CISC 3115 TY2

Object-Oriented Thinking

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Discussed
  - How to define classes, create objects, and use objects
- Concepts of two programming paradigms
  - Procedural and Object-Oriented
- Design classes for problem solving
  - Think in terms of class

Procedural Approach

- "The proposed solution is decomposed by breaking it into a sequence of tasks. These tasks form the basic building blocks for a procedural application." [Korson & McGrego, 1990]
  - First, the designer needs to identify a possible solution.
  - The solution is decomposed into logical modules and submodules, e.g., methods (that we further decompose into steps with sequences of actions, loops, and branches)
  - The execution of tasks can be easily traced from start to finish once the solution is implemented.
- Task-centric

Object-Oriented Approach

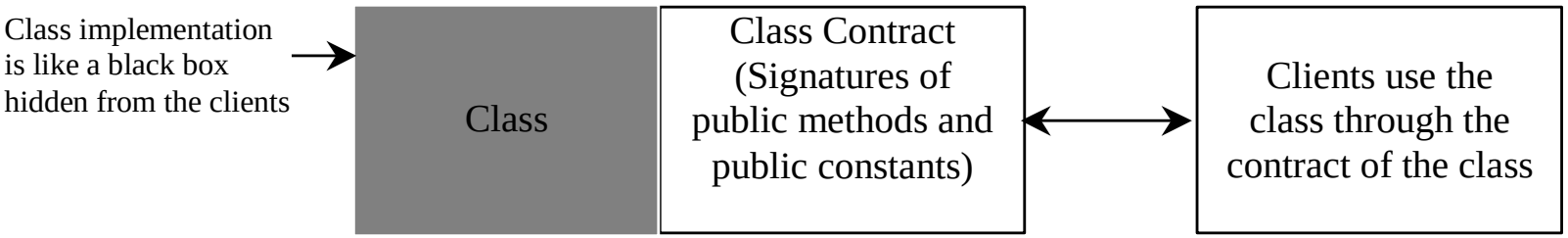
- “The object oriented paradigm assumes a modeling point of view. The model is constructed by viewing the problem domain as a set of interacting entities and the relationships between them." [Korson & McGrego, 1990]
  - First, the designer needs to analyze the problem, and decompose the problem domain into interacting entities and their relationships.
  - Entities are modeled by designing classes to represent them and then using these classes to instantiate objects.
- Data-centric, embodies procedural approach
  - Couples data and methods into objects
    - What are we dealing with?
    - What can be done with it?
  - Designs methods
    - How should it be done?

Class Abstraction

- Separate class implementation from the use of the class.
- Focus on “reuse”. Two roles, creators and users.
- The creator of the class provides a description of the class and let the user know how the class can be used.
- The user of the class does not need to know how the class is implemented.
- To use the class, the users need to know,
  - What are we dealing with?
  - What can be done with it?
- To implement the class, the creators need to know additionally,
  - How is it done?

Encapsulation

- The detail of implementation of a class is encapsulated and hidden from the user.
- The user only needs to know the class contract
  - Class contract: signatures of accessible methods and constants (what are we dealing with? what can be done with it?)
  - The user's classes/objects (clients) interact with it via its contract. (invoking method ≡ passing messages)

