

Introduction to Programming

with Java, for Beginners

Style Rules
Debugging Tips
Scope

Coding Style Rules/Conventions

- Hardly any software is maintained for its whole life by the original author
- Coding conventions make code easier to read, debug, and modify
 - Both for the author and for others
- If you ship code as a product, it should be clean and "well engineered".

1

Style Rules #1: Comments

- Put a comment above every class and non-trivial method
- Use it to
 - Indicate authors of the program
 - When it was last modified
 - To explain your algorithm
 - Approach to your solution
 - How you decomposed your problem ?

2

Style Rule # 2: Naming

- Use descriptive names for classes, methods and variables
- Start classes with Capital Letter
- Start methods and variables name with lower case letter
 - > 1 word, then 2nd word should be capitalized (this also applies to classes)
- Exceptions to variables with single letter
 - Used in loops (for and while)
 - E.g. i, k, j etc

3

Style Rule # 3: Spaces

- You should put a single space around every binary operator, including comparisons and =
- Example:
`perimeter = 2 * (width + height);`
- Do *not* put spaces just inside parentheses:
`perimeter = 2 * (width + height); //bad`

4

Style Rule # 4: Indentation

- `if (x < 10) {`
 `x = x + 1;`
 `}`
 `else {`
 `x = x - 1;`
 `}`
- Recommended indentation is from 2 to 4 spaces, but *must be consistent* throughout the program
- In Dr Java you can set the indent level:
 Edit > Preferences > Miscellaneous
- `while (x > 0) {`
 `System.out.println(x);`
 `x = x - 1;`
 `}`

5

Breaking Style Rules

- These are *style rules*, not Java rules
 - If you break these rules, your program will still compile
 - This is done so that your code is readable and clear
- If you do not follow the rule you get *penalized* in grading!
- More info:
<http://www.cis.upenn.edu/~palsetia/java/styleRules.html>

6

Plan Ahead

- Start Early
- Make a plan on paper
- Write your algorithm/recipe/approach in a series of comments.
- Then write code below each comment/step.

7

Reading Error Messages

- *Read the error messages* and *note the line number*
- Turn on Line Numbers in DrJava
 - Edit Preferences -> Display
- Fix the first error listed first. Recompile and repeat.

8

Debugging with System.out.println

```
public static int sumOneTo(int num){
    //Complete the method
    int sum = 0;
    for (int i = 0; i <= num; i = i + 1){
        sum = sum + i;
        System.out.println("Partial Sum = " + sum);
    }
    return sum;
}
```

Remember to comment it out when you are done testing

9

Scope

- *Scope* means the area of code in which an entity is known (or alive)
- We will discuss the scope of a:
 - Variable - what code can access it?
 - Method - what code can call it?
- Sometimes scope is *explicitly* designated with a keyword
 - *private*: known only within the class
 - *public*: known outside of (and within) the class
- Other times it is *implicitly* designated by location
 - e.g. method parameters are known only in the method in which they are defined

10

Method Parameters

- A method parameter is an “input variable”
- *Scope*: the method in which it is defined
- It “comes alive” when the method is entered
- It “dies” when the method is exited
- No other method can access (read/write) it

Note: The input variable should be modified within a method

11

Local Variables

- A “local variable” is defined within a method body { }
- They are inherently private to the method in which they are defined
- It may be defined in a block { } within a method body
- Scope: point of declaration to end of closest enclosing block
- We don't use public/private for local variables

12

Example

Code	Notes
<pre>// Converts 0,1,2,3 to // "north", "east", "south", "west" public String direction(int d){ String result; if (d == 0) result = "north"; else if (d == 1) result = "east"; else if (d == 2) result = "south"; else if (d == 3) result = "west"; else result = "unknown"; return result; } }</pre>	<p><i>Method parameters (e.g. <code>d</code>) are only known within the method.</i></p> <p><i>Local variables (e.g. <code>result</code>), are known from the point of declaration until the end curly brace of the block in which they are declared.</i></p>