1. Introduction to Object-Oriented Programming

6 Aug 2015

Objectives

- Object-oriented programming concept
- Classes and objects
- Class and instance methods
- Class and instance data values
- Class and object diagrams
- Inheritance
- Software life cycle

Programming Paradigm

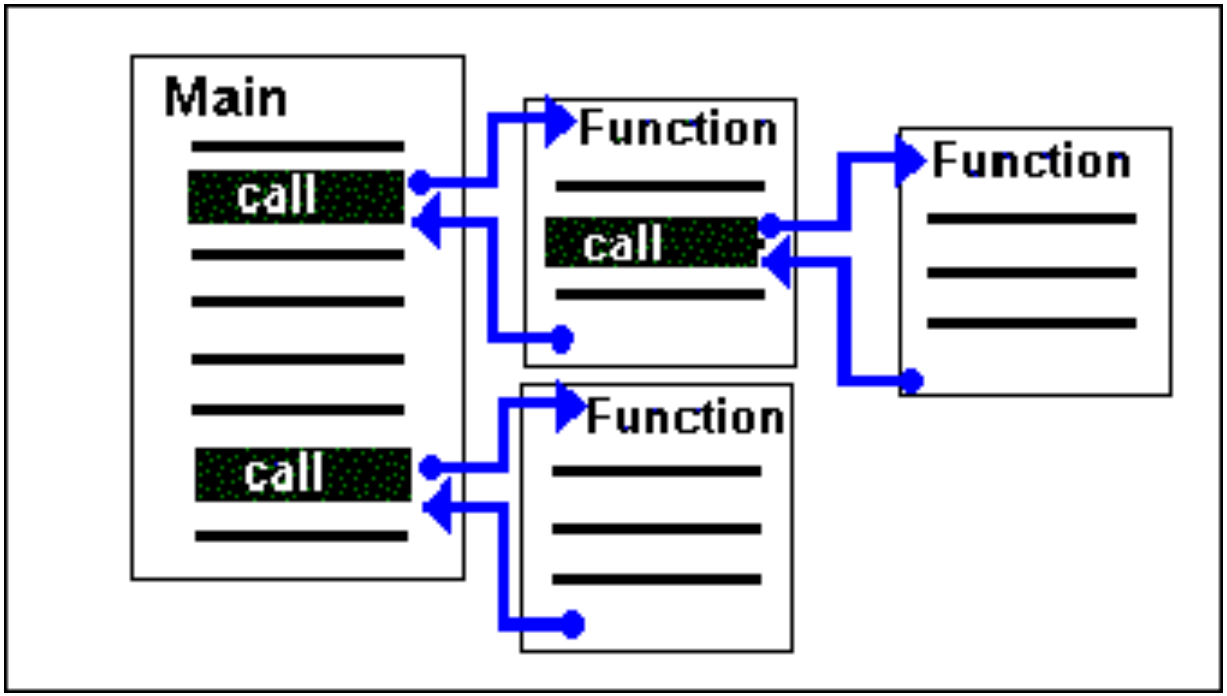
- Functional Programming (or Structured/ Procedural Programming)
  - Ex. Pascal, Fortran, C

- Object-Oriented Programming
  - Ex. C++, Java, C#, Objective-C

Functional Programming

- Functional Programming:
  - A list of instructions telling a computer, step-by-step, what to do, usually having a linear order of execution from first to last statement
- There are 2 parts
  - Main Program
  - Functions
- Ex. Pascal, Fortran, C

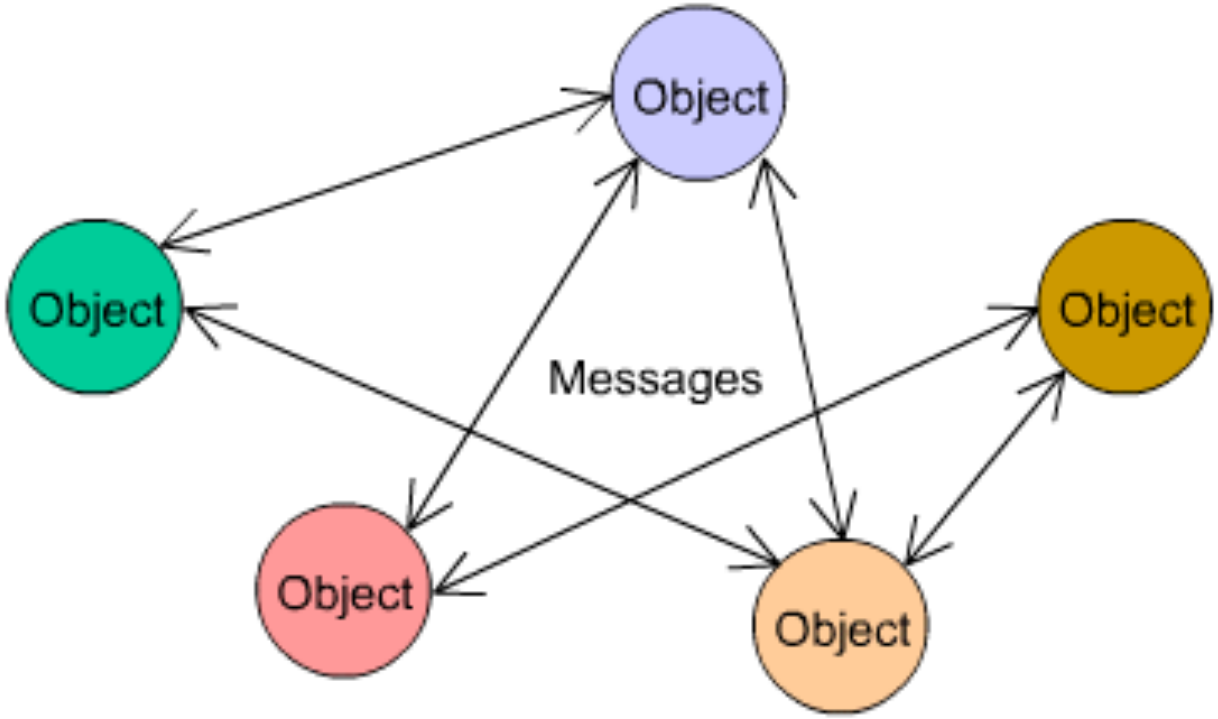
Functional Programming



Object-Oriented Programming

- Object-Oriented Programming:
  - A programming paradigm based on the concept of "objects", that contain data, often known as attributes; and code, often known as methods. In OO programming, computer programs are designed by making them out of objects that interact with one another.
- Ex. C++, Java, C#, Objective-C

