University of Pretoria
Software Engineering - COS 301

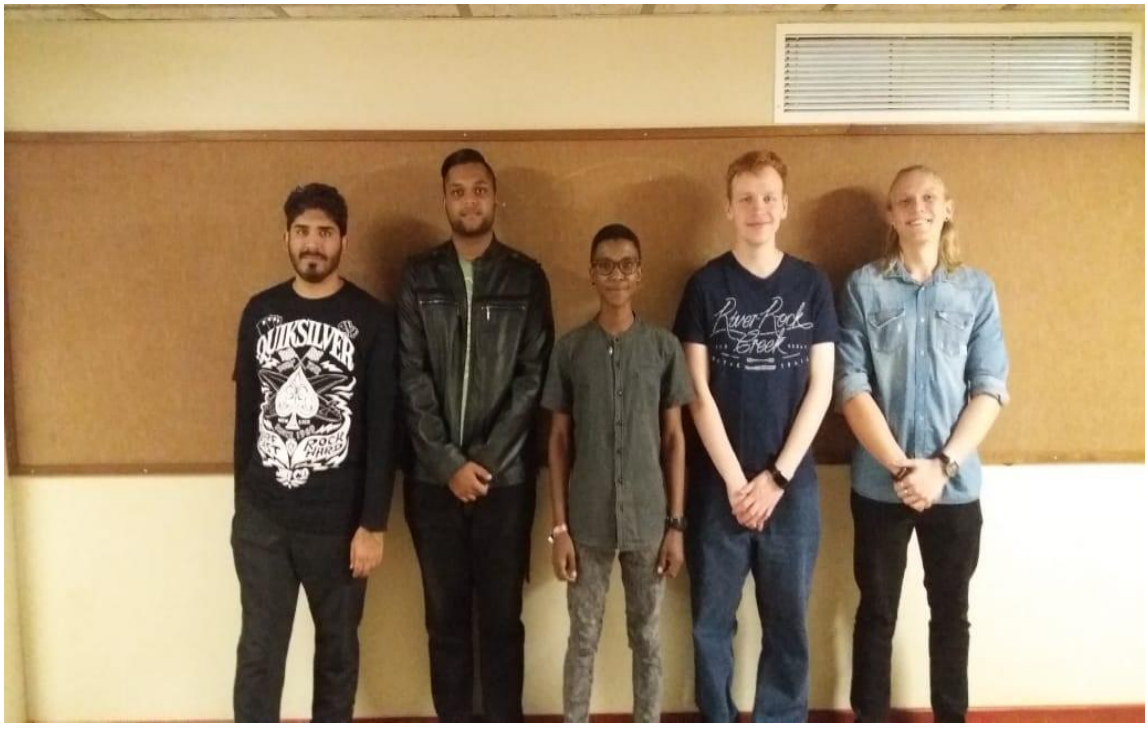
Defendr Coding Standard

Dark nITes

12 October 2019

Team members:

MI (Muhammed) Carrim	R (Ruslynn) Appana	SNL (Sisa) Khoza	CP (Christiaan) Opperman	J (Jeandre) Botha



Contents

1. Introduction

The aim of this document is to optimise and realise improvements in 4 key criteria:

- Readability: the ease of understandability of the code to human eyes
- Writability: the speed and efficiency one can produce a software solution
- Reliability: the evaluation of closeness to expected performance
- Cost: the overall financial implications when producing a software solution

2. Whitespace

Whitespace is to appear before, but not necessarily after, opening parenthesis.

3. Tab characters

Tab characters are to be 4 characters long.

4. Indentation

Indentations will be 1 tab space (4 whitespace charater) long, and K&R or 1TBPS (1 true brace placement style) brace placement will be used.

```
for (i = 0; i < num_elements; i++){
        foo[i] = foo[i] + 43;

        if (foo[i] < 35){
                printf ("Foo!");
                foo[i]--;
        } else {
                printf ("Bar!");
                foo[i]++;
        }
}
```

5. Brace placement

Curly braces should not be used for blocks containing single statements

6. Line width

Line widths of 80-120 characters long is the preferred width. If a continuation line is not indented, indent it one tab space (four spaces).

