Chapter 5 Methods

Motivating Example

- Often we need to find the maximum between two numbers

```java
int result;
if (num1 > num2)
    result = num1;
else
    result = num2;
```

- A method is a construct for grouping statements together to perform a function.
- A method is defined and implemented once but can be used repeatedly
- A method can be used as a blackbox

Method Abstraction

- Method abstraction a black box that contains the detailed implementation for the method.

Benefits of Methods

- Write a method once and reuse it anywhere.
- Information hiding: hide the implementation from the user
- Reduce complexity

Defining and Using Methods

- Define a method – give a definition of what the method is to do

```java
modifier returnType methodName(list of parameters) {
    collection of statements;
}
```

- Call or invoke a method – use a method

```java
methodName(list of parameters);
```

Return Value Type

- A method may return a value (int, double, char, String, …) – value-returning method
- A method may perform desired operations without returning a value (void) – void method
Method Signature

- **Method signature**
  - The combination of the method name and the parameter list

main method

- The main method is a method that's invoked by JVM
- The main method's header is always the same
  
  ```java
  public static void main(String[] args) {
    // method body
  }
  ```
- The statements in main method may invoke other methods that are defined in the class or in other classes

  ```java
  System.out.println("Hello World!");
  int number = (int) (Math.random() * 100);
  ```

Parameters

- The variables defined in the method header are known as formal parameters
- When invoking a method, a value is passed to the parameter and this value is referred to as actual parameter or argument.
- The arguments must match the parameters in order, number, and compatible type
- Parameters are optional

Example

- A program that defines and uses a method `max` to return the larger of the two `int` values

  ```java
  TestMax.java
  ```

Method Body

- The method body contains a collection of statements
- A return statement is required for a value-returning method

Calling Methods

- `pass the value of i`
- `pass the value of j`
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println(
        "The maximum between " + i + " and " + j + " is " + k);
}

public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println("The maximum between " + i + " and " + j + " is " + k);
}

public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println("The maximum between " + i + " and " + j + " is " + k);
}

public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println(
        "The maximum between " + i + 
        " and " + j + " is " + k);
}

public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
Passing Parameters

- When calling a method, the arguments must match the parameters in order, number, and compatible type.

```java
public static void nPrintln(String message, int n) {
    for (int i = 0; i < n; i++)
        System.out.println(message);
}
public static void main(String[] args) {
    nPrintln("Hello!", 3);
    nPrintln("So that's how the methods work", 10);
}
```

- When invoking a method, the value of the argument is passed to the parameter. The variable itself is not affected. This is referred to as pass-by-value.

Pass by Value

```
public static void main(String[] args) {
    int x = 3;
    System.out.println("x in main method: "+ x);
    someMethod(x);
    System.out.println("x in main method: " + x);
}
```
Program Example - Pass by Value

- The program demonstrates the effect of pass-by-value
- It creates a method for swapping two variables
- The values of the arguments are not changed after the method is invoked
- What if we change the method to
  
  ```java
  swap(int num1, int num2)
  ```

  TestPassByValue.java

Modulizing code

- Write a method once and reuse it anywhere.
- Methods can be used to reduce redundant code and enable code reuse
- Methods can also be used to modularize code and reduce complexity of the program

Problem Revisited: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is prime if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Solution: The problem can be broken into the following tasks:
- Start with number 1
- Determine whether the number is prime.
- Print the number if it is prime
- Count the prime numbers so far.
- If count < 50, repeat the above steps

Overloading methods

- Method overloading: multiple methods can have the same name but different parameter lists
- Compiler determines which method is used based on the method signature (method name and parameters)

```java
public static double max(double num1, double num2) {
    if (num1 > num2)
        return num1;
    else
        return num2;
}

public static int max(int num1, int num2) {
    if (num1 > num2)
        return num1;
    else
        return num2;
}

max(1, 3);
max(1.0, 3.0);
```
Scope of Local Variables

- A local variable: a variable defined inside a method
- Scope: the part of the program where the variable can be referenced
- The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable
- A local variable must be declared before it can be used.

```java
public static void main(String[] args) {
    int a = 0;
    ... 
    if (a == 1) {
        int x = 0;
        ... 
    }
    System.out.println("x= + x);
}
```

**Error**

```java
public static void main(String[] args) {
    int a = 0;
    ... 
    if (a == 1) {
        int x = 0;
        //do some calculations on x
    }
    int x = 2;
}
```

**Okay**

Scope of Local Variables, cont.

What does the following method print?

```java
public static void method() {
    int sum = 0;
    for (int i = 0; i < 5; i++) {
        sum += i;
    }
    System.out.println("sum: + sum");
}
```

The Math Class

- Class constants:
  - PI
  - E
- Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods
Trigonometric Methods

- sin(double a)
- cos(double a)
- tan(double a)
- acos(double a)
- asin(double a)
- atan(double a)
- toRadians(double degree)
- toDegrees(double radians)

Examples:
- Math.sin(0) //returns 0.0
- Math.sin(Math.PI / 6) //returns 0.5
- Math.sin(Math.PI / 2) //returns 1.0
- Math.cos(0) //returns 1.0
- Math.cos(Math.PI / 6) //returns 0.866
- Math.cos(Math.PI / 2) //returns 0

Exponent Methods

- exp(double a)
  Returns e raised to the power of a.
- log(double a)
  Returns the natural logarithm of a.
- log10(double a)
  Returns the 10-based logarithm of a.
- pow(double a, double b)
  Returns a raised to the power of b.
- sqrt(double a)
  Returns the square root of a.

Rounding Methods

- double ceil(double x)
  x rounded up to its nearest integer. This integer is returned as a double value.
- double floor(double x)
  x is rounded down to its nearest integer. This integer is returned as a double value.
- double round(double x)
  x is rounded to its nearest integer. Returns (int)Math.floor(x+0.5)

min, max, and abs

- max(a, b) and min(a, b)
  Returns the maximum or minimum of two parameters.
- abs(a)
  Returns the absolute value of the parameter.

The max, min, and abs methods are overloaded so they work for int, long, float or double.

The random Method

- random()
  Generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() < 1.0).

Examples:
- (int)(Math.random() * 10) + 50 //returns a random integer between 50 and 59.
- (int)(Math.random() * 50) + 50 //returns a random number between 50 and 99.

In general,
- a + Math.random() * b //returns a random number between a and a + b, excluding a + b.

Generating Random Characters

- How do you generate random capital character?
  Solution: generate random number in range 65 - 90
The RandomCharacter Class

- Generalization
  - We want to generate character from range ch1 to ch2
  - Remember: characters can be treated as integers
    
    `(char)(ch1 + Math.random() * (ch2 - ch1 + 1))`

RandomCharacter.java

TestRandomCharacter.java

---

Stepwise Refinement

- The concept of method abstraction can be applied to the process of developing programs
- When writing a large program, you can use the “divide and conquer” strategy to decompose it into subproblems
- The subproblems can be further decomposed into smaller, more manageable problems

---

PrintCalendar Case Study

Write a program that reads year and month and prints out a calendar.

Enter full year (e.g., 2001): 2009
Enter month in number between 1 and 12: 2
February 2009

Sun Mon Tue Wed Thu Fri Sat
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28

PrintCalendar.java
How to get the start day for the first date in a month, assuming we know that Jan 1, 1800, was Wednesday?

Implementation

- **Top-down**
  - Implement one method in the structure chart at a time from the top to bottom
  - Stubs can be used for the methods waiting to be implemented

- **Bottom-up**
  - Implement one method in the structure chart at a time from the bottom to the top
  - For each method implemented, write a test program to test it