Object-Oriented Programming



Interaction of objects via message passing

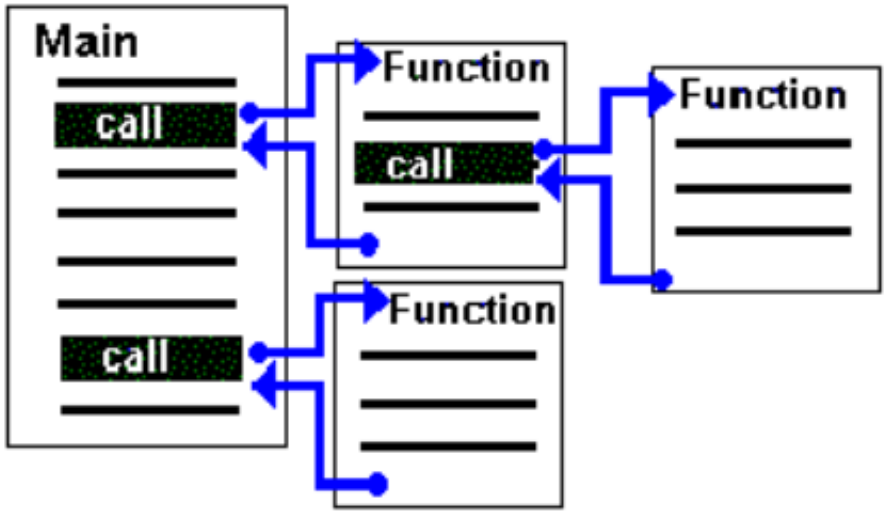
Functional vs OOP

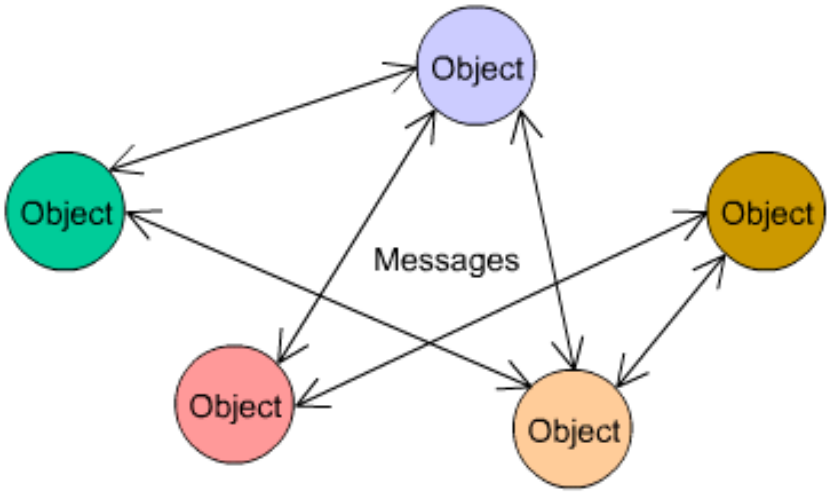
Functional	Object-Oriented
• Divided into sub-modules or functions	• Organized by classes and objects
• Function call is used	• Message passing is used
• Good for small program	• Good for complex program

Functional vs OOP

Functional	Object-Oriented
• Top-down approach	• Bottom-up approach
• Less code reusability	• More reusability
• Hard to maintain	• Easier to maintain

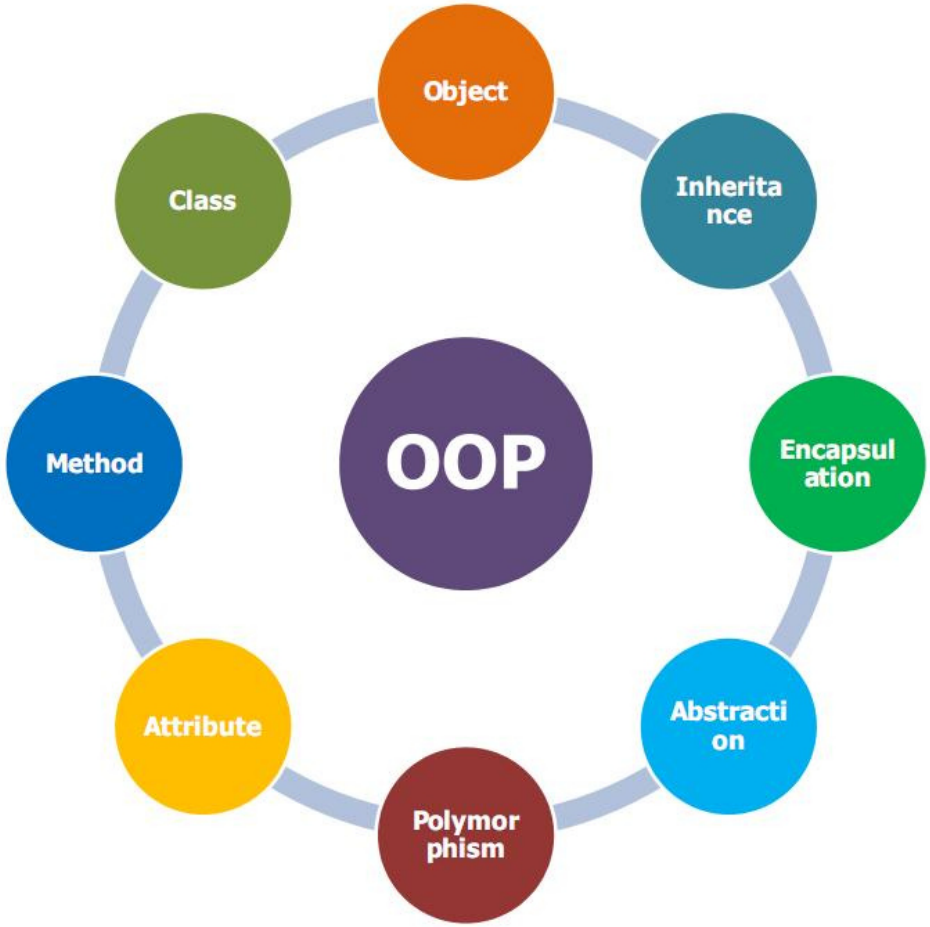
Functional vs OOP





Interaction of objects via message passing

Object-Oriented Programming



Object

- Object:
  - A thing, both tangible and intangible.
  - Instance of a class

Class

- Class:
  - A kind of mold or template that dictates what objects can do or cannot do.
  - A new complex data type, defines data and behavior of objects.

- Class contains:
  - Data (called attribute, property or data member)
  - Code (called method or behavior)

Class and Object

- Ex. Student class

Student.java - Notepad

File Edit Format View Help

```
public class Student {
    //data members------------------//
    private int id;
    private String name;


    //methods-----------------------//
    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public String getStudentInfo() {
        return name + " (id: " + id + ")";
    }
}
```

Class and Object

- Ex. Student object

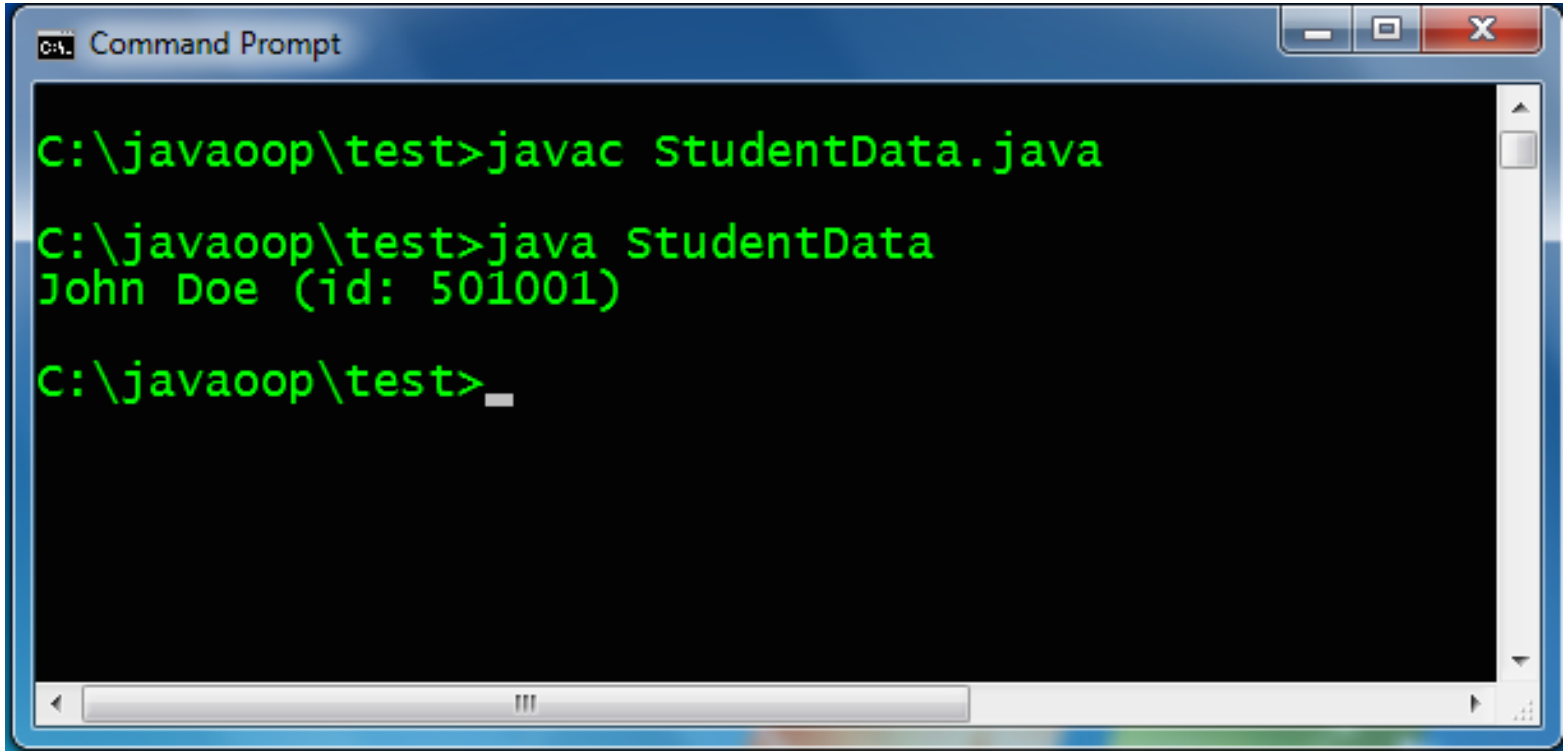
StudentData.java - Notepad

File Edit Format View Help

```
public class StudentData {

    public static void main(String[] args) {

        Student s1 = new Student(501001, "John Doe");

        System.out.println(s1.getStudentInfo());

    }

}
```

