Coding Standard & Javadoc

Java Naming Convention

class name begins with Uppercase: `Double, String`

method name uses camelCase: `length(), valueOf()`

variable name also uses camelCase: `myCoffee`

constants use UPPER_CASE and _: `MAX_VALUE`

package names are all lowercase (with a few exceptions):

```
java.lang  java.io  java.time  java.util
org.junit
```

primitive type names are all lowercase:

```
boolean, char, int, double, long
```

Example - a Java class

```
package ku.ske;
import java.util.Scanner;

/** A customer has one or more accounts */
public class Customer extends Person {
   private String customerId;
   private List<Account> accounts;

   public Customer(String name) . . .

   public List<Account> getAccounts( )...
   public void addAccount(Account acct)...
```

Identify each of these

Martian
org.nerd.hacker
System
System.nanoTime( )
System.out
System.out.println( )
double
Double
"Hello nerd".length( )
Double.MAX_VALUE
java.text
java.util.ArrayList
java.util.*Comparable*

Is it a ...

class

package

primitive type

attribute ("field")

method

(static or instance)

constant

(static final attribute)

interface *(more advanced)*

something else???

Use Full Words as Names

Good Name	Bad Name
BankAccount	BankAct
balance	bal
count	n
accountId	num, id

Exception: short names OK for local variables, esp. loop vars.

```
double getTotal( ) {
    double sum = 0.0;
    for(int k=0; k<transactions.length; k++) {
        sum += transaction.getValue( );
        ...
```

Writing Javadoc (Required)

```
package ku.ske;
/**
 * A Person contains information about a
 * person including name and contact info.
 * @author Bill Gates
 * @since 2014.01.12
 */
public class Person {
    /** person's name, of course */
    private String name;
```

Must start with a complete sentence, ending with a period.

@author, @since are tags.
Use only the standard tags.

Method Javadoc

```
/**
 * Set the person's birthday.
 * @param birthday a date containing the
 *   person's birthday. Must not be null.
 */
public void setBirthday(Date birthday) {
   if (birthday == null)
     throw new IllegalArgumentException(
      "Read the javadoc, stupid!");
   this.birthday = (Date)birthday.clone();
   .
```

Method Javadoc with Return

```
/**
 * Withdraw money from the coin machine.
 * @param amount is amount to withdraw.
 * @return array of money withdrawn,
 *    or null if cannot withdraw the
 *    requested amount.
 */
public Money[] withdraw(double amount) {
   if (double <= 0.0) return null;
   .
   .
```

More Method Javadoc

```
/**
 * Compare 2 coins by value.
 * @param a the first Coin to compare.
 * @param b the second Coin to compare
 * @return -1 if first coin's value is
 *   smaller, +1 if first coin's value is
 *   larger, and 0 if values are same.
 * @throws IllegalArgumentException if
 *   the currencies are not the same.
 */
public int compare(Coin a, Coin b) {
    .
```

Bad Javadoc - what's wrong?

```
/**
 * The Person class
 * @Bill Balmer
 * @Version 1.0
 */
package ku.ske.badcode;
public class Person {
    private String name;
    /**
     * get the firstname
     * @param k is index of first space in name
     */
    public String getFirstname( ) {
         int k = name.indexOf(' ');
         return name.substring(0,k); // bug?
    }
```

Good Code has Documentation

- Use documentation to describe a class and its methods.
- Describe what and why – not "how" which is obvious from the code.
- Describe rationale and logic that is not obvious from code.

No Javadoc = No Credit

```
// sum elements in the array (BAD: it's obvious)
int sum = 0;
for(int k=0; k<array.length; k++) {
    sum += array[k];
}
```

Generate Javadoc from your Code

3 ways to create HTML pages from Javadoc:

- the `javadoc` command
- let Eclipse or BlueJ or Netbeans create it for you
- use an automatic build system, like Maven

Demo

Demo how BlueJ will create Javadoc (HTML) from your Javadoc comments.

Demo how any IDE gives interactive help using Javadoc, including Javadoc in your code.

`javadoc` Command

You can use the javadoc command line to create Javadoc. It has many options, only a few are shown here.

First, open a terminal "shell" and "cd" to the top-level directory of your project:

```
cmd>  cd workspace/lab2

cmd>  dir

stopwatch/

util/
```

This shows your source code is in packages named `stopwatch` and `util`.

Using the `javadoc` Command

Create HTML doc files in a directory named "docs", and include everything with visibility package (default) and above, and include @author tags

```
cmd>  javadoc --frames -author -package
                -d docs stopwatch util

Constructing Javadoc information...
Standard Doclet version 11.0.5
Building tree for packages and classes...
Generating docs/stopwatch/Main.html...
Generating docs/stopwatch/Stopwatch.html...

Generating docs/index.html...
```

View Your Javadoc!

Open the file `docs/index.html` in a web browser: