CS 112 Introduction to Programming

Nested Loops; Parameterized Methods

Yang (Richard) Yang
Computer Science Department
Yale University
208A Watson, Phone: 432-6400
Email: yry@cs.yale.edu
Outline

- Admin and recap
- Nested loops
- Parameterized methods
Admin

- Coding style review 6:30 pm Wednesday?

- Informal lunch next week after the review sessions?
Recap: Coding Style

System.out.print("T-minus ");
for (int i = 10; i >= 1; i--) {
    System.out.print(i + ", ");
}
System.out.println("blastoff!");

System.out.print("T-minus ");
for (int i = 1; i <= 10; i++) {
    System.out.print(i + ", ");
}
System.out.println("blastoff!");

System.out.print("T-minus ");
for (int i = 0; i < 10; i++) {
    System.out.print(10-i + ", ");
}
System.out.println("blastoff!");

Counting down or up, starting w/ 0 or 1 is mostly a personal style.
Recap: Coding Style

Convey your intention to the computer to help you.

```java
final int N = 10;
System.out.print("T-minus ");
for (int i = 1; i <= N; i++) {
    System.out.print(N+1-i + ", ");
}
System.out.println("blastoff!");
```

Minimize # of magic numbers (make change easier).
Public static void main() {
    final int N = 10;
    int i;
    for (i = 1; i <= N; i++) {
        System.out.print(N+1-i + " ");
    }
    System.out.println();
    for (i = 1; i <= N; i++) {
        System.out.print(N+1-i + " ");
    }
}

Personally, I prefer this one as it declares variables with minimal scope.

Recap: Coding Style
Recap: Nested Loop

```java
for (int set = 1; set <= 5; set++) {
    for (int rps = 1; rps <= set; rps++) {
        System.out.print("*");
    }
    System.out.println();
}
```

- There can be another loop inside one loop to form a nested loop
- Nested loops are common, important programming patterns.
A good practice of nested for loops is to draw ASCII art.

Why draw ASCII art?

- Real graphics will require some finesse (Monday).
- ASCII art has drawing constraints and hence requires good algorithmic thinking.
A size SIZE (S) ASCII X has 2 * SIZE rows, 2*SIZE columns
Top-Down Decomposition

A size SIZE (S) ASCII X has 2 * SIZE rows, 2*SIZE columns

This decomposition is hard to implement because ASCII drawing cursor movement constraint: left -> right, top -> bottom

Better decompose along movement line
Drawing ASCII X

A size SIZE (S) ASCII X has 2 * SIZE rows, 2*SIZE columns

1. Bound

2. Top half (drawV)

3. Bottom half (top half upside-down, drawUV)

4. Bound
A size SIZE (S) ASCII X has 2 * SIZE rows, 2*SIZE columns

```
SIZE 3
==    ==
\    /  \
|  /    |
| /      |
|/        |
==    ==

SIZE 4
==    ==
\    /  \
|  /    |
| /      |
|/        |
==    ==

SIZE 5
==    ==
\    /  \
|  /    |
| /      |
|/        |
==    ==
```

Top-Down Decomposition
A size SIZE (S) ASCII X has 2 * SIZE rows, 2*SIZE columns

== , 2 SIZE - 2 *2 spaces, ==
Two dimensional => nested loop
S - 1 lines

for (int line = 1;
     line <= S-1; line++)
{
    ...}

Structure of each line:
• some white space
• /
• some white space
Top Half (V)

Structure at line:
- some white space
  - line # white spaces
- \ /
- some white space
  - slope : -2
  - line=1: 2S-4 spaces
  - V0 = 2S-4 + 2
  - 2S - 2 - 2 line spaces
- /
Bottom Half (IV)

Structure at line:
• some white space
  • slope : -1
  • line=1: S-1 spaces
  • V0 = S
  • S - line spaces

• /

• some white space
  • slope : 2
  • line=1: 0 spaces
  • V0 = -2
  • -2 + 2 line spaces

• \
Complete Pseudo Code

1. Bound
   - == , 2S - 4 spaces, ==

2. for line = 1 to S -1
   line spaces
   \  
   2 S - 2 - 2 * line spaces
   /  

3. for line = 1 to S - 1
   S - line spaces
   /  
   (line - 1) * 2 spaces
   \  

4. Bound
Identify Subtask (Method)?

1. **Bound**
   - $==, 2S - 4$ spaces, $==$

2. **for line = 1 to $S - 1**
   - **line spaces**
   - $2S - 2 - 2 \times \text{line spaces}$

3. **for line = 1 to $S - 1**
   - $S - \text{line spaces}$
   - $(\text{line} - 1) \times 2 \text{ spaces}$

4. **Bound**

Drawing spaces is a reusable function.
**Top-Down Decomposition**

Issue: Drawing spaces need to draw different numbers of spaces.
Method Parameterization

- Specify a parameter to control the behavior of a method
  - Methods with parameters solve an entire class of similar problems

- Redundancy removal/abstraction through generalization
  - The more general a building block, the easier to reuse it
  - We will learn more techniques on generalization/abstraction
Parameterization

- **parameter**: A value passed to a method by its caller, e.g.,
  
  - When *declaring* a method, we will state that it requires a parameter for the number of spaces.
  