Class and Object

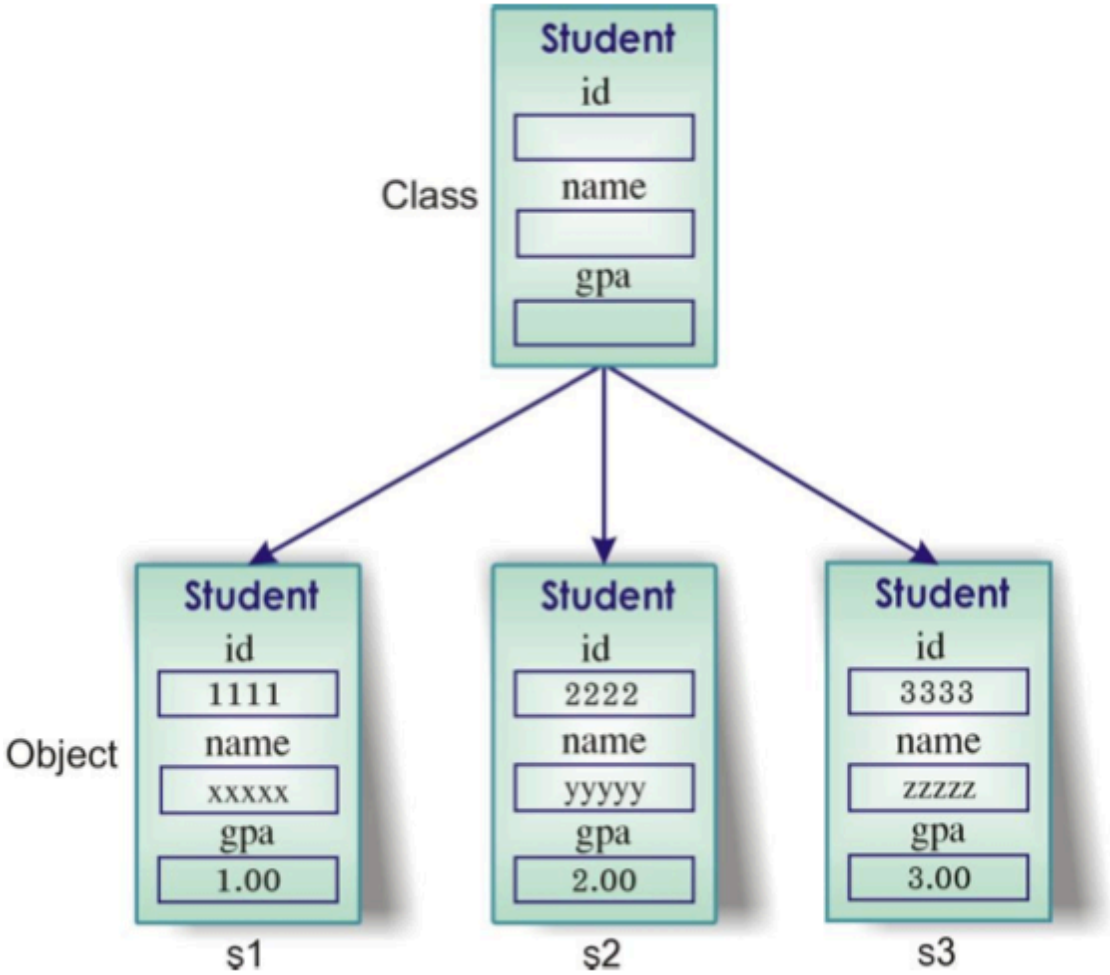
- Output



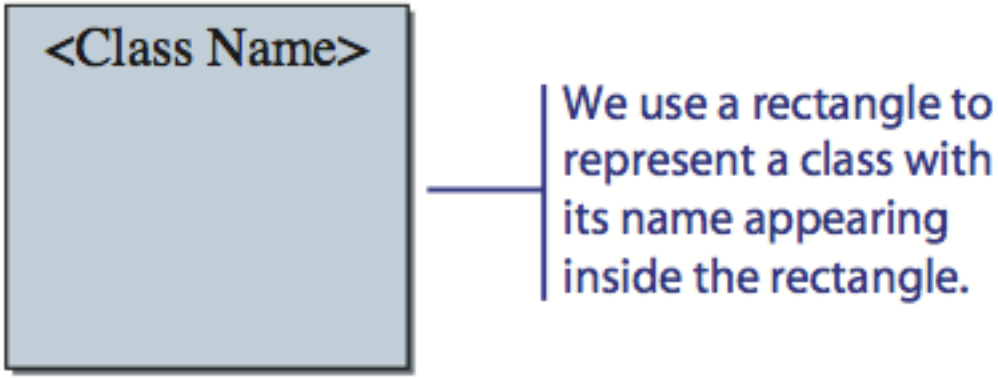
Class and Object

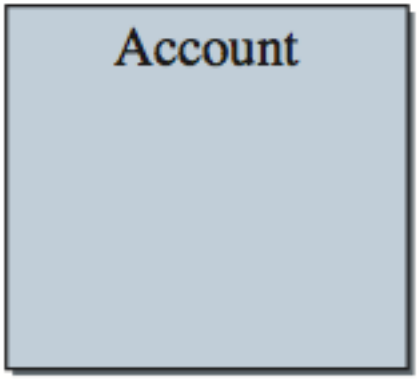
Class	Object
• Bank Account	• John's Bank Account
• Room	• Room #204
• Student	• Student #45010
• Subject	• English 101

Class and Object



Representation of a class





Representation of a class

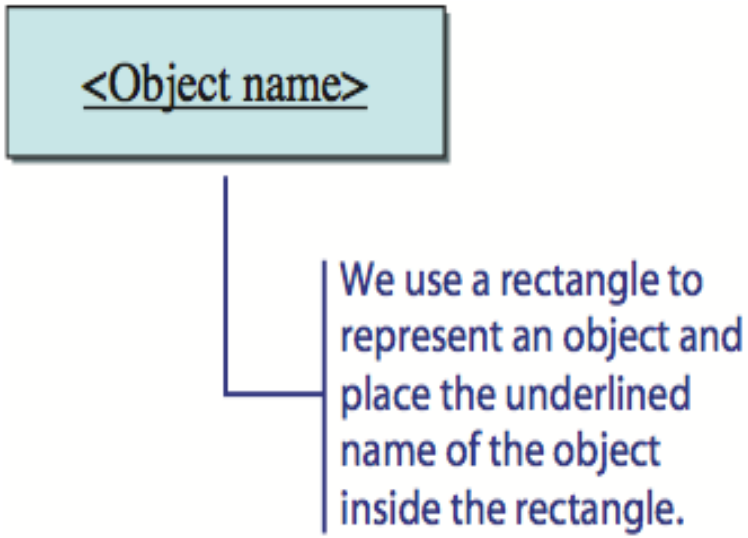
Student
name grade
getName() printGrade()

