Chapter 13. Object Oriented Programming

Byoung-Tak Zhang
TA: Hanock Kwak
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

http://bi.snu.ac.kr
The history of computer programming is a steady move away from *machine-oriented views* of programming towards concepts and metaphors that more closely reflect the way in which *we ourselves understand the world*.
Computer Programming

- Programming has progressed through:
  - machine code
  - assembly language
  - machine-independent programming languages
  - procedures & functions
  - objects
History of Object-Oriented Programming

- Started out for simulation of complex man-machine systems, but was soon realized that it was suitable for all complex programming projects
- **SIMULA I (1962-65) and Simula 67 (1967)** were the first two object-oriented languages
  - Developed at the Norwegian Computing Center, Oslo, Norway by Ole-Johan Dahl and Kristen Nygaard
  - Simula 67 introduced most of the key concepts of object-oriented programming: objects and classes, subclasses (“inheritance”), virtual procedures
The Ideas Spread

- Alan Kay, Adele Goldberg and colleagues at Xerox PARC extend the ideas of Simula in developing Smalltalk (1970’s)
  - Kay coins the term “object oriented”
  - Smalltalk is first fully object oriented language
  - Grasps that this is a new programming paradigm
  - Integration of graphical user interfaces and interactive program execution

- Bjarne Stroustrup develops C++ (1980’s)
  - Brings object oriented concepts into the C programming language
class Time {

    private int hour, minute;

    public Time (int h, int m) {
        hour = h;
        minute = m;
    }

    public void addMinutes (int m) {
        int totalMinutes =
                        ((60*hour) + minute + m) % (24*60);
        if (totalMinutes<0)
            totalMinutes = totalMinutes + (24*60);
        hour = totalMinutes / 60;
        minute = totalMinutes % 60;
    }
}

Objects
Objects

**Time**

*Attributes:*
- hour = 8
- minute = 30

*Methods:*
- void addMinutes(int m)

**inTime**

*Attributes:*
- hour = 8
- minute = 30

*Methods:*
- void addMinutes(int m)

**outTime**

*Attributes:*
- hour = 17
- minute = 35

*Methods:*
- void addMinutes(int m)
Classes and Objects

- A *class* is a prototype for creating objects
- When we write a program in an object-oriented language like Java, we define classes, which in turn are used to create objects
- A class has a *constructor* for creating objects
A Simple Class, called “Time” (partial)

class Time {
    private int hour, minute;

    public Time (int h, int m) {
        hour = h;
        minute = m;
    }

    public void addMinutes (int m) {
        int totalMinutes =
            ((60*hour) + minute + m) % (24*60);
        if (totalMinutes<0)
            totalMinutes = totalMinutes + (24*60);
        hour = totalMinutes / 60;
        minute = totalMinutes % 60;
    }
}

constructor for Time
An object is a computational entity that:

- **Encapsulates** some state
- Is able to perform actions, or **methods**, on this state
- Communicates with other objects via **message passing**
Encapsulates some state

- **Set of variables**
  - that describe an object’s state
  - These variables are sometimes called an object’s attributes (or fields, or instance variables, or data members, or …)

```java
class Time {
    private int hour, minute;

    // . . .
}
An object can also include a group of **procedures/functions** that carry out actions.

```java
class Time {
    private int hour, minute;

    public Time (int h, int m) {
        hour = h;
        minute = m;
    }

    public void addMinutes (int m) {
        int totalMinutes = ((60*hour) + minute + m) % (24*60);
        if (totalMinutes<0)
            totalMinutes = totalMinutes + (24*60);
        hour = totalMinutes / 60;
        minute = totalMinutes % 60;
    }
}
```
Communicates with other objects via message passing

- Sends messages to objects, triggering methods in those objects

In one of bill's methods, the following code appears:

```java
Time inToWork = new Time(8, 30);
inToWork.addMinutes(15);
```

```java
inToWork.addMinutes(15)
```
Structure of a Class Definition

```cpp
class name {
    declarations

    constructor definition(s)

    method definitions
}
```

- Attributes and symbolic constants
- How to create and initialize objects
- How to manipulate the state of objects
Advantages

- Building the system as a group of interacting objects:
  - Allows extreme modularity between pieces of the system
  - May better match the way we (humans) think about the problem
  - Avoids recoding, increases code-reuse
Inheritance

- Classes can be arranged in a hierarchy
- Subclasses *inherit* attributes and methods from their parent classes
- This allows us to organize classes, and to avoid rewriting code – new classes *extend* old classes, with little extra work!
- Allows for large, structured definitions
Example of Class Inheritance

Objects made from this class, for example, have all the attributes and methods of the classes above them, all the way up the tree.
Polymorphism

- An object has “multiple identities”, based on its class inheritance tree
- It can be used in different ways
- A Circle is-a Shape is-a Object
How Objects are Created

```java
Circle c = new Circle();
```
Using Polymorphism in Arrays

- We can declare an array to be filled with “Shape” objects, then put in Rectangles, Circles, or Triangles.

```java
firstShape
Attributes:
  length = 17
  width = 35
Methods:
  int computeArea()

secondShape
Attributes:
  radius = 11
Methods:
  int computeArea()

thirdShape
Attributes:
  base = 15
  height = 7
Methods:
  int computeArea()
```

samples (an array of Shape objects)
Polymorphism give us a powerful way of writing code that can handle multiple types of objects, in a unified way

```java
public int calculatePaint (Shape myFigure) {
    final int PRICE = 5;
    int totalCost = PRICE * myFigure.computeArea();
    return totalCost;
}
```
Object-Oriented Programming in Industry

- Large projects are routinely programmed using object-oriented languages nowadays
- MS-Windows and applications in MS-Office – all developed using object-oriented languages
- This is the world into which our students are graduating…