between two nodes means the relationship of the two classes.
5. Uni-directional association from A -> B means that A knows class B but B does not know class A
Q(b)

• Checklist for requirements:
  – 1. **Constructor** of the **Cafe** class:
    ➢ Initial amounts of espresso, foam, milk and drinks
  – 2. The **order()** method of **Cafe** class:
    ➢ When making an order, check if the orders are full
    ➢ When making an order, check if the materials are enough
  – 3. **Constructor** method of **Latte** and **Cappuccino** class:
    ➢ Generating orderID
  – 4. **Get()** methods of **Latte** and **Cappuccino** class:
    ➢ getMilk(), getFoam() and showIngredients()
  – 5. Overload the **order()** method for **Latte** and **Cappuccino** class
Q(c)

- Checklist for requirements:
  1. Complete the **Custom** class:
     - Declare the member variables
     - Complete the member functions
  2. What other classes to have to be modified?
     - Add an array of custom orders in the **Cafe** class
     - Overload the **order()** method to support the custom orders.
     - Add an option in the **main()** function of **CafeSystem** to allow the administrator to order a custom drink
Q(d)

- Search the order array for to find an order.
Q(e)

• Modify the class **CafeSystem** to include an option to change a custom drink.
  – Add a *case* branch of the *switch-case* statements.
  – Input an *orderID*:
    ➢ Check whether the *orderID* is valid
  – Input the customized *milk/foam* amount:
    ➢ Check whether there is enough materials
    ➢ Update the amount of materials of the **CafeSystem**.
    ➢ Update the amount of materials of the *customized order*. 
Thanks!
QA?