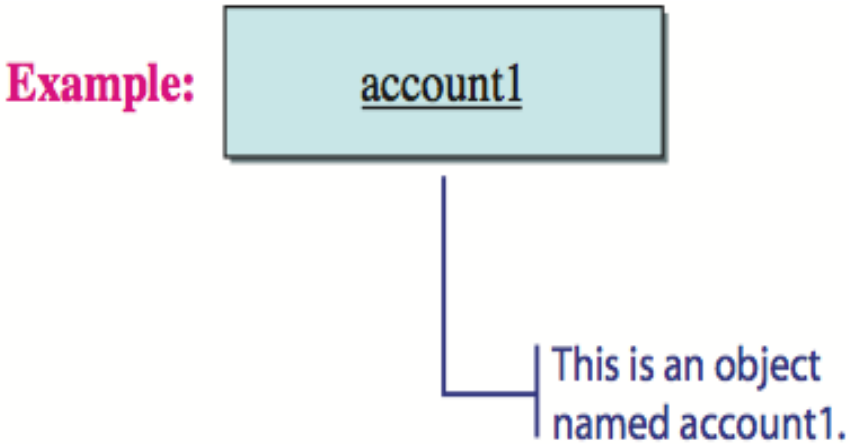
Circle
radius color
getRadius() getArea()

SoccerPlayer
name number xLocation yLocation
run() jump() kickBall()

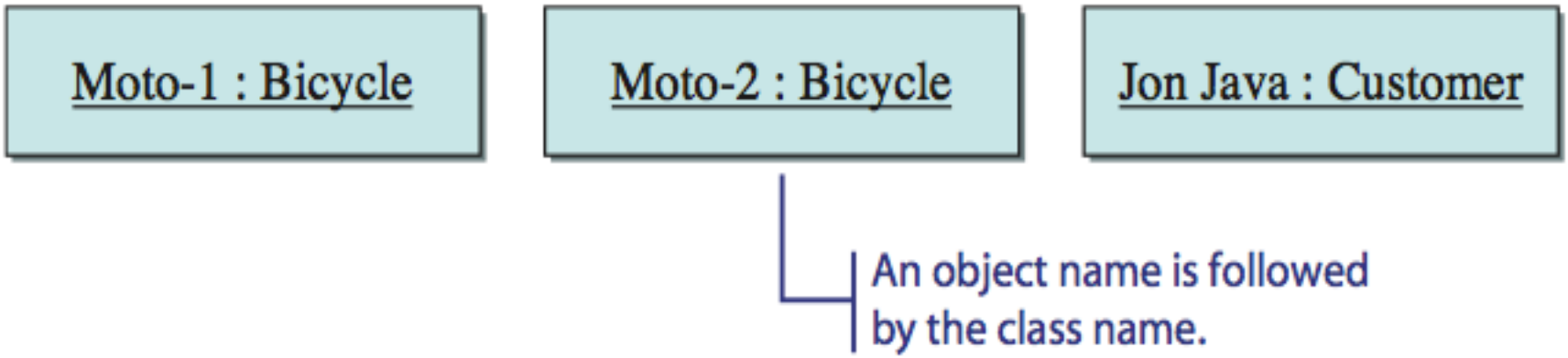
Car
plateNumber xLocation yLocation speed
move() park() accelerate()

Representation of an object

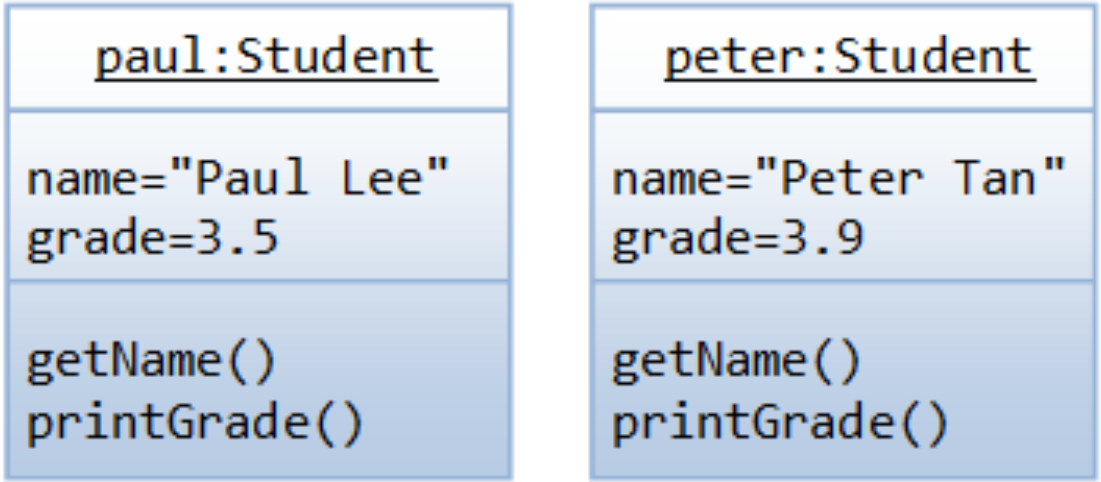




Representation of an object



Representation of an object



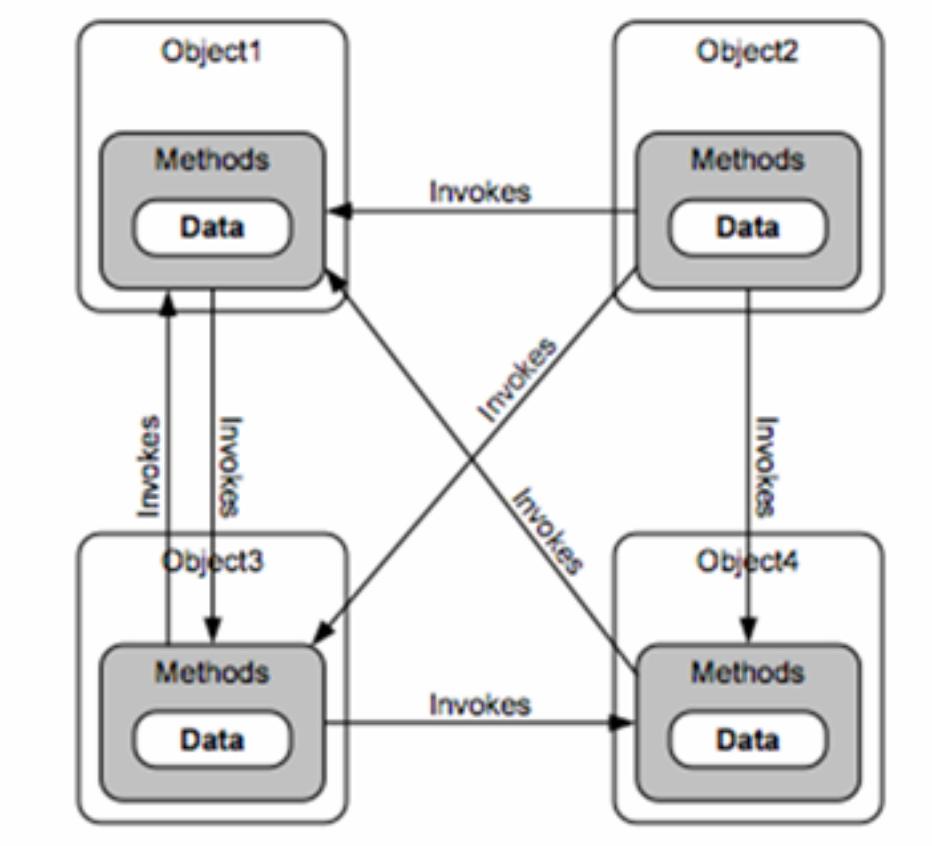
Messages and Methods

- Message:
  - A request to an object to invoke one of its methods. A message therefore contains
    - the name of the method
    - the arguments of the method.
- Ex. Customer object send a message deposit to an Account object to deposit $100.

