Clean Formatting

Christopher Simpkins
chris.simpkins@gatech.edu

Formatting

Code should be written for human beings to understand, and only incidentally for machines to execute. – Hal Abelson and Gerald Sussman, SICP

The purpose of a computer program is to tell other people what you want the computer to do. – Donald Knuth

The purpose of formatting is to facilitate communication. The formatting of code conveys information to the reader.

Vertical Formatting

- Newspaper metaphor
- Vertical openness between concepts
- Vertical density
- Vertical distance
- Vertical ordering

Vertical Openness Between Concepts

Notice how vertical openness helps us locate concepts in the code more quickly.

```
package fitnesse.wikitext.widgets;

import java.util.regex.*;

public class BoldWidget extends ParentWidget {
  public static final String REGEXP = "'''.+?'''";
  private static final Pattern pattern = Pattern.compile("'''(.+?)'''",
    Pattern.MULTILINE + Pattern.DOTALL
  );

  public BoldWidget(ParentWidget parent, String text) throws Exception
    {
    super(parent);
    Matcher match = pattern.matcher(text);
    match.find();
    addChildWidgets(match.group(1));
  }
}
```

Vertical Openness Between Concepts

If we leave out the blank lines:

```
package fitnesse.wikitext.widgets;
import java.util.regex.*;
public class BoldWidget extends ParentWidget {
  public static final String REGEXP = "'''.+?'''";
  private static final Pattern pattern = Pattern.compile("'''(.+?)'''",
    Pattern.MULTILINE + Pattern.DOTALL
  );
  public BoldWidget(ParentWidget parent, String text) throws Exception
    {
    super(parent);
    Matcher match = pattern.matcher(text);
    match.find();
    addChildWidgets(match.group(1));
  }
}
```

- It's harder to distinguish the package statement, the beginning and end of the imports, and the class declaration.
- It's harder to locate where the instance variables end and methods begin.

Vertical Density

Openness separates concepts. Density implies association. Consider:

```
public class ReporterConfig {
  /** The class name of the reporter listener */
  private String m_className;

  /** The properties of the reporter listener */
  private List<Property> m_properties = new ArrayList<Property>();

  public void addProperty(Property property) {
    m_properties.add(property);
  }
```

The vertical openness (and bad comments) misleads the reader.
Better to use closeness to convey relatedness:

```
public class ReporterConfig {
  private String m_className;
  private List<Property> m_properties = new ArrayList<Property>();

  public void addProperty(Property property) {
    m_properties.add(property);
  }
}
```

Vertical Distance and Ordering

Concepts that are closely related should be vertically close to each other.

- Variables should be declared as close to their usage as possible.
- Instance variables should be declared at the top of the class.
- Dependent functions: callers should be above callees.

Horizontal Openness and Density

- Keep lines short. Uncle Bob says 120, but he's wrong. Keep your lines at 80 characters or fewer if possible (sometimes it is impossible, but very rarely).
- Put spaces around = to accentuate the distinction between the LHS and RHS.
- Don't put spaces between method names and parens, or parens and paramter lists - they're closely related, so should be close.
- Use spaces to accentuate operator precedence, e.g., no space between unary operators and their operands, space between binary operators and their operands.
- Don't try to horizontally align lists of assignments – it draws attention to the wrong thing and can be misleading, e.g., encouraging the reader to read down a column.
- Always indent scopes (classes, methods, blocks).

Team Rules

- Every team should agree on a coding standard and everyone should adhere to it.
- Don’t modify a file just to change the formatting, but if you are modifying it anyway, go ahead and fix the formatting of the code you modify.
- Code formatting standards get religious. My rule: make your code look like the language inventor’s code.
- If the language you’re using has a code convention (like Java’s), use it!