  - When *calling* the method, we will specify the number.
Syntax: Declaring a Method with a Parameter

```java
public static void <method_name> (<type> <param_name>) {
    <statement>(s);
}
```

- The parameter is called the **formal argument**

- Example:

```java
public static void chant(int times) {
    for (int i = 1; i <= times; i++) {
        System.out.println("Just a salad...");
    }
}
```
How Parameters are Passed

- When a method with a formal argument is called:
  - A value is passed to the formal argument
  - The passed value is called the **actual argument**
  - The method's code executes using that value.

```java
public static void main(String[] args) {
    chant(3);
    chant(3+4);
}

public static void chant(int times) {
    for (int i = 1; i <= times; i++) {
        System.out.println("Just a salad...");
    }
}
```
Common Errors

- If a method accepts a parameter, it is illegal to call it without passing any value for that parameter.
  
  ```
  chant(); // ERROR: parameter value required
  ```

- The value passed to a method must be of the correct type.
  
  ```
  chant(3.7); // ERROR: must be of type int
  ```
for (int set = 1; set <= 5; set++) {
    for (int rps = 1; rps <= set; rps++) {
        System.out.print("*");
    }
    System.out.println();
}

Exercise: Turn the inner loop into a method to be invoked by the outer loop.
Method Exercise

Exercise: Design and implement the DrawX program.
Out-Class Read: Another Way to Motivate Methods With Parameters
Example: Draw Matrix

- Print two copies of inverse diagonal matrix

```
..1
.2.
3..
....1
...

..3..
.
4...
5....
```

```java
int N = 5;
for (int line = 1; line <= N; line++) {
    // initial dots
    for (int j = 1; j <= (-1 * line + N); j++) {
        System.out.print(".");
    }

    // the number
    System.out.print(line);

    // the second set of dots
    for (int j = 1; j <= (line - 1); j++) {
        System.out.print(".");
    }
    System.out.println();
}
```
public class DrawMatrix1 {
    public static void main(String[] args) {
        drawMatrix3();
        drawMatrix5();
    }

    public static void drawMatrix3() {
        int N = 3;
        for (int line = 1; line <= N; line++) {
            for (int j = 1; j <= (-1 * line + N); j++) {
                // initial dots
                System.out.print(".");
            }
            System.out.print(line); // the number
            for (int j = 1; j <= (line - 1); j++) {
                // the second set of dots
                System.out.print(".");
            }
            System.out.println();
        }
    }

    public static void drawMatrix5() {
        int N = 5;
        for (int line = 1; line <= N; line++) {
            for (int j = 1; j <= (-1 * line + N); j++) {
                // initial dots
                System.out.print(".");
            }
            System.out.print(line); // the number
            for (int j = 1; j <= (line - 1); j++) {
                // the second set of dots
                System.out.print(".");
            }
            System.out.println();
        }
    }
}

• This code is redundant.

Solution 1
public class DrawMatrix2 {

    public static void main(String[] args) {
        int N = 3;
        drawMatrix(); // should print 3

        N = 5;
        drawMatrix(); // should print 5
    } // end of main

    public static void drawMatrix() {
        for (int line = 1; line <= N; line++) {
            for (int j = 1; j <= (-1 * line + N); j++) { // initial dots
                System.out.print(".");
            }
            System.out.print(line);
            // the number
            for (int j = 1; j <= (line - 1); j++) { // the second set of dots
                System.out.print("");
            }
            System.out.println();
        } // end of for line
    } // end of drawMatrix
public class DrawMatrix2 {

    static int N; // introduce a class scope variable

    public static void main(String[] args) {
        N = 3;
        drawMatrix(); // should print 3

        N = 5;
        drawMatrix(); // should print 5
    } // end of main

    public static void drawMatrix() {
        for (int line = 1; line <= N; line++) {
            for (int j = 1; j <= (-1 * line + N); j++) { // initial dots
                System.out.print("");
            }
            System.out.print(line);
            for (int j = 1; j <= (line - 1); j++) { // the second set of dots
                System.out.print("");
            }
            System.out.println(); // end of for line
        } // end of for line
    } // end of drawMatrix
}
Problem of Class Variable Indirection

```java
public class DrawMatrix2 {
    static int N; // introduce a class scope variable

    public static void main(String[] args) {
        drawMatrix();
        drawMatrix();
    } // end of main

    public static void drawMatrix() {
        for (int line = 1; line <= N; line++) {
            for (int j = 1; j <= (-1 * line + N); j++) {
                // initial dots
                System.out.print(".");
            }
            System.out.print(line); // the number
            for (int j = 1; j <= (line - 1); j++) {
                // the second set of dots
                System.out.print(".");
            }
            System.out.println();
        } // end of for line
    } // end of drawMatrix
}
```

Not self clear; Implicit, depends on context;
Solution 3: Parameterization

- Specify a parameter to control the behavior of a method
  - Methods with parameters solve an entire class of similar problems

- Redundancy removal/abstraction through generalization
  - The more general a building block, the easier to reuse it
  - We will learn more techniques on generalization/abstraction
End of Out-Class Read