Messages and Methods

- Method:
  - A sequence of instructions that a class or an object follows to perform a task.
- Type of method
  - Class method: A method defined for a class
  - Instance method: A method defined for an object

Messages and Methods



Messages and Methods

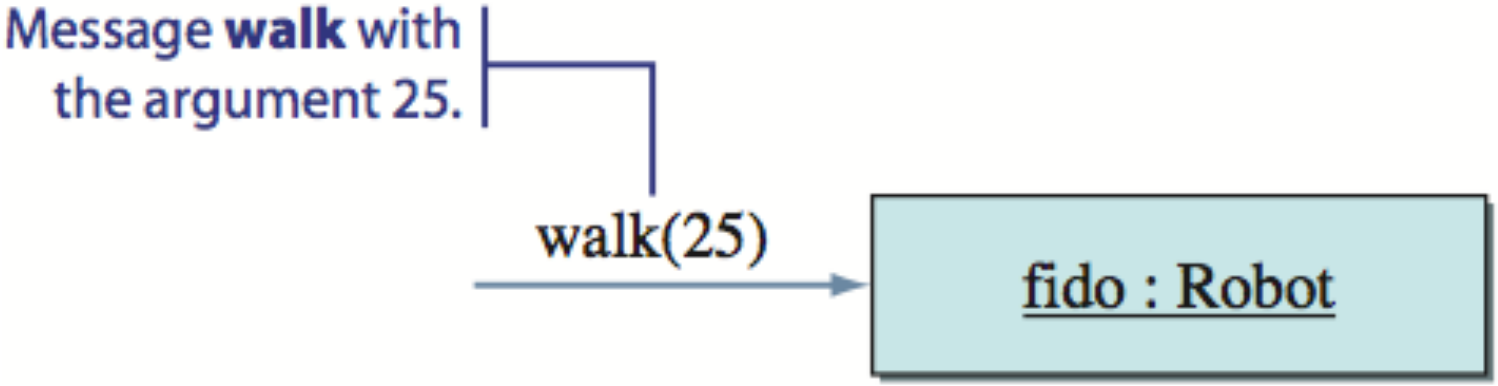


Figure 1.4 Sending the message **walk** to a **Robot** object.

Messages and Methods

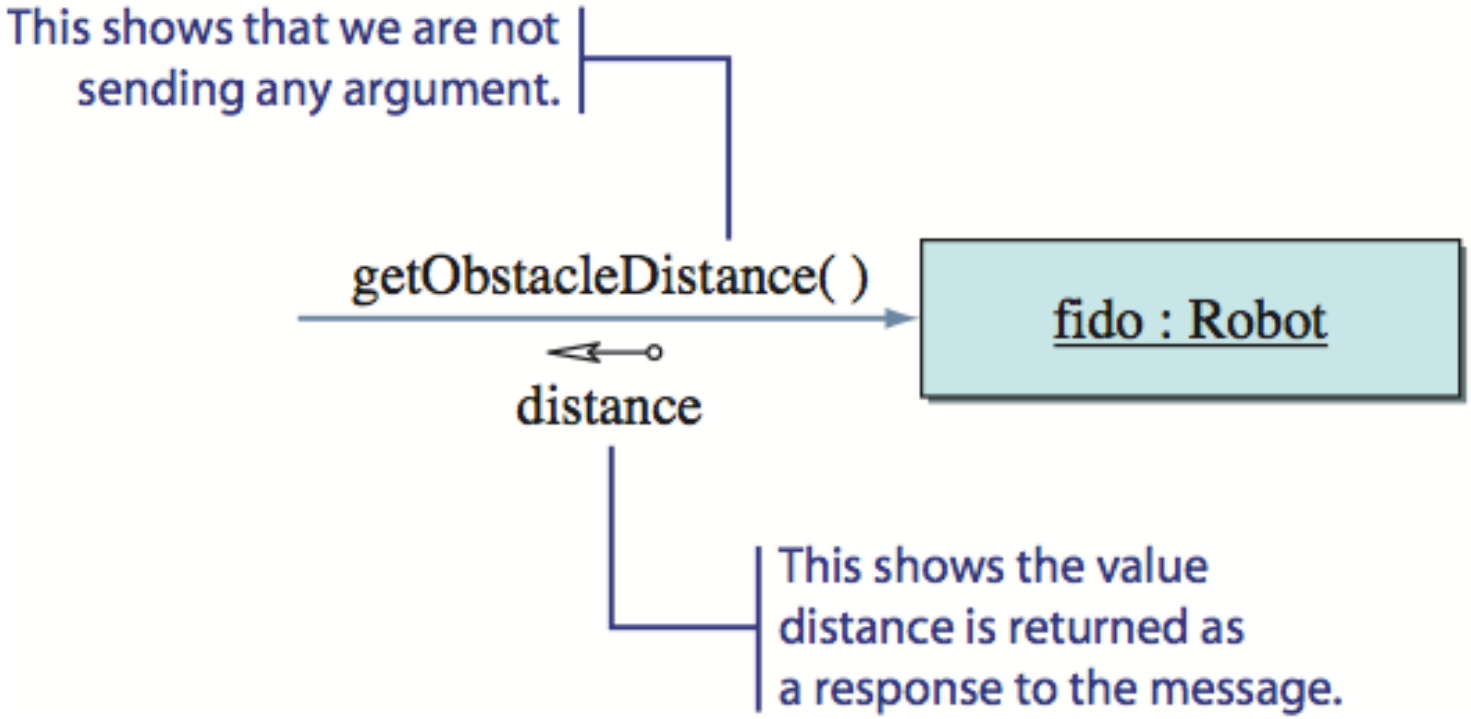


Figure 1.5 The result **distance** is returned to the sender of the message.

Messages and Methods

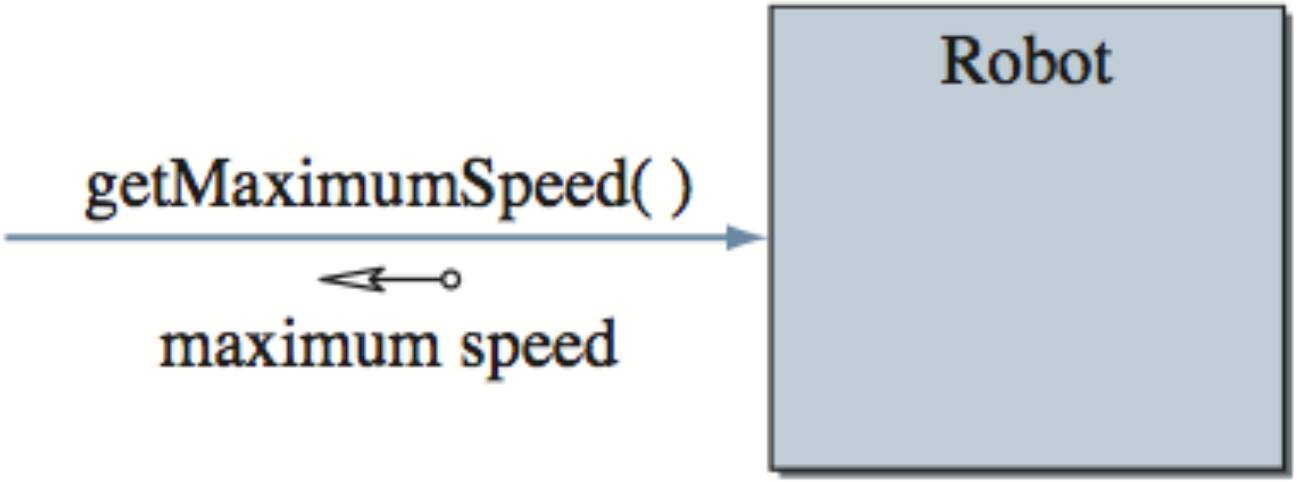


Figure 1.6 The maximum possible speed of all **Robot** objects is returned by the class method **getMaximumSpeed.**

