Coding Standard for Swift Programs

Coding standards define a programming style to be used by software developers writing in a particular language. It is essentially a set of rules and guidelines for the formatting of source code in that language. The intent is to make the code easier to understand by other developers in the programmer's community, much like the way a grocery store chain may have the same floor plan for all of its stores, so that customers visiting one of their stores in a different city feel an immediate sense of familiarity and can quickly find what they're looking for.

Although code that does not follow the standard is technically not "wrong", in the sense that the program will still behave the way the developer intended, it has not been written in such a way that others in the community can quickly recognize the structure of the code and understand how it works.

General

1. Any new code should be consistent with the default code format generated by Xcode, except where noted below.

2. Any code added to existing code should be formatted to reflect the existing standard in the code already written.

3. Error checking and input validation are expected. Common-sense rules include (but are not limited to):

   - cases when the user fails to enter data (such as empty text fields).
   - cases when the user enters inappropriate data types (such as entering a word or a floating-point number in a text field when an integer is expected).
   - the possibility of an attempting division by zero.
   - the possibility of an index out of range for an array or tuple.

   Note: this is the default expectation. Some assignments may specify additional requirements for error checking and input validation behavior, and other assignments may waive requirements.

Naming

Variable Names: camelCased, starting with a lowercase character
Example: `speedOfCar`

Property Names: camelCased, starting with a lowercase character
Example: `speedOfCar`

Class Names: camelCased, starting with an uppercase character

Example: `Automobile`

Method Names: camelCased, starting with a lowercase character
Example: `speedUp`

Blank lines

There should be no more than one blank line between any two lines of code.

Acceptable:

```
line of code
<blank line>
line of code
```

Acceptable:

```
line of code
line of code
```

Unacceptable:

```
line of code
<blank line>
<blank line>
line of code
```

Comments

1. Each commented line should start with two forward-leaning slashes.
2. Do not use multi-line comment characters - /* comment */
3. A comment at the end of a line is ok.

Acceptable:

```
// comment line 1
// comment line 2
```

Acceptable:

```
int total = 50;   // initialized to 50 because blah
```

Unacceptable:

```
/* comment line 1
   comment line 2  */
```

Assignment statements

There should be one space, and only one space, on either side of the assignment operator.

Acceptable:

```
total = 0;
```

Unacceptable:

```
total=0;
total= 0
total =0;
total     =      0;
```

Indentation

Every scope should have a new indentation level.

Acceptable:

```
if (<some condition>) {
    if (<some condition>) {
        —yada—
    }
}
```

Unacceptable:

```
if (<some condition>) {
if (<some condition>) {
     —yada—
}
}
```

Braces

Opening braces

You can have the opening brace either immediately following or on the next line. But you MUST be consistent in all your code for a given homework or project.

Closing braces

These should always be on their own line and either lined up with the starting statement or the opening brace - depending on where you placed the opening brace.

Acceptable:

```
-(instancetype) init
{
    …statements…
}

-(instancetype) init {
    …statements…
}

if (<some condition>)
{
       —yada—
}


if (<some condition>) {
       —yada—
}
```

Unacceptable:

```
-(instancetype) init
{
    …statements…
   }

-(instancetype) init {
    …statements…
    }

if (<some condition>)
{
       —yada—
  }


if (<some condition>) {
       —yada—
  }
```

Spaces

There should generally be no more than one space between any two characters.

Acceptable:

```
int total = 0;
```

Unacceptable:

```
int     total         = 0;
```

Property and Variable Declarations

A given property or variable declaration must be for one and only one property or variable.

Acceptable:

```
var firstName:String
var lastName:String
```

Unacceptable:

```
var firstName:String, lastName:String
```

Class definitions

Each class definition should be defined in its own .swift file. That is, a source file should contain the definition of one and only one class.

**** End of Coding Standard ****