Mohammadmahdi Farrokhy

Clean Code:

A Handbook of Agile Software Craftsmanship

Chapter 05: Formatting

What does formatting exactly mean?

Guidelines for Well-Structured Code: To ensure a stable, readable, and easily understandable code format, you need to adhere to some simple rules. Similar to the grammar of a language, communication between various code parts—methods and properties—should possess an integrated structure. Formatting refers to factors like the horizontal and vertical length of a code file, the spacing between code elements, and instruction density.

Vertical Formatting

Vertical formatting

How large could a source file(class) ever be?

The smaller, the better. Smaller classes are understood easier.

The code should be akin to reading a newspaper. The names should be simple but descriptive. The module name tells the reader the story behind it. The upper parts of the file should talk about the general issues. The more we go down a file, the more details we see.

Vertical formatting

- Vertical Spacing Between Concepts

Each line of code represents a command or condition, and groups of lines represent complete thoughts. The thoughts must be separated using empty line.

Vertical Spacing between concepts

```
private List<Flight> getFlightRows() {
    List<Flight> flightList = new LinkedList<>();
    for (Flight flight : flightListFromFlightDTOList(flightsDTOList))
        flightList.add(readFlightFromTable());
    flightList = flightList.sortFlightsByDepartureTime();
    return flightList;
}
```

```
private List<Flight> getFlightRows() {
    List<Flight> flightList = new LinkedList<>();
    for (Flight flight : flightListFromFlightDTOList(flightsDTOList))
        flightList.add(readFlightFromTable());

    flightList = flightList.sortFlightsByDepartureTime();
    return flightList;
}
```

Vertical formatting

- Vertical Density

Code lines that are related to the same concept should stick together and create a high density of instructions. Avoid interjecting comments between them, especially.

Vertical density

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

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

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

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

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

Vertical formatting

- Vertical Distance

  Related ideas should be close.

  Methods: Similar and linked methods should reside in the same class and near each other. If method A calls method B, B should follow A without gaps. If A calls B, C, and D, they should follow A in that order.

  Class Properties: Declare them at the class's top.

  Local Variables: Their declaration and usage should be close to each other. Since our methods are simple and short, we declare the variables at the top the methods.

Vertical distance

```
public void A{
    B();
    C();
    if (condition)
        D();
}

private void B(){}
private void C(){}
private void D(){}
```

Vertical formatting

- ## Conceptual Dependency

  Elements with stronger conceptual links should be nearby, maintaining a clean and coherent code structure. For instance, methods with similar naming or logic are best declared sequentially.

Conceptual Dependency

```
public class Assert{
   public static void assertTrue(String message, boolean condition){
      if(!condition)
         fail(message);
   }
   public static void assertTrue(boolean condition){
      assertTrue("", condition);
   }
   public static void assertFalse(String message, boolean condition){
      assertTrue(message, !condition);
   }
   public static void assertTrue(boolean condition){
      assertFalse("", condition);
   }
}
```

Vertical formatting

- ## Vertical Sequence

  The called method should be declared under the caller method. The details of the implementation belong to the lower parts of the class. At the top of the class we talk about the general abstractions.

Horizontal Formatting

Horizontal formatting

How long could a line of code be?

Avoid overly long lines; It makes reading the code more difficult. Most IDEs recommend around 120 characters per line and have a line length limit indicator.

Horizontal formatting

- Horizontal Spacing And Density

Horizontal space connects related concepts with a loose relationship.

Assignment Operations: Variable Name + Space + = + Space + Value + ;

Method Arguments: Method Name + (+ First Argument + , + Space + Next Argument + ... +)

Mathematical Equations: Operations with the same priority group together. This might not be followed by IDE auto-formatting systems.

Horizontal Spacing And Density

```
int n = 10;

private void methodName(int number, String name){}

double delta = b*b - 4*a*c;
```

Horizontal formatting

- Horizontal Alignment

Avoid using horizontal alignment. It draws attention to a minor aspect and might distract the reader from noticing the variable's type in assignment operations.

Horizontal Alignment

```
private     String                  firstName;
private     String                  lastName;
private     Address                 address;
private      int                    age;
public      boolean                 isMarried;
protected EmploymentStatus employmentStatus;
```

```
private String firstName;
private String lastName;
private Address address;
private int age;
public boolean isMarried;
protected EmploymentStatus employmentStatus;
```

Horizontal formatting

- Indentation

Maintaining clear indentation is one of the most critical considerations. It significantly enhances code readability. Typically, IDEs automatically apply appropriate indentation based on {} and scope.

Indentation

```
private void setFirstName(String firstName) {this.firstName = firstName;}
```

```
private void setFirstName(String firstName) {
    this.firstName = firstName;
}
```

Formatting

- Team Guidelines

At times, development teams devise their own code formatting rules. However, it's advisable to follow established standard formatting.

Regardless of the chosen format, consistency is key among team members. This ensures that the code doesn't appear as if it's been written by multiple teams.

A strong software system comprises a legible documentation set, and the best document is the code source. Readers should be confident that all system files adhere to the same formatting, allowing them to peruse it like well-crafted literature.