Class and Instance Data Values

- Instance data value: is used to represent information for each instance

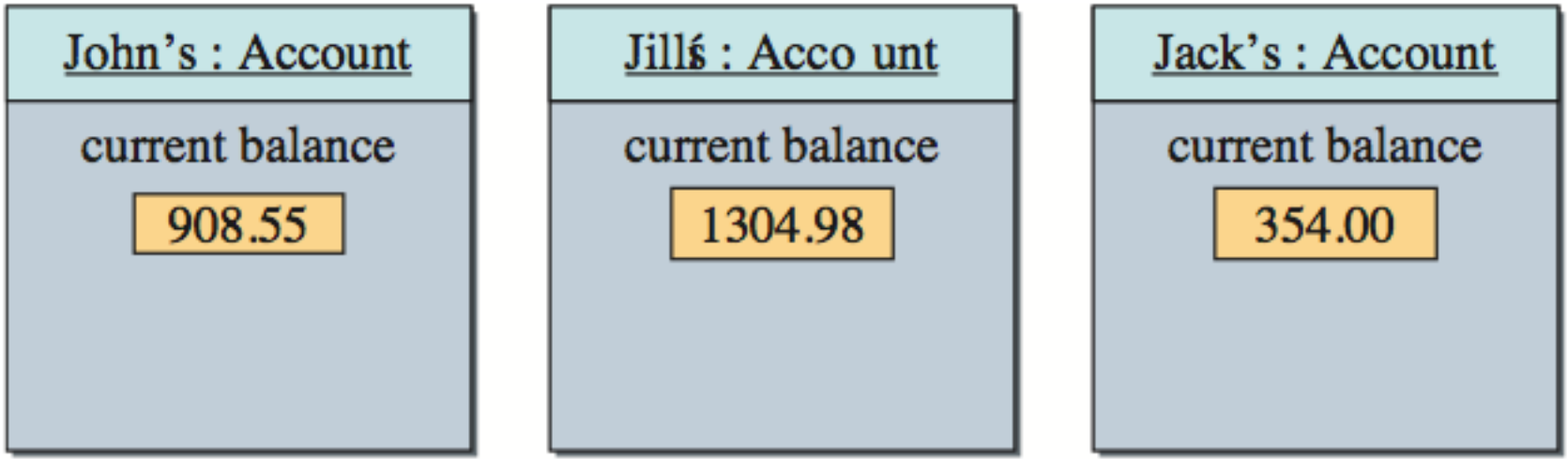


Figure 1.7 Three **Account** objects possess the same data value **current balance,** but the actual dollar amounts differ.

Class and Instance Data Values

- Class data value: is used to represent information shared by all instances instance

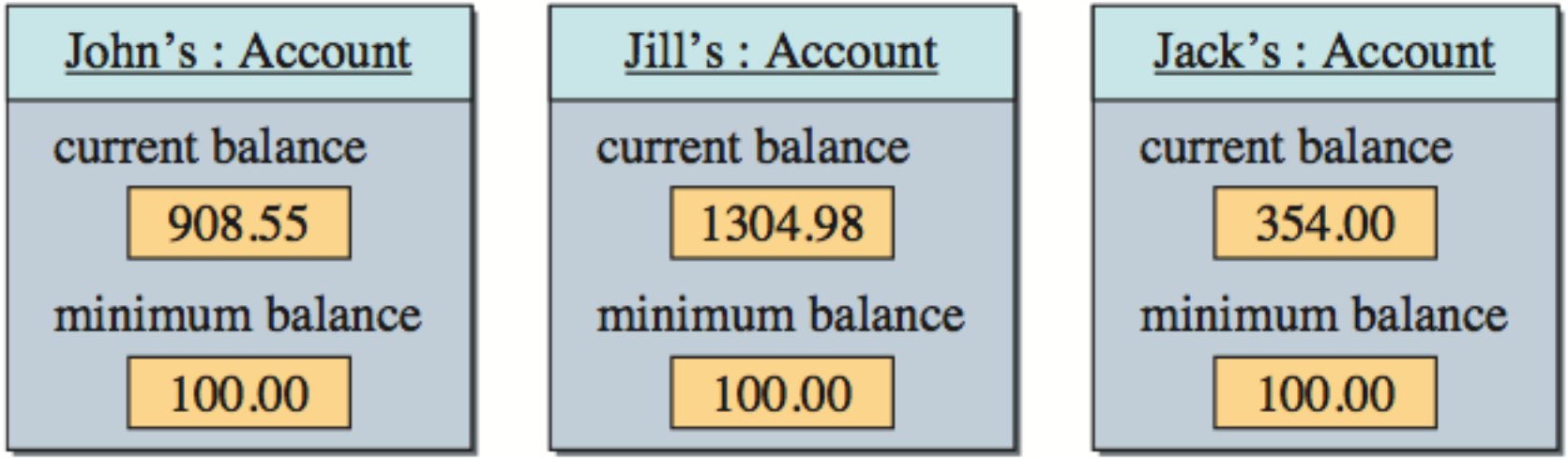


Figure 1.9 Three **Account** objects duplicating information (**minimum balance** = $100) in instance data values.

Class and Instance Data Values

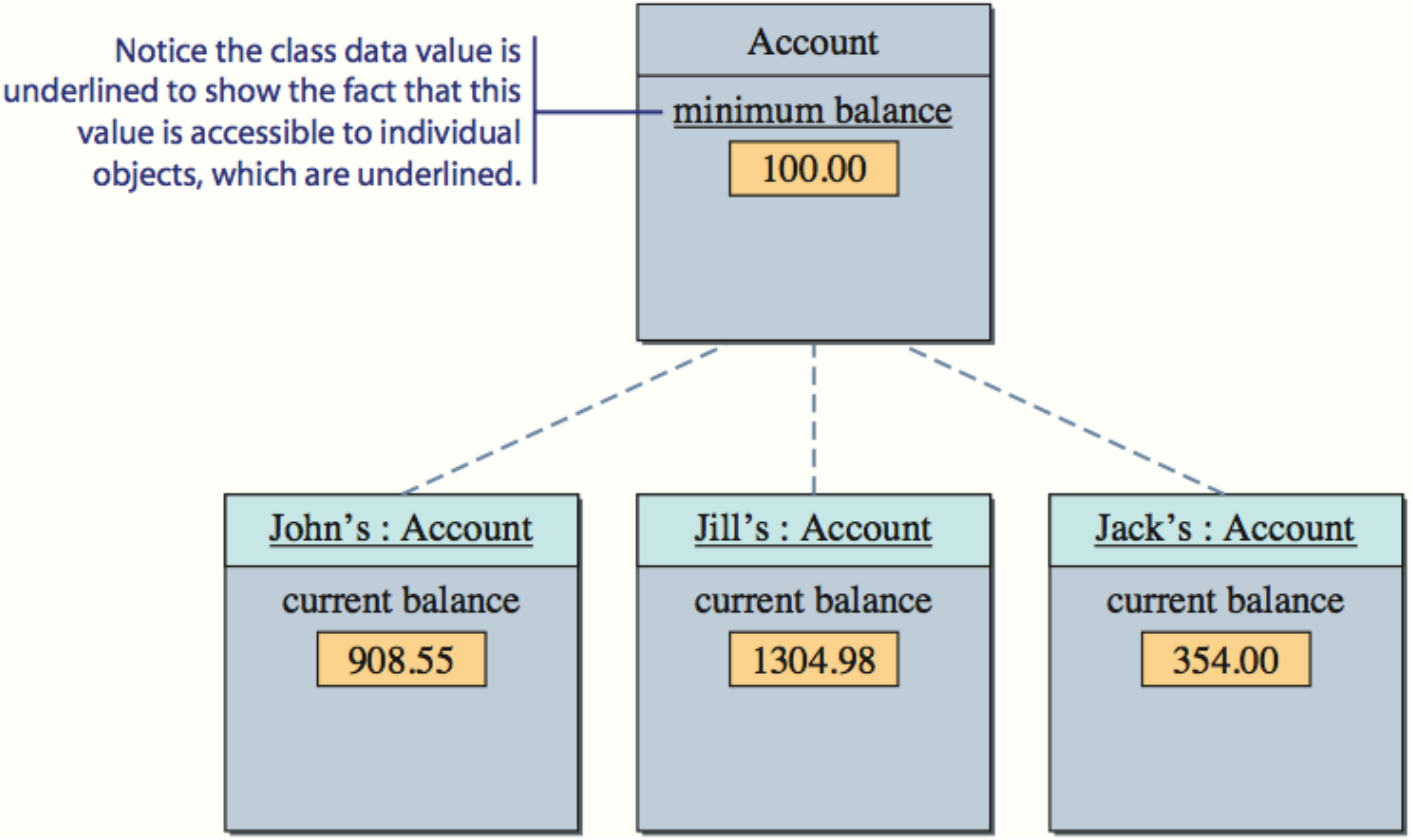


Figure 1.8 Three **Account** objects sharing information (**minimum balance** = $100) stored as a class data value.

