Coding Standards

Golf Course Mapper

Team Recursive Recursion

April 12, 2018

1 File Standards

1.1 Format

- Files are encoded using the **UTF-8** character set.
- Lines should not be longer than **80 columns**.
- **Soft tabs** equating to **4 spaces** should be used.
- Each level of indentation uses **1 tab**.
- Continuation indentation uses **2 tabs**.
- Every file contains a **file header** (described below).

1.2 Headers

Headers are always on the first line of a file and are placed in comments. See section 2 for language-specific header styling. Each header contains the following information:

- Name of the file
- Author of the file
- Name of the class(es) contained within the file (if any)
- Short description of the file

Header files are structured as the following:

```
Filename: File.ext
Author  : John Doe
Class   : SampleClass

    The SampleClass contains many different sample
    methods.
```

2 Language Standards

2.1 Java, C# & TypeScript

2.1.1 Naming Conventions

- **Variables** are named using camel casing. Descriptive names should be used with the exception of counters in loops.
- **Member variables** are named similarly to regular variables with the addition of an *underscore* prefix.
- **Classes** start with a *capital* letter and use camel casing.
- **Functions** are also named similar to regular variables and should be descriptive.
- **Recursive functions** are named similarly to regular functions, but have a `_r` postfix to the name.

```
class SampleClass {

    int _someMember;

    public void myFunction() {
        int someInteger;
    }

    public void myFunction_r() {
        myFunction_r();
    }
}
```

2.1.2 Style

Braces will be styled in the following manner:

- Opening braces are placed on the same line as the header.
- Closing braces are placed on a separate line at the same indentation level as the header.
- Else clauses are placed on the same line as the closing brace.
- While clauses of a do-while are placed on the same line as the closing brace.
- Braces are never left out for one-line loops or conditions.

```
if (condition) {
    statement;
} else {
    statement;
}

while (condition) {
    statement;
}

do {
    statement;
} while (condition);

class SampleClass {
    public void method() {
        statement;
    }
}
```

Continuation lines should end on the operator as to indicate that the line is not complete and has a continuation.

```
double result = example + of + (a * very) /
        long - equation;
```

2.1.3 Comments

File headers are structured according to section 1.2 and are styled as the following:

```
/***
 * Filename: SampleClass.java
 * Author  : John Doe
 * Class   : SampleClass
 *
 *      The SampleClass contains many different sample
 *      methods.
 ***/
```

Function headers are provided for every function and take the following form (the description can be omitted in the case of simple functions, such as mutators & accessors).

```
/***
 * function(ParType1, ParType2) : ReturnType
 *
 *      Description of the function.
```

```
 ***/
ReturnType function(ParType1 p1, ParType2 p2) {
    ...
}


/***
 * SampleClass(ParType1, ParType2) <<constructor>>
 *
 *     Description of the constructor.
 ***/
SampleClass(ParType1 p1, ParType2 p2) {
    ...
}
```

Inline comments should be kept to a minimum, since code should be self-documenting. Only use inline comments in the case of code that may be difficult to understand.

2.2 HTML & CSS

2.2.1 Style

CSS should be styled as follows:

```
selectors {
    some-attribute : style;
    another-attribute : style;
}

more {
    some-attribute : style;
}
```

HTML style follows the following rules:

- Opening and closing tags of **block** elements should be kept on their own lines, with the content indented.
- Opening and closing tags of **inline** elements should be kept on the same line, with the content between the tags.

```
<div>
    <p>
        Here is some <span class="red">red</span> text!
    </p>
</div>
```

2.2.2 Comments

File headers are structured according to section 1.2 and are styled for CSS and HTML, respectively, as the following:

```
/***
 * Filename: style.css
 * Author  : John Doe
 *
 *      The styling for some example page is contained
 *      here and applies a material style.
 ***/
```

```
<!--
    Filename: page.html
    Author  : John Doe

        The page displays some content.
-->
```

3 Repositories Structure

3.1 Git Repositories

There are four (4) repositories that are used for the project: `web-app`, `mobile-app`, `mapper-api` and `documentation`.

- `web-app`

  The web application used for mapping golf courses.

- `mobile-app`

  The Android application used for viewing golf courses.

- `mapper-api`

  The API used to access the database.

- `documentation`

  All documentation relating to the project, including this file.

3.2 File Structure

Each repository has a `README.md`, `LICENSE.txt` and a `.gitignore` file in the root of the repository.

`mapper-api` and `web-app` contain a folder that contains the source code of the project. The folder is organised according to the standard template of *ASP.NET Core* web applications.

`mobile-app` contains a folder that contains the source code of the project. The folder is organised according to the standard template of *Android Studio* projects.

`documentation` contains a folder for each of the four (4) documents. These folders contain the `.tex` source files of the documents, as well as any auxiliary files that are needed by the document. In addition, there is a `publish` folder that contains the final PDF versions of each document, and a `other` folder that contains extra documents that do not form part of one of the four (4) main documents.