

§1.16-2.8: Selection structure

17 January 2007
CMPT167
Dr. Sean Ho
Trinity Western University

Review of last time

- **Languages**: machine, assembly, high-level
- Java code **translation**
- **JDK** vs. **JRE**
- A first Java **program**
- **Comments** and **doc-comments**
- **Compiling** and **running** a Java program

What's on for today

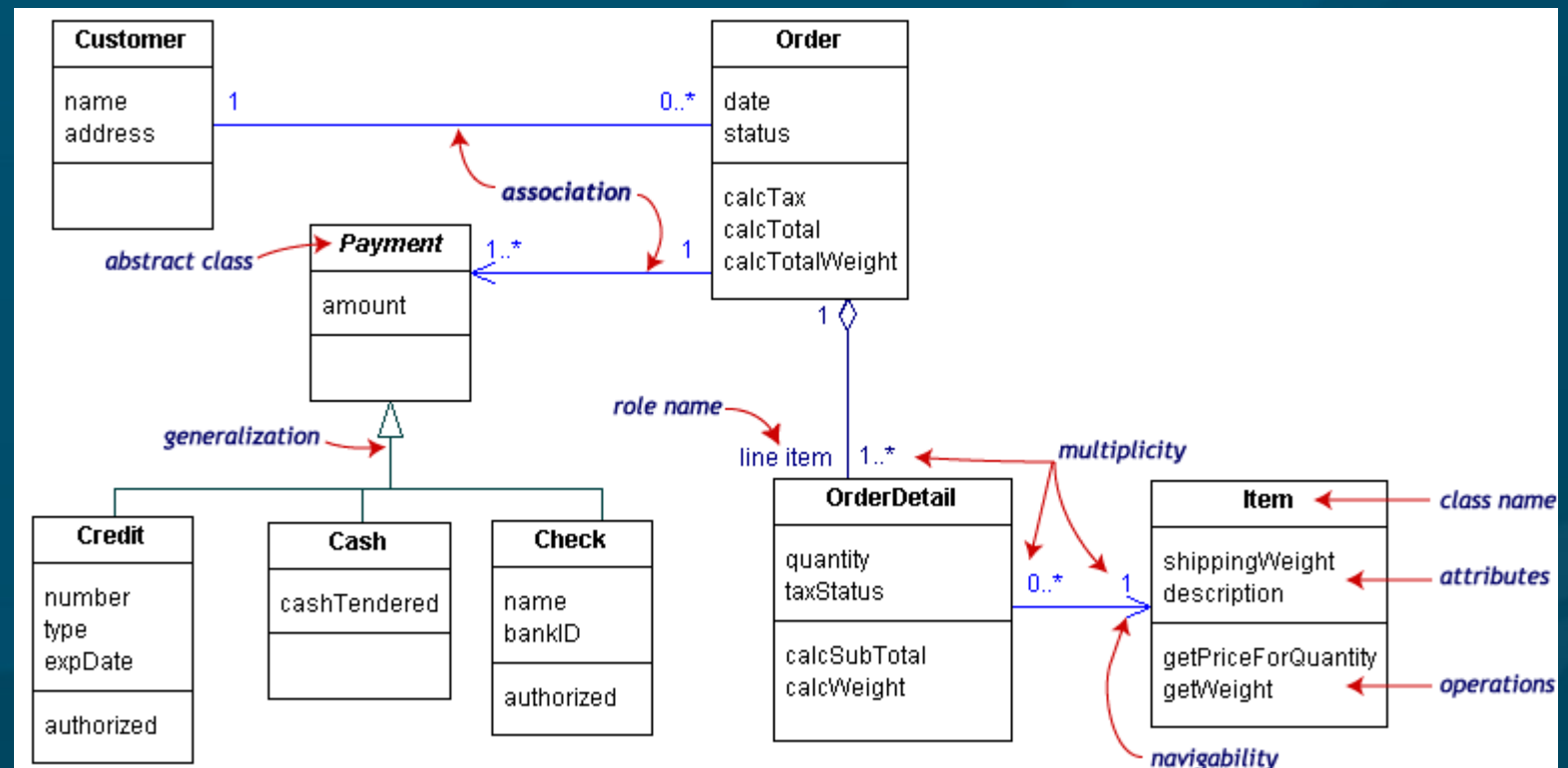
- UML: diagrams for software design
- Design patterns: not reinventing the wheel
- Java coding style, identifiers
- Booleans and the if statement
- Applets
- Swing

UML: Unified Modeling Language

- Diagrams for use in designing your programs
- Main diagram types:
 - Static: Class diagram, object, package
 - Dynamic: Use case diagram, sequence diagram, state chart
- Handy for diagramming by hand, or
- UML software tools, e.g., Visio, Sun JSEnterprise
- Developed by Booch, Rumbaugh, and Jacobson, of OMG (Object Management Group)
- Current version is 2.0: www.uml.org