7. Naming conventions

- If a *using [namespace]* directive is not included, use namespace qualifications.
- If a namespace is imported by default, qualification is not necessary.
- If a qualified name is too long for a single line, it can be broken after a dot(.)
- Underscore separated names should be used if a name is more than one word long

```
var current_performance_counter_category = new System.Diagnostics.
    PerformanceCounterCategory();
```

8. Layout conventions

The following layout will emphasise the structure of code, as well as to make code easier to read:

- Write only one statement per line
- Write only one declaration per line
- If continuation lines are not indented, indent them one tab space (four characters)

9. Commenting conventions

Comments in code should follow this structure:

- Place the comment on the a separate line from the code, preferably before the code
- Begin comments with an uppercase character
- End the comment with a full stop dot(.)
- Insert one character space between the comment delimiter and the text of the comment

The following structures are to be used with single-line and multi-line comments respectively:

```
// The following declaration creates a query.  It does not execute or run any further
// queries.
```

```
/**
* The following declaration creates a query.
* Execution of the query is delegated the execution handler.
*/
```

10. Function names and return values

The declaration of functions names will use the GNU; all lowercase characters, with words being delimited with an underscore (“_”).

11. Syntax conventions

11.1. String data types

- When concatenating strings, make use of string interpolation
- When appending strings using loops, or when working with large strings, employ the use of a StringBuilder object

11.2. Implicitly typed local variables

- If the type of a local variable can be inferred from the assignment or the use of a precise type is not important, then implicit typing may be used

```
var var_1 = "This is a string";
var var_2 = 32;
var var_3 = Convert.ToInt32(Console.ReadLine());
```

```
var var_4 = ExampleClass.ResultSoFar ();
```

- Do not make use of *var* when the type is not easy to infer from the assignment
- Do not expect the variable name to specify the type of the variable
- The use of implicitly typed variables is recommended for the loop variables in *for* and *foreach* loops

11.3. Signed and Unsigned data types

```
var inputInt = Console.ReaderLine();
Console.WriteLine(inputInt);
```

```
var syllable = "ha"
var laugh = new StringBuilder();

for (var i = 0; i < 1000; i++){
        laugh.Append(syllable);
}

Console.WriteLine(laugh);
```

For general use, and if precision is not specified, use *int* instead of unsigned types. *Int* is commonly used throughout and interacts easier with the use of libraries.

11.4. Exception handling and try-catch statements

Use a *try-catch* statement for most exception handling

```
try{
        return array[index]
} catch (System.IndexOutOfRangeException e){
        Console.WriteLine("Index out of range: {0}", index);
        throw;
}
```

11.5. && and || comparison operators

```
static string get_value_from_array (string[] array, int index){
        try{
                return array[index];
        } catch (System.IndexOutOfRangeException e){
                Console.WriteLine("Index is out of range: {0}", index);
                throw;
        }

}
```

When performing comparisons, use the && and || operators (instead of & and | bit-level operators) to increase performance by skipping unnecessary comparisons.

```
if ( (divisor != 0) && (dividend/divisor > 0)){
        Console.WriteLine("Quotient: {0}", dividend/divisor);
} else {
        Console.WriteLine("Attempt to perform division by 0 is invalid");
}
```

11.6. New operator

Use the concise form of object instantiation, with implicit typing, as shown in the following declaration

```
var instance_1 = new ExampleClass();
```

is the equivalent of

```
ExampleClass instance_2 = new ExampleClass();
```

11.7. Static members

When access to a static member is required, call said static member by using the class name as well. This will ensure static accesses are clear, and makes code more readable. Should the member be defined in a base class, do not qualify said access with the name of a derived class.

```
var var_1 = ExampleClass.StaticMember;
```

12. Naming of UI components

12.1. Formatting

Components names must be prefixed with the abbreviated component type followed by the use of the component.

```
self.txt_Email = tk.Entry(self.Frame1)
```

12.2. Abbreviated component type list

Abbreviated list	**Full name**
lbl	Labels
btn	Button
txt	Textbox/Entry
img	Images
lst	List box
txtb	Text

12.3. Button click event function

When a button is clicked. It will call the command lambda followed by the previously defined functions.

```
self.btn_Cancel.configure(command=lambda: self.cancel_Login())
```

12.4. Fonts

Font names must be prefixed font followed by the use of the font.

```
fnt_Text = "-family {DejaVu Sans} -size 15 -weight normal -slant roman -underline 0 -overstrike 0"
```