CS-259 Coding Standards

- All project and Labs must follow the great and hallowed CS-259 coding standards.
- These standards do not necessarily represent the best nor the only good way to write Java code.
- If you have experience programming, then these standards may not be the standards you are used to using.
- However, in this class, these are the standards we will use.
Primary Reasons for Defined Standard

1. A standard makes it easier for the instructors to read your code.

2. A class standard makes it easier for a grader to recognize when a program does not use a consistent standard.
   Often when each student is allowed to define his or her own standard, students switch standards multiple times in a single project. It is tedious for a grader to deduce each person’s standard and then check for self-consistency.

3. It is good practice to learn to follow a standard.

Coding Standard: Naming

- All variable names (fields) shall begin with a lower case letter.
- All instance variables will be given descriptive names.
- All methods will be given descriptive names.
- All class names shall begin with an uppercase letter.
- All variables declared with `final` shall be all uppercase.
Coding Standard: Comments

- Each Java Source file shall begin with a heading in the form:
  ```
  /**
   * @version 1.2 Aug, 24 2009
   * @author Ender Wiggin
   */
  ```

- Each method shall begin with a comment explaining its inputs, what it does, how it does it, and its return value.

Coding Standard – Open Brackets

Open brackets will be placed at the beginning of a line (not at the end).

<table>
<thead>
<tr>
<th>ok</th>
<th>if (x == 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{ y = y+1; }</td>
</tr>
<tr>
<td>Not CS-259 standard</td>
<td>if (x == 5) {</td>
</tr>
<tr>
<td></td>
<td>y = y+1;</td>
</tr>
</tbody>
</table>
Coding Standard – Closing Brackets

Closing brackets will be indented on a line with no other commands. The only exception being comments placed on the line with a closing bracket.

```java
if (x == 5) {
  y=y+1;
}  //Comment here ok
else if (x == 7) {
  y=y+2;
}
```

Bad

```java
if (x == 5) {
  y=y+1;
}  else if (x == 7) {
  y=y+2;
}
```

Coding Standard – Blocks and { }

- Whenever a structure spans more than one line, brackets must be used. For example:

<table>
<thead>
<tr>
<th></th>
<th>if (x == 5) y=y+1;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>if (x == 5) y=y+1;</td>
</tr>
<tr>
<td>ok</td>
<td>if (x == 5)</td>
</tr>
<tr>
<td></td>
<td>{ y=y+1;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>Not CS-259 standard</td>
<td>if (x == 5)</td>
</tr>
<tr>
<td></td>
<td>y=y+1;</td>
</tr>
</tbody>
</table>
Coding Standard - Indenting

- Code blocks will be indented to show the block structure with **two spaces** per level.
- Tab characters shall **not** be used for indenting.
- All statements within a block must be indented to the same level.

Coding Standard – 80 Character Line Max

No line shall be more than 80 characters.

The best way to avoid overly long statements is by not doing too much in a single statement.

```java
if (getVolume(length1, width1, height1) >
getVolume(length2, width2, height2)) System.out.println
("box 1 is bigger"); else System.out.println ("box 2 is bigger");
```

```java
int volume1 = getVolume(length1, width1, height1);
int volume2 = getVolume(length2, width2, height2);
if (volume1 > volume2)
{ System.out.println("box 1 is bigger");
} else
{ System.out.println("box 2 is bigger");
}
```
Fixing Too Long a Line Example 2

- Another case where a temporary variable can shorten a line and improve readability.
- Creating the temporary variable `c` also improves code maintenance:

  If the code changes so that the comparison needs to check `stack[topOfStack]` or `stack[topOfStack-2]`, then Line 2 and 3 require only a single change while line 1 requires 4 changes.

```
1 if (stack[topOfStack - 1] == '*' ||
    stack[topOfStack - 1] == '+' ||
    stack[topOfStack - 1] == '-' ||
    stack[topOfStack - 1] == '/')
2 char c = stack[topOfStack - 1];
3 if (c == '*' || c == '+' || c == '-' || c == '/')
```

Fixing Too Long a Line Example 3

- There are times when breaking a long statement into multiple statements is more awkward than keeping the long statement.
- In such cases, the statement should be broken in a logical place and each line over which the long statement is continued must be indented.
- The indenting must be at least 2 spaces, but can be more spaces it that improves readability. Code example 8, indents line 3 so that the comparisons match up.

```
1 if (commandOption == 'f' ||
    commandOption == 'c' ||
    commandOption == 'd' ||
    commandOption == 'g')
2 if (commandOption == 'f' ||
    commandOption == 'c' ||
    commandOption == 'd' ||
    commandOption == 'g')
3 if (commandOption == 'f' ||
    commandOption == 'c' ||
    commandOption == 'd' ||
    commandOption == 'g')
```
Quiz 1-4: Coding Standard

Which line does NOT follow the standard?

a: `for (i=0; i<10; i++)`

b: `{ int c = i*10;`

c: `if (c == 30)`

d: `c=c+6;`

e: `else if (c == 40) c = c-6;`

Coding Standard: Class Layout

```java
import javax.swing.JFrame;

public class GUI_Frame extends JFrame{
  //1st: All class variables
  private static final long serialVersionUID = 1L;
  private GUI_DrawPanel drawPanel;

  //2nd: Constructor(s) – if they exist.
  public GUI_Frame() ║

  //3rd: Other methods in any order – if they exist.
  public void clear() ║
  public void drawLines(int r, int g, int b) ║

  //last: main() – if it exists.
  public static void main(String[] args) ║
}
```
Coding Standard: Minimal Scope

**Variables** used in only one method shall be local variables.

**Variables** used in more than one method within a class shall be private class variables.

A **data class** (a class with no methods) is the only type of class that may have public variables.

**Constants**, even when used in only one method, may be public static final or private static final.

---

Do not hurt where holding is enough;  
do not wound where hurting is enough;  
do not maim where wounding is enough;  
and kill not where maiming is enough;  
the greatest warrior is one who does not need to kill.  
--Berek's Code, The Illearth War By Stephen R. Donaldson

---

Setting Code Style in Eclipse

Window → Preferences

- Java → Code Style → Formatter → New...
Code Style → Formatter → Indentation

Profile 'Joel_Style'

Profile name: Joel_Style

General settings
- Tab policy: Spaces only
- Use tabs only for leading indentations
- Indentation size: 2
- Tab size: 2

Alignment of fields in class declarations
- Align fields in columns

Top part shown on last slide.

Code Style → Formatter → Indentation

- Indent
  - Declarations within class body
  - Declarations within enum declaration
  - Declarations within enum constants
  - Declarations within annotation declaration
  - Statements within method/constructor body
  - Statements within blocks
  - Statements within 'switch' body
  - Statements within 'case' body
  - 'break' statements
  - Empty lines
Code Style → Formatter → Braces

Code Style → Formatter → Control Statements
As You type: Automatically Close...

Show 80 Column Print Margin
Applying Style

- Click OK to close dialog.
- Then, Right Click in source code, and select: Source → Format.

Quiz 1-5: Coding Standard

Which line does NOT follow the standard?

```java
for (i=0; i<10; i++)
{
    char c = inStr[i];
    if (c == '+') c=a+b;
    else if (c == '*') c = a*b;
    else if (c>'0' && c<='9')
    a: else if (c>='0' && c<='9')
    b: { for (j=0; j<c; j++)
    c: { System.out.print("j="+ j);
    d: }
    e: System.out.print("\n");
    }
    }
```