Class and Instance Data Values

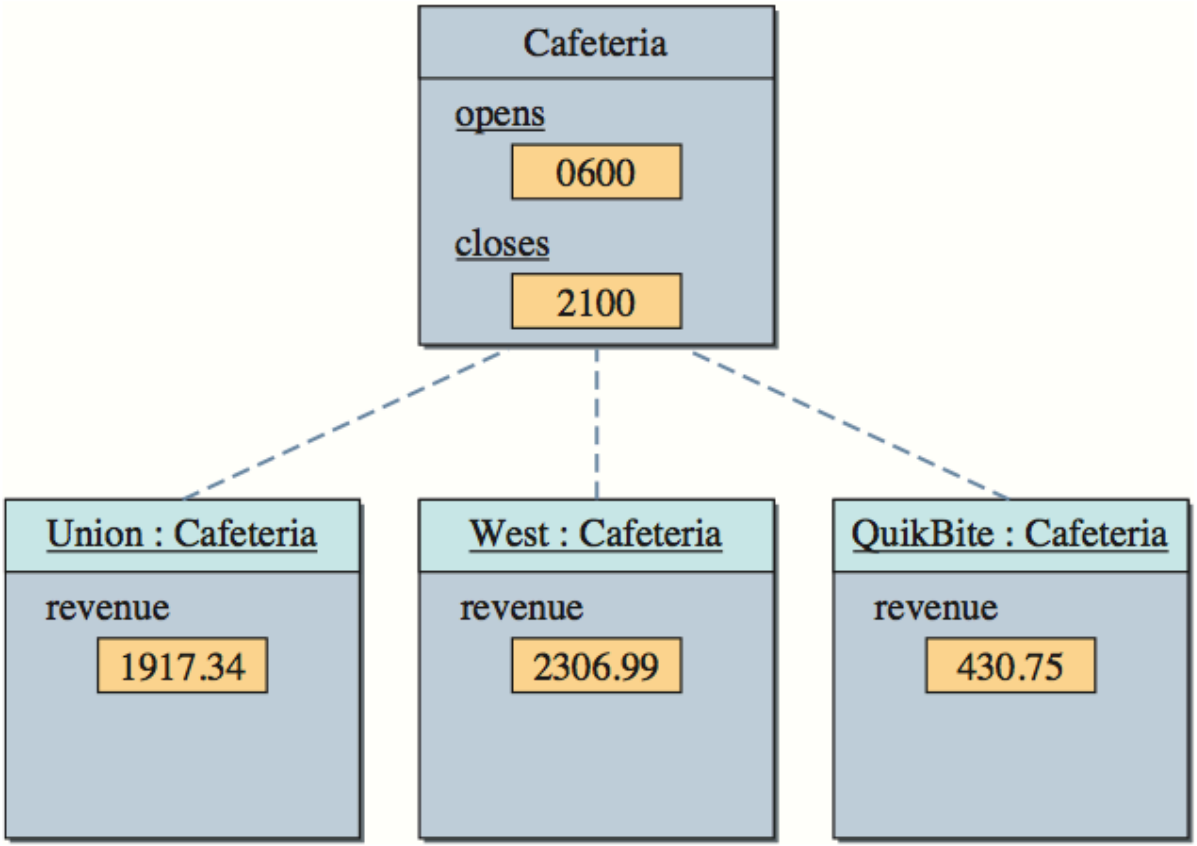


Figure 1.10 Three **Cafeteria** objects sharing the same opening and closing times, stored as class data values.

Class and Instance Data Values

- 2 types of data values
  - Variable: A data value that can change.
  - Constant: A data value that cannot change

Class and Instance Data Values

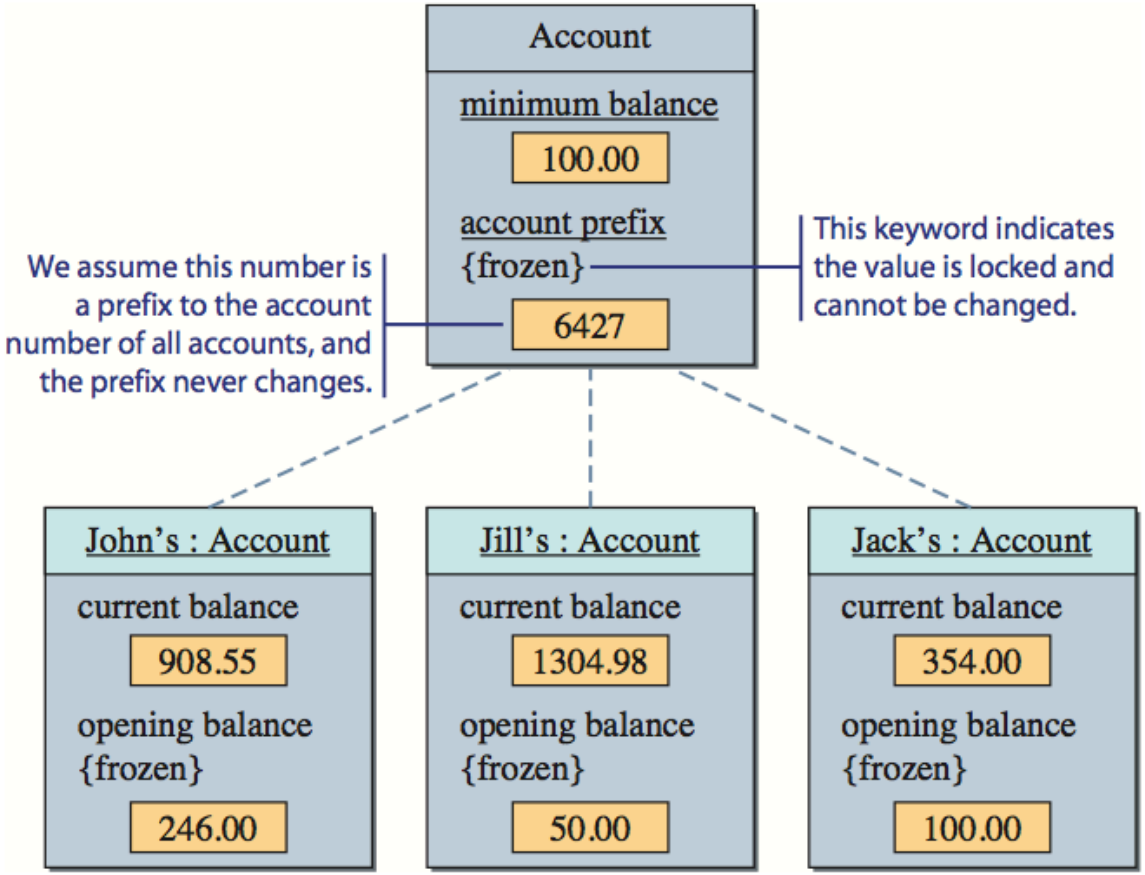


Figure 1.11 Graphical representations for four types of data values: class variable, class constant, instance variable, and instance constant.

Inheritance

- Inheritance:
  - The transfer of characteristics of class to other classes derived from it

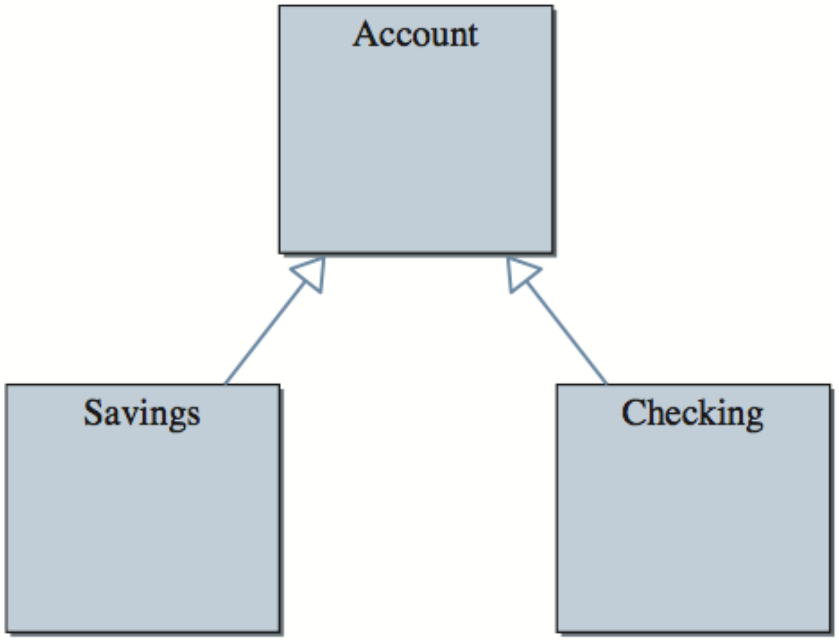


Figure 1.12 A superclass **Account** and its subclasses **Savings** and **Checking.**

Inheritance

- Superclass:
  - A class that has been extended by another class. It allows the extending class to inherit its state and behaviors.
  - Called superclass, base class or parent class.

