CS 112  Introduction to Programming

Variable Scoping;
Nested Loops;
Parameterized Methods

Yang (Richard) Yang
Computer Science Department
Yale University
308A Watson, Phone: 432-6400
Email: yry@cs.yale.edu
Admin

- **PS1**: We encourage that you go over any issues with any of the teaching staff
- **PS3 to be posted Thursday morning**
- **Coding style reviews led by Debayan**: 5-6pm today @ DL 220
- **Informal lunch together at Commons**: This Friday at 12:40 pm, Next Thursday
Recap: Code Style

Counting down or up is mostly a personal style.

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

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!");
```
Recap: Code Style

Minimize # magic numbers.

System.out.print("T-minus ");
for (int i = 1; i <= 10; i++) {
    System.out.print(11-i + ", ");
}
System.out.println("blastoff!");

Convey your intention to the computer to help you.
What if we want to print out the values of \( N \) and \( i \) after the loop:

```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!");
System.out.println("N = "+ N); //?
System.out.println("Final i =" + i); //?
```
Counting Down: Code Puzzle

% javac CountDownValue.java
CountDownValue.java:25: cannot find symbol
  symbol   : variable i
  location: class CountDownValue
     System.out.println( "Final i = " + i );
           ^
1 error
Variable Scope

- **Scope**: The part of a program where a variable exists.

- **Basic rule**: from its declaration to the end of the enclosing `{ }` braces

- **Examples**
  - A variable declared in a `for` loop exists only in that loop.
  - A variable declared in a specific method exists only in that method.
  - A variable declared not inside any method but in a class is said to have class scope.
public class CountDown {
    static int N = 10;
    public static void main(String[] args) {
        countDown();
    }

    public static void countDown() {
        System.out.print("T-minus ");
        int sum = 0;
        for (int i = 1; i <= N; i++) {
            System.out.println(N + 1 - i);
            sum += i;
        }
        System.out.println("N: " + N);
        System.out.println("Sum: " + sum);
    } // end of countDown
} // end of class
Why Scope?

- Encapsulation
  - e.g., different methods can use the same variable name without the need for coordination
  - many analogies: folders allow same file name so long in different folders

```java
public static void aMethod()
{
    int x = 1
    ...
}
```

```java
public static void bMethod()
{
    int x = 2;
    ...
}
```
Does the following code work?

```java
public static void main() {
    for (int i = 1; i <= 10; i++) {
        System.out.print(11 - i + " ");
    }
    System.out.println();
}

for (int i = 1; i <= 10; i++) {
    System.out.print(11 - i + " ");
}
}

Output:
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
Does the following code work?

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

Output:

```
*
**
***
****
*****
```

Loop Example
A loop inside another loop is said to form a nested loop

The #loop times of the inner loop can depend on the outer loop variable
Practice: Nested for loop example

What is the output of the following nested for loops?

```java
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

Output:

1
22
333
4444
55555
What is the output of the following nested for loops?

```java
for (int i = 1; i <= 9; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(i*j + "\t");
    }
    System.out.println();
}
```
Nested Loop Design Example: Drawing A Complex Pattern

- Use nested for loops to draw ASCII X
- A size SIZE ASCII X has 2 * SIZE rows
- Why draw ASCII art?
  - Real graphics will require some finesse (shortly)
  - ASCII art has complex patterns
  - Can focus on the algorithms
Design Decomposition

1. Bound
   • == , (SIZE - 2)*2 spaces, ==

2. Top half (V)

3. Bottom half (top half upside-down)

4. Bound

SIZE 3

SIZE 4

SIZE 5
Observation: V can be produced using a nested for loop

outer loop (loops SIZE – 1 times)
Outer and inner loop

- First write the outer loop, from 1 to the number of lines.

```java
for (int line = 1; line <= SIZE-1; line++) {
    ...
}
```

- Now look at the line contents. Each line has a pattern:
  - some white space
  - \\
  - some white space
  - /

```
==    ==
1 \    /
2 \  /
3 /\
```

```
==    ==
```
Final Pseudo Code

1. Bound
   - == , (SIZE - 2)*2 spaces, ==

2. for line = 1 to SIZE -1
   line spaces
   \ 
   2 (SIZE - 2) + 2 - 2 * line spaces
   /

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

4. Bound
Implementation Problem

1. Bound
   • == (SIZE - 2)*2 spaces, ==

2. for line = 1 to SIZE -1
   line spaces
   2 SIZE - 2 - 2 * line spaces

3. for line = 1 to SIZE - 1
   SIZE - line spaces
   (line - 1) * 2 spaces

4. Bound

Drawing spaces is a reusable function, but 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.
Declaring a Parameter

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

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

- Example:

```
public static void sayPasswcode(int code) {
    System.out.println("The passcode is: " + code);
}
```
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);
}
```

```java
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.

```java
chant(); // ERROR: parameter value required
```

- The value passed to a method must be of the correct type.

```java
chant(3.7); // ERROR: must be of type int
```
Method Exercise

- Exercise: Design and implement the DrawX program.
Multiple Parameters

- A method can accept multiple parameters. (separate by , )
  - When calling it, you must pass values for each parameter.

- Declaration:
  ```java
  public static void <name>(<type> <name>, ..., <type> <name>) {
      <statement>(s);
  }
  ```

- Call:
  ```java
  <name>(<exp>, <exp>, ..., <exp>);
  ```
public static void main(String[] args) {
    printNumber(4, 9);
    printNumber(17, 6);
    printNumber(8, 0);
    printNumber(0, 8);
}

public static void printNumber(int number, int count) {
    for (int i = 1; i <= count; i++) {
        System.out.print(number);
    }
    System.out.println();
}

Output:
444444444
1717171717
000000000
Multiple Parameter Invocation

- Corresponding actual *argument* in the invocation is copied into the corresponding *formal argument*

```java
printNumber(2, 5);

public static void printNumber(int number, int count)
{
    // equiv: number = 2; count = 5;
    for (int i = 1; i <= count; i++) {
        System.out.print(number);
    }
    System.out.println();
}
```
Java Graphics Methods

- Java provides a large number of methods for graphics
  - We use graphics to see many examples of methods with parameters and loops

- To simplify the usage the Graphics methods, multiple libraries are provided
  - Textbook: define class `DrawingPanel`, which contains many Graphics methods
  - Sedgewick & Wayne book: defines class `StdDraw`, which contains many Graphics methods
Access Methods

- To access a method or class variable defined in another class, using the `<class-name>.<method-name>(...)`, for example,
  - `StdDraw.setCanvasSize(100, 100);`
StdDraw Methods

```java
void line(double x0, double y0, double x1, double y1)
void point(double x, double y)
void text(double x, double y, String s)
void circle(double x, double y, double r)
void filledCircle(double x, double y, double r)
void square(double x, double y, double r)
void filledSquare(double x, double y, double r)
void polygon(double[] x, double[] y)
void filledPolygon(double[] x, double[] y)
void setXscale(double x0, double x1)  // reset x range to (x0, x1)
void setYscale(double y0, double y1)  // reset y range to (y0, y1)
void setPenRadius(double r)  // set pen radius to r
void setPenColor(Color c)  // set pen color to c
void setFont(Font f)  // set text font to f
void setCanvasSize(int w, int h)  // set canvas to w-by-h window
void clear(Color c)  // clear the canvas; color it c
void show(int dt)  // show all; pause dt milliseconds
void save(String filename)  // save to a .jpg or .png file
```
Color and Class Constants

- **class constant**: A static class variable with a fixed value
  - value can be set only at declaration; cannot be reassigned

- **Syntax**: 
  
  ```java
  public static final type name = value; // in class scope
  ```
  
  - name is usually in **ALL_UPPER_CASE**

- **Examples**:
  
  ```java
  public static final int DAYS_IN_WEEK = 7;
  public static final double INTEREST_RATE = 3.5;
  public static final int SSN = 658234569;
  ```
Color

- Java predefine many class constants in the `Color` class:
  
  ```java
  Color.COLOR_CONSTANT_NAME
  ```

  where `COLOR_CONSTANT_NAME` is one of:

  - BLACK, BLUE, CYAN, DARK_GRAY,
  - GRAY,
  - GREEN, LIGHT_GRAY, MAGENTA, ORANGE,
  - PINK, RED, WHITE, YELLOW

http://download.oracle.com/javase/6/docs/api/java/awt/Color.html
Example: Using Colors

- Pass a `Color` to `StdDraw's setPenColor` method
  - Subsequent shapes will be drawn in the new color.

```java
StdDraw.setPenColor(Color.BLACK);
StdDraw.filledRectangle(10, 30, 100, 50);
StdDraw.line(20, 0, 10, 30);
StdDraw.setPenColor(Color.RED);
StdDraw.filledEllipse(60, 40, 40, 70);
```

See SimplePanel.java
Exercise