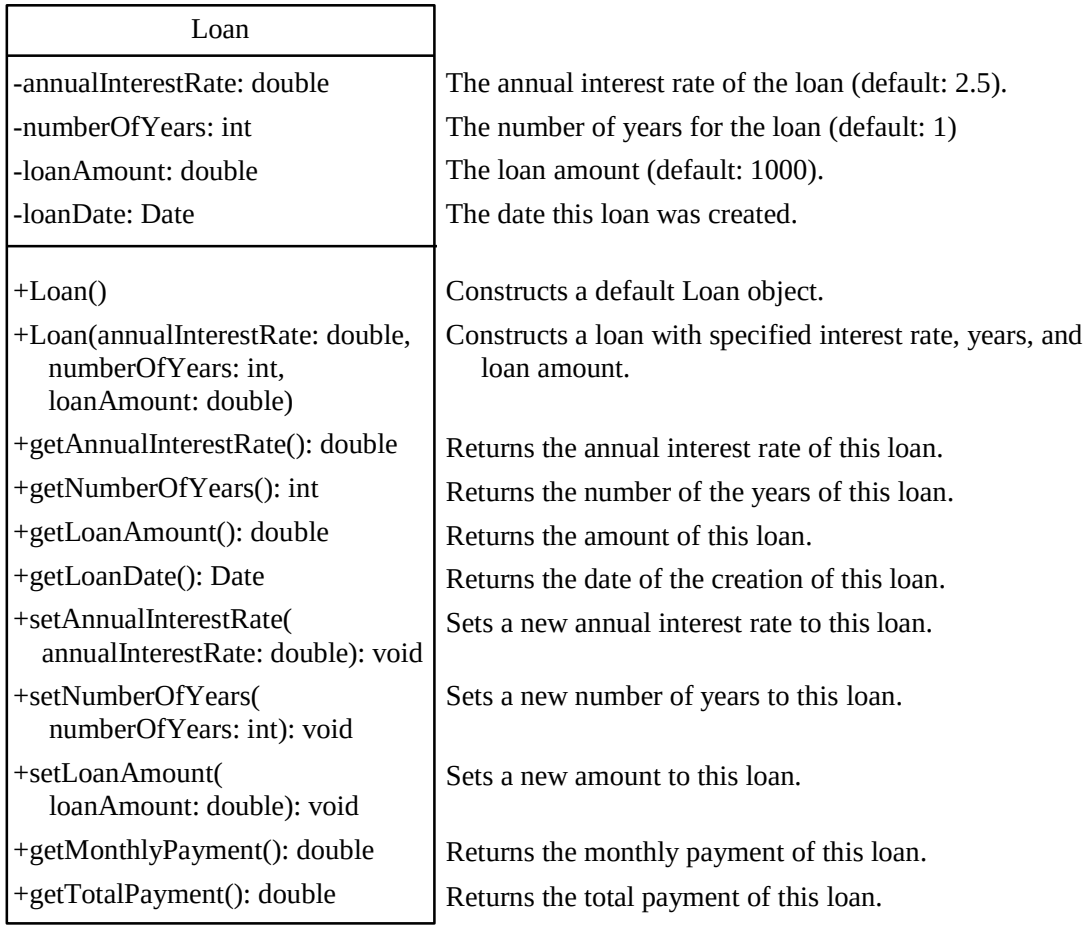


Designing the Loan Class

- The designer's first concern is the class's contract

The Design of The Loan Class

- "-"
  - Private
- "+"
  - public



Data: what are we dealing with?

Methods: what can be done with it?

As the User …

- Can you develop a client class to use the Loan class?
- Example
  - Problem: A borrower enters the information about the loan, i.e., annual interest rate, term (or the number of years), the amount borrowed (or the principal), what will be the monthly payment and the total payment?
  - Solution: Develop a TestLoan.java program as a client to the Loan class (without knowing the implementation detail of the Loan class).
  - Can you do it?

TestLoan.java

TestLoan.java

```java
import java.util.Scanner;

public class TestLoan {
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.println("Enter annual percentage rate of the interest, e.g., 8.25: ");
      double apr = sc.nextDouble();

      System.out.println("Enter the term of the loan (the number of years as an integer): ");
      int term = sc.nextInt();

      System.out.println("Enter the principal of the loan (the amount of borrowed): ");
      double principal = sc.nextDouble();

      Loan loan = new Loan(apr, term, principal);

      System.out.printf("This loan was created on %s\n" +
        "The monthly payment is %.2f\n" +
        "The total payment is %.2f\n",
        loan.getLoanDate().toString(),
        loan.getMonthlyPayment(),
        loan.getTotalPayment()
      );
  }
}

```

As the Creator …

- First, analyze and design
  - What should be the data fields?
  - What should be the public methods? How should client classes/objects interact with the object of this class?
  - In order to be useful in a wide range of applications, the class should provide a variety of ways for clients to interact with (e.g., public constructors and methods)
- Second, design and implement
  - How should the public methods be implemented?

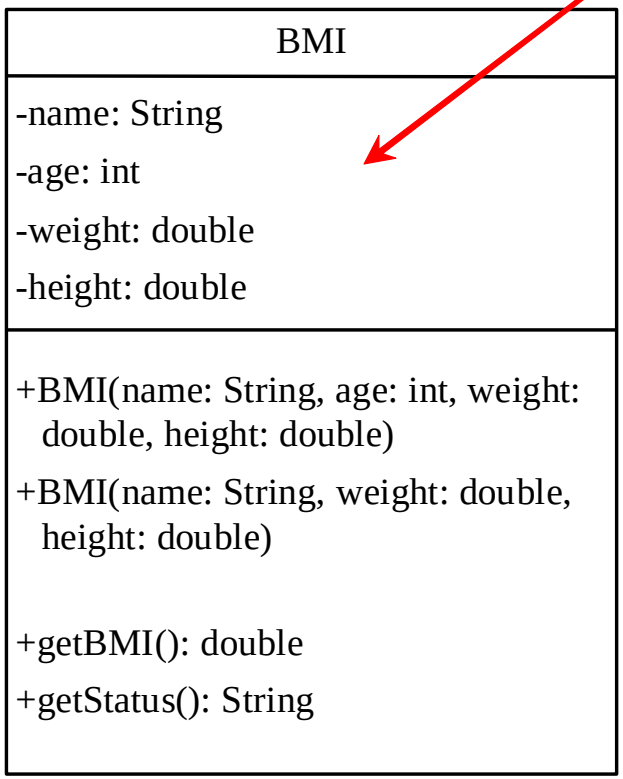
Questions?

- A comparison of procedural and object-oriented approach
- Class abstraction and encapsulation
- Develop client classes

The BMI Class

- Procedural approach
  - Section 3.8 in the textbook
  - ComputeAndInterpretBMI.java
- Analyze and redesign with the Object-Oriented approach
  - Section 10.3 in the textbook
  - BMI.java
  - BMIClient.java

The BMI Class



The get methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The name of the person.

The age of the person.

The weight of the person in pounds.

The height of the person in inches.

Creates a BMI object with the specified name, age, weight, and height.

Creates a BMI object with the specified name, weight, height, and a default age 20.

Returns the BMI

Returns the BMI status (e.g., normal, overweight, etc.)

Questions?

- Procedural approach
  - Focusing on designing methods
- Object-Oriented approach
  - Coupling data and methods together into objects
    - Data: what data are we dealing with?
    - Operations: what can be done to/with the data?
    - Additionally, how should the operations be done? (procedural paradigm)
  - Combining the power of the procedural paradigm with an added dimension that integrates data with operations into objects
  - Data and operation in objects
    - Mirroring the real world

Exercise

- Section 3.9 of the textbook has a program that computes the U.S. federal income tax. The program is designed using the procedural approach.
- In this exercise, you shall use the Object-Oriented approach to design a FederalIncomeTax class, and use the class in a client class called FederalIncomeTaxClient.
  - Examine the ComputeTax.java from Section 3.9, and design the FederalIncomeTax class with care.
  - Your work should include two files, FederalIncomeTax.java and FederalIncomeTaxClient.java