Inheritance

- Subclass:
  - A class that extends another class. The subclass inherits the state and behaviors of the class it extends.
  - Called subclass, derived class or child class.

Inheritance

- Inheritance is very powerful, and if it is used properly, we can develop complex programs very efficiently and elegantly.
- Inheritance is not limited to one level. A subclass can be a superclass of other classes, forming an inheritance hierarchy.

Inheritance

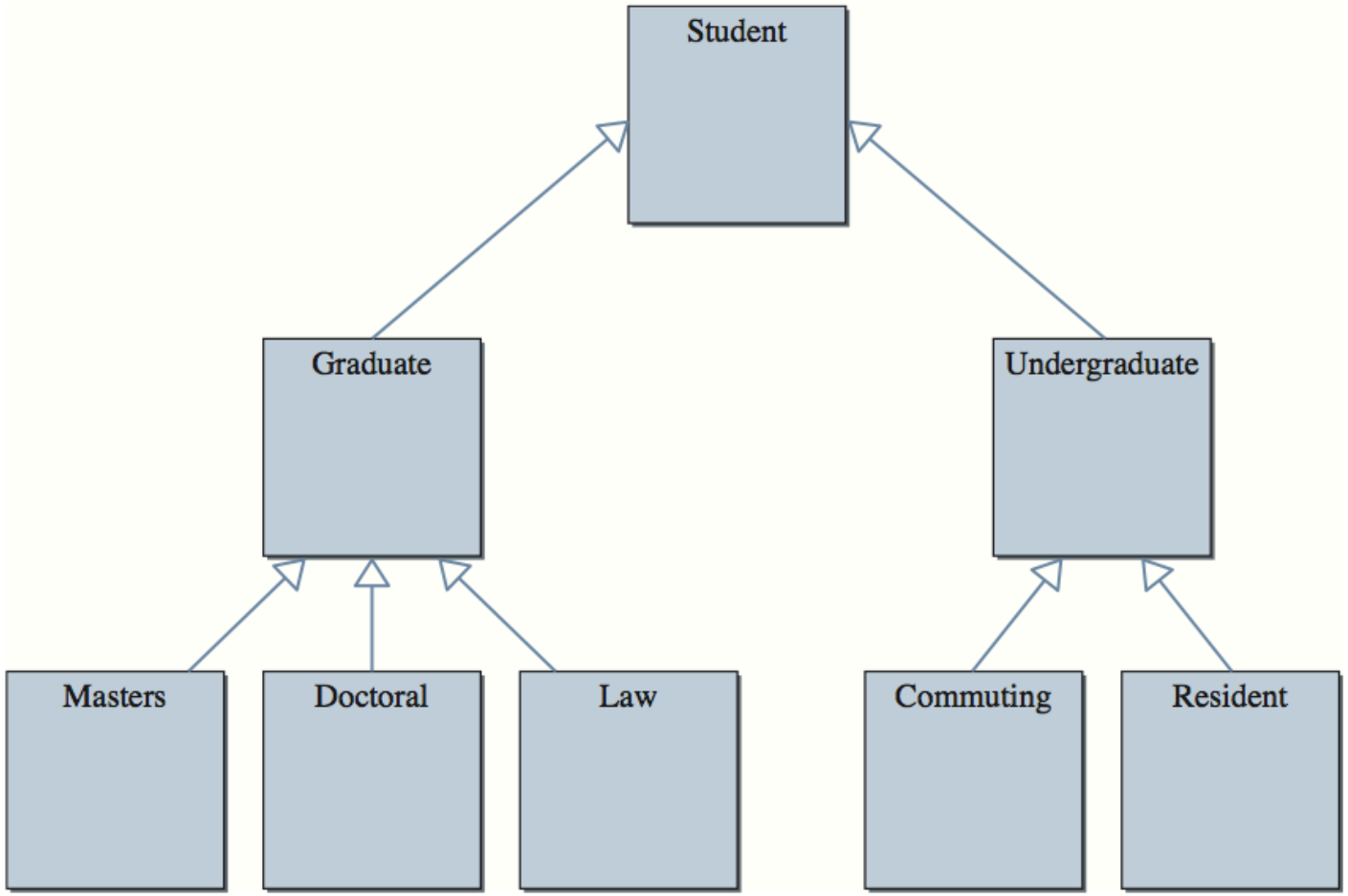


Figure 1.13 An example of inheritance hierarchy among different types of students.

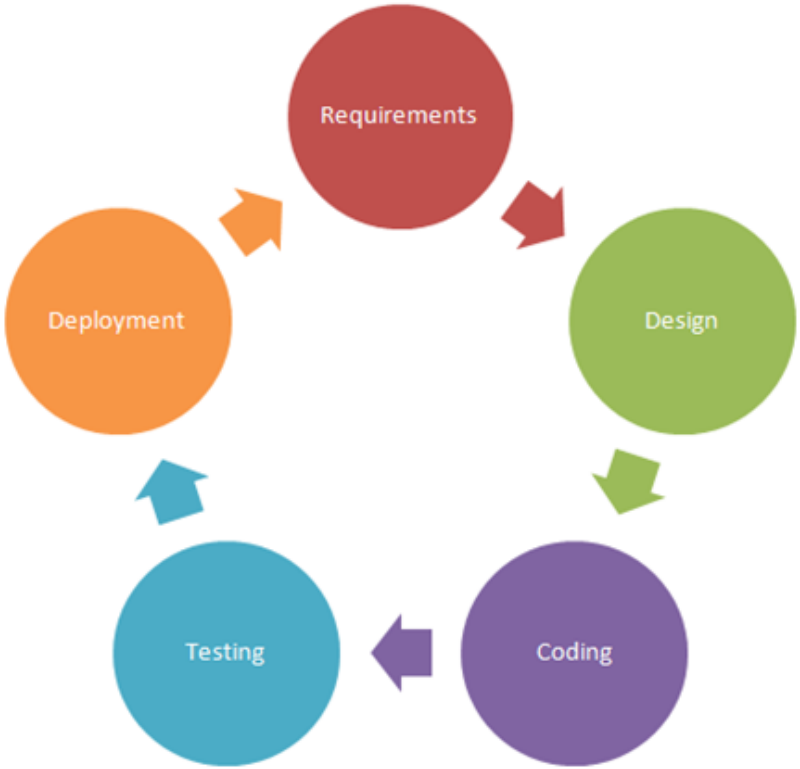
Software Engineering

- Software Engineering:
  - The application of a systematic and disciplined approach to the development, testing, and maintenance of a program.
  - การนำเอาแนวทางที่เป็นระบบระเบียบมาใช้ในการพัฒนา ทดสอบ และบำรุงรักษาโปรแกรม

Software Life Cycle

- There are five major phases in the software life cycle:
  1. Analysis
  2. Design
  3. Coding (Development)
  4. Testing
  5. Operation (Deployment)

Software Life Cycle



Sample SDLC Process

Software Life Cycle

1. Analysis
   - Output: Requirement specification (Software spec.)
2. Design
   - Output: Class diagram
3. Coding (Development)
   - Output: Source code
4. Testing
   - Output: Error
5. Operation (Deployment)
   - Output: Software

Summary

- Object-oriented programming is a programming paradigm based on the concept of objects.
- Class is a template for defining objects.
- Object is a thing, both tangible and intangible.
- There are class and instance methods. We can send messages to objects and classes if they possess matching methods.

Summary

- There are class and instance data values. Data values are also called data members.
- Inheritance is a powerful mechanism to model two or more entities that are different but share common features.
- Five major phases of the software life cycle are analysis, design, coding, testing, and operation.

Reference

- C. Thomas Wu, An Introduction to Object-Oriented Programming with Java, 5th Edition
  - Chapter 1: Introduction to Object-Oriented Programming and Software Development

Question?