Northeastern University

Lecture 2: Java Review - 2

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Materials are edited by Prof. Jones Yu from

Data Structures and Abstractions with Java, 5th edition. By Frank M. Carrano and Timothy M. Henry.
ISBN-13 978-0-13-483169-5

Outline

- Abstraction
- Specifying Methods

Abstraction

Abstraction

- Focus on *what* instead of *how*
  - » What needs to be done?
  - » For the moment ignore how it will be done.
- Divide class into two parts
  - » Client interface
  - » Implementation

Abstraction

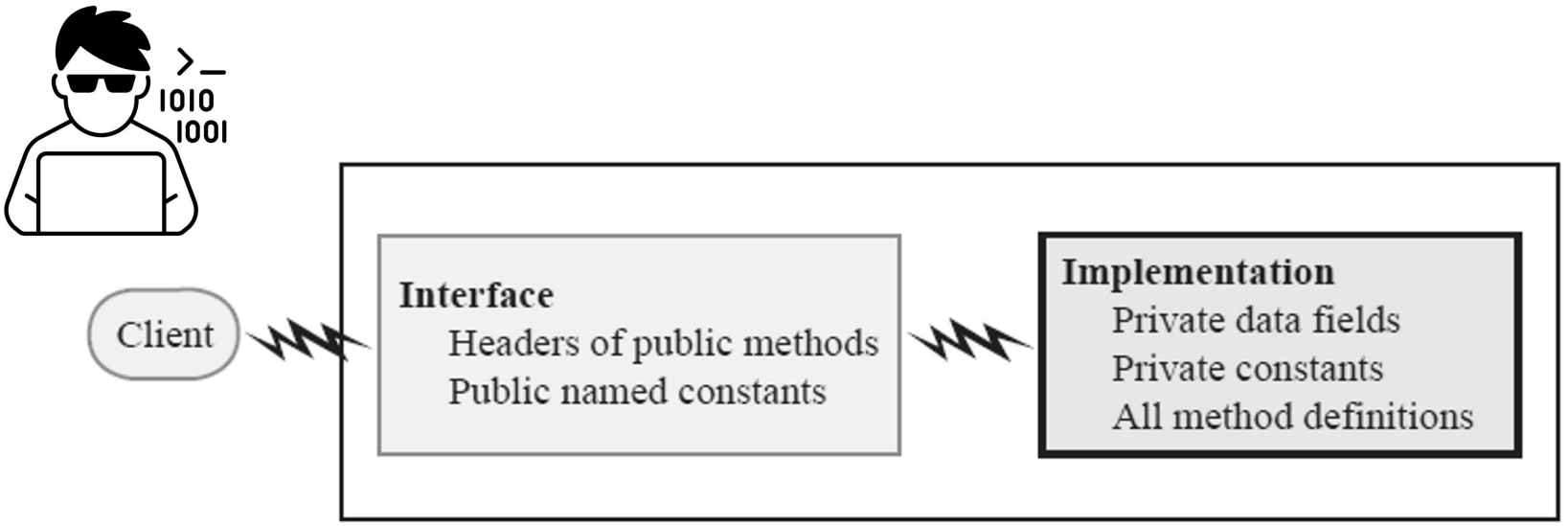

Figure P-2: An interface provides well-regulated communication between a hidden implementation and a client

Image source: https://www.vecteezy.com/vector-art/2227847-programmer-computer-expert-black-linear-icon

Exercise

- How does a class interface differ from a class implementation?

Answer

- A class interface describes how to use the class.
- It contains the headers for the class's public methods, the comments, and any publicly defined constants.
- The class implementation consists of all data fields and the definitions of all methods, including public, private and protected.

Specifying Methods

Specifying Methods

- Preconditions
- Postconditions
- Assertions

Specifying Methods (cont.)

- Preconditions
  - What must be true before method executes
  - Implies responsibility for client
- Example
  - A method to compute the square root of *x*
  - Have $x > 0$ as a precondition

Specifying Methods (cont.)

- Postconditions
  - What is true after method executes
  - Describe all the effects produced by a method invocation
  - Two cases:
    - A valued method: describe the value returned by the method
    - A void method: describe actions taken and any changes to the calling object

Specifying Methods (cont.)

- Use *assertions*
  - » A statement of truth about some aspect of your program's logic
  - » Usage:
    - In comments, or
    - With `assert` statement

Specifying Methods (cont.)

- The **`assert`** statement
  - **`assert`** `sum > 0;`
    - If the boolean expression is true, do nothing
    - If it is false, an assertion error occurs and the program terminates.

```
Exception in thread "main" java.lang.AssertionError
```

  - **`assert`** `sum > 0 : sum;`

```
Exception in thread "main" java.lang.AssertionError: -5
```

Specifying Methods

- How to enable assertions in Eclipse?
  - » Open the Run Dialog (Run > Run Configurations)
  - » Click on the tab, "(x)= Arguments."
  - » Under the field for "VM arguments," type -ea to enable assertions.
  - » Click on the "Apply" and "Run" button

Exercise

- The following statements find the largest integer in the array. What assertion can you write as a comment after the `if` statement?

```
int max = 0;
for(int index = 0; index < array.length; index++){
  if(array[index] > max){
    max = array[index];
  }
  //Assertion:
}
```

Answer

```
int max = 0;
for(int index = 0; index < array.length; index++){
  if(array[index] > max){
    max = array[index];
  }
  //Assertion: max is the largest of array[0],...,array[index]
}
```

Exercise

- Create a Java class named SumTest. Sum up the numbers from 1 to 10. If the sum is not 55, assert an exception with the error value of sum.

Answer

```
public class SumTest {
  public static void main(String[] args){
    int sum = 0;

    for(int i = 1; i<=10; i++){
      sum += i;
    }

    assert sum == 55 : sum;
  }
}
```

If the sum is not 55, you will see:

```
Exception in thread "main" java.lang.AssertionError: 66
        at SumTest.main(SumTest.java:10)
```

the value of sum