Modularizing Code with Methods

Hui Chen

Department of Computer & Information Science

Brooklyn College

Objectives

- To develop reusable code that is modular, easy to read, easy to debug, and easy to maintain (§6.6).
- To apply the concept of method abstraction in software development (§6.10).

Outline

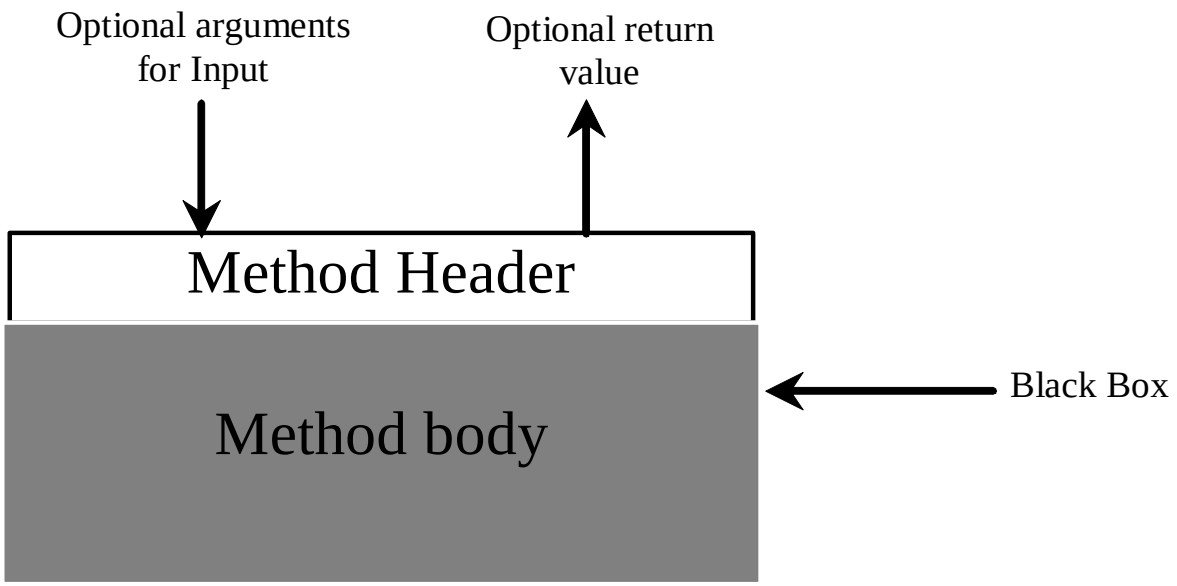
- Discussed
  - Defining and invoking value-returning methods
  - Defining and invoking void methods
  - Parameter passing and passing by value
  - Pitfalls and errors
- To discuss
  - Using method to modularize several example problems
  - Converting hexadecimal to decimal

Modularizing Code

- Methods can be used to reduce redundant coding and enable code reuse.
- Methods can also be used to modularize code and improve the quality of the program

Method Abstraction

- You can think of the method body as a black box that contains the detailed implementation for the method



Benefits of Methods

- Methods can be used to reduce redundant coding and enable code reuse.
  - Write a method once and reuse it anywhere.
  - Reduce redundancy and complexity.
- Information hiding.
  - Hide the implementation from the user.
- Methods can also be used to modularize code and improve the quality of the program

Problem. Generating Random Characters

- Write a program to generate random characters, such as, random lower case letters.

Review and Background: Characters in Java

- Each character has a unique Unicode between 0 and FFFF in hexadecimal (65535 in decimal, this is a simplification, since Unicode has …).
  - To generate a random character is to generate a random integer between 0 and 65535, e.g.,

    ```
    (int)(Math.random() * (65535 + 1))
    ```

- The Unicode for lowercase letters are consecutive integers starting from the Unicode for 'a', then for 'b', 'c', ..., and 'z'. The Unicode for 'a' is

  ```
  (int)'a’
  ```

  - So, a random integer between (int)'a' and (int)'z' is

    ```
    (int)((int)'a' + Math.random() * ((int)'z' - (int)'a' + 1)
    ```

Some Simplification

- All numeric operators can be applied to the char operands.
- The char operand is cast into a number if the other operand is a number or a character.
- So, the preceding expression can be simplified as follows

```
'a' + Math.random() * ('z' - 'a' + 1)
```

  - So a random lowercase letter is

```
(char)('a' + Math.random() * ('z' - 'a' + 1))
```

Generate Random Characters

- To generalize the foregoing discussion, a random character between any two characters ch1 and ch2 with ch1 < ch2 can be generated as follows

```
(char)(ch1 + Math.random() * (ch2 – ch1 + 1))
```

Solution. Generate Random Characters

```
public class RandomCharacter {
  /** Generate a random character between ch1 and ch2 */
  public static char getRandomCharacter(char ch1, char ch2) { // TODO }

  /** Generate a random lowercase letter */
  public static char getRandomLowerCaseLetter() { // TODO  }

  /** Generate a random uppercase letter */
  public static char getRandomUpperCaseLetter() {   // TODO  }

  /** Generate a random digit character */
  public static char getRandomDigitCharacter() { // TODO }

  /** Generate a random character */
  public static char getRandomCharacter() { // TODO }
}
```

Questions

More Programming Problems

- Rewriting using methods
  - Greatest Common Divisors
  - Print Prime Numbers
  - Is Palindrome?
  - Converting Decimal to Hexadecimal
- New problem
  - Convert Hexadecimal to Decimal

Questions