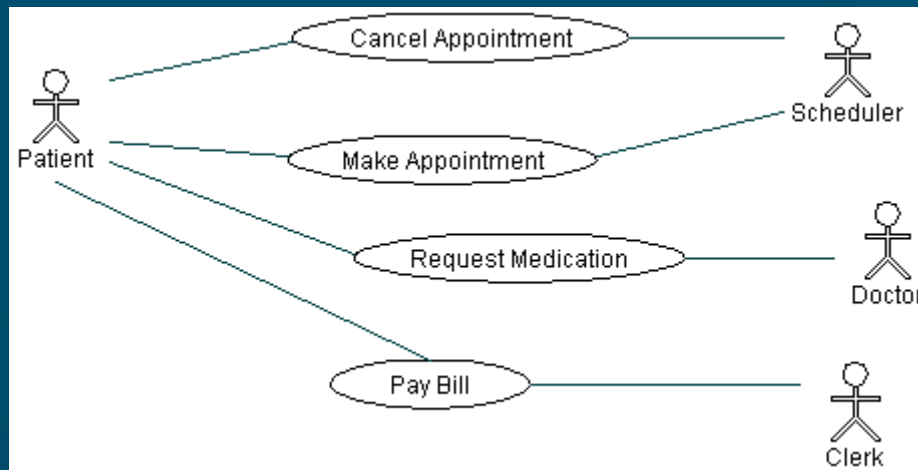
UML: Class diagram

- Each box represents a **class** (type)
 - Name, attributes, methods
- Lines show **relationships** between classes



UML: Use case diagram

- Describes **relationships** between **actors**:
 - ◆ **Patient** calls the clinic to make an appointment
 - ◆ **Receptionist** books timeslot
 - ◆ **Patient** sees **doctor** and requests medication
 - ◆ **Patient** pays bill to **clerk**



- See Borland's UML tutorial for more details

Design patterns

- Commonly used software **designs**
- Not **reinventing** the wheel
 - Similar to **libraries**, but for program design
- Similar to **architectural** elements: **arch**, **column**
- “**Gang of Four**” standard reference (1995):
 - **Gamma, Helm, Johnson, Vlissides**, “Design Patterns: Elements of Reusable OO Software”
 - **Creational** patterns: e.g., **abstract factory**
 - **Structural** patterns: e.g., **proxy**
 - **Behavioural** patterns: e.g., **observer**

Java coding style: HelloWorld.java

```
public class HelloWorld {  
    public static void main( String args[] ) {  
        System.out.println( "Hello, World!" );  
    }  
}
```

- Class names are **nouns** in **CamelCase**
- Method names are usually **verbs** in lowercase:
 - **useLowerCamelCase()** or **use_underscores()**
- Local **variable** names are also **lowercase**
- **Legal** identifiers: **alphanumeric**, **_**, **\$**
 - Cannot start with a **digit**

Selection structure: if, Booleans

- *if (condition) statement;*
- Condition is of type **boolean**
 - Literals: **true**, **false**
 - Binary operators: **==**, **!=**, **<**, **>**, **<=**, **>=**,
 - Boolean operators (**shortcut**): **&&**, **||**
- **Compound** statement using **{}**:

```
if (condition) {  
    statement1;  
    statement2;  
}
```

Selection: if ... else ...

```
if (condition)
    statement1;
else
    statement2;
```

- How to do **elif**?

```
if (condition)
    statement1;
else if (condition2)
    statement2;
```

The “dangling else” problem

```
if (cond1)
    if (cond2)
        statement1;
else
    statement2;
```

- Which **if** is the **else** attached to?
- Solution: always use **braces**

```
if (cond1) {
    if (cond2) {
        statement1;
    }
} else {
    statement2;
```

Java user interfaces

■ Command line

- HelloWorld example (java/ dir)
- System.out.print()
- System.in exists, but reading lines is harder

■ Applets

- Addition example (“Lab0”)
- TextField, Label

■ Swing

- SayHello example (java/ dir)
- JOptionPane

Text output: System.out

- `System` is a class in the `java.lang` library
- `java.lang` is automatically imported
 - Can import other libraries with `import`
- `System.out` is the `standard output` file object
- Its `methods` include `print` and `println`:
 - `System.out.println("Hello!");`
 - `System.out.print("Hello!\n");`
- Other `escape` characters:
 - Tab (`\t`), backslash (`\\`), quote (`\"`)

Java Applets

- **Applets** are small applications designed to be within a webpage
- **GUI** components: text **input** boxes, **buttons**, etc.
- See the “**Lab0**” (Addition) **template lab**
 - ◆ `import java.applet.Applet;`
 - ◆ `public class MyClass extends Applet { ...`

Java Swing

- **Swing** is Java's built-in **GUI** toolkit
- Can build **stand-alone** GUI programs
- See “**SayHello**” example (cmpt167.seanho.com/java)
 - **import** `javax.swing.*`;
 - **Input** dialog: `JOptionPane.showInputDialog()`
 - **Output**: `JOptionPane.showMessageDialog()`
- See Sun's tutorial for more details
- Lab1 can be done in either Swing or an applet

TODO

- Lab1 a due next week Wed 24Jan:
 - Selection structure
 - Swing program: see “SayHello” example, or
 - Java Applet: see “Lab0” (Addition) template
- Lab1 b due Wed 31Jan:
 - Repetition structure