“The Elements of Java Style”

COMP SCI / SFWR ENG 2S03

Natalie Perna

Department of Computing and Software
McMaster University

Week 2: Sept 17 - 21

Outline

Resource

All of the following material is adapted from:
Vermeulen, Ambler, Bumgardner, et al. "The Elements of Java Style". Cambridge New York: Cambridge University Press SIGS Books, 2000.

Disclaimer

Most rules are not covered in their entirety throughout these slides. Reference "The Elements of Java Style" for full details, including exceptions to some of the rules stated here.

Indentation

- Improves code readability
- Use 2 spaces (or indentation style produced by IDE)
- Do *not* use tabs
- '{' at end of first line of block; '}' on own line at end of block

```
class MyClass {
**void function(int arg) {
****if (arg < 0) {
******for (int i = 0; i <= arg; i++) {
********// Code here
******}
****}
**}
}
```

Single Spaces

To separate:

- ')' or '}' from keyword that immediately follows
- '(' or '{' from keyword that immediately precedes
- ')' from '{' that immediately follows
- Binary operator (except ".") from expressions preceding and following

```
if∗(x∗+∗y∗>∗0)∗{
   ...
}
else∗if∗(x∗+∗y∗<∗0)∗{
   ...
}
else∗{
   ...
}
```

Blank Lines I

To separate:

- Logical sections of a method
- Members of a class/interface definition

Blank Lines II

```
void handleMessage(Message messsage) {

  DataInput content = message.getDataInput();
  int messageType = content.readInt();

  switch (messageType) {

    case WARNING:
      ... do some stuff here ...
      break;

    default:
      ... do some stuff here ...
      break;

  }
}
```

General Rules I

- Use meaningful names (i.e. "age" is more meaningful than "a")
- Exception: Some temporary variables (i.e. "i" for an loop index/counter)
- Use constants for values that can be meaningfully described

General Rules II

Example:

```
if (a < 65) { // What property does 'a' describe?
  y = 65 - a; // What is being calculated here?
}
else {
  y = 0;
}
```

vs.

```
if (age < RETIREMENT_AGE) {
  yearsToRetirement = RETIREMENT_AGE - age;
}
else {
  yearsToRetirement = 0;
}
```

General Rules III

- Tip: If you can't describe your object with a short, simple name, you may be trying to accomplish too much with a single object
- Do not shorten words or remove vowels (i.e. use "message" not "msg")
- Capitalize first letter only in acronyms (i.e. use "importHtml" not "importHTML")

Classes

- Capitalize the first letter of each word (i.e. “FilterOutputStream”)
- Use nouns (remember that classes define objects) (i.e. “CustomerAccount”)

Methods

- First letter should be lowercase (i.e. "deposit")
- Each subsequent word in the name begins with a capital letter (i.e. "withdrawFromSavings")
- Use verbs (remember that methods define actions)
- Name accessor methods using "is", "get", "set"

Variables

- First letter should be lowercase (i.e. "address")
- Each subsequent word in the name begins with a capital letter (i.e. "billingAddress")
- Use nouns (remember that variables refer to objects) (i.e. "shippingAddress")
- Pluralize names of collection references (i.e. Customer[] customers = newCustomer[MAX_CUSTOMERS];)

Fields I

- Qualify field variables with “this” keyword to distinguish from local variables
- When a constructor or “set” method assigns a parameter to a field, give that parameter the same name as the field.

Fields II

```
class Dude {

  private String name;

  public Dude(String name) {
    this.name = name;
  }

  public setName(String name) {
    this.name = name;
  }
}
```

Constants

- Use all uppercase letters (i.e. "AGE")
- Separate words with an underscore (i.e. "MAX_AGE")

Documentation Comments

- Begins with "/**" and ends with "*/"
- Used to describe programming interface

```
/**
 * The Rectangle2D class describes
 * a rectangle defined by location (x,y) and
 * dimensions (w,h).
 */
public abstract class Rectangle2D
  extends RectangularShape {
    // ... Class contents here ...
}
```

Standard Comments

- Begins with "/*" and ends with "*/"
- Used to (temporarily) hide code without removing it

```
/*
 I have temporarily removed this method because
 it has been deprecated for some time, and I
 want to determine whether any other packages
 are still using it! — J. Kirk on 9 Dec 1997

public void thisOldFunction() {
  // There has got to be a better way!
   ...
}
*/
```

One-Line Comments

- Begins with "//"
- Single line or end-line
- Used to explain implementation details:
  - The purpose of specific variables or expressions
  - Implementation-level design decisions
  - The source material for complex algorithms
  - Defect fixes or workarounds
  - Code that may benefit from further optimization or elaboration
  - Known problems, limitations, or deficiencies