



# Style Checklist

Style checklist for CS406: Data Structures<sup>1</sup>.

1. Variable identifiers begin with a lowercase letter.

```
int count
```

2. Run the words in multiword identifiers together. The 2nd and subsequent words begin with a capital letter (“camel”-case).

```
parenthesesCounter  
ageEmployee  
countStudents
```

3. Use a blank lines to logically group your program statements. Comment your code accordingly.

```
// read the data set from the supplied file  
int[] xs = readFromFile("data.txt");  
  
// calculate the sum of data set  
int sum = 0;  
for(int i=0; i<xs.length; i++)  
    sum += xs[i];  
  
System.out.println(sum);    // display the sum
```

4. Method names begin with a lowercase letter.

```
sort()  
readFromFile()
```

5. Class and Enum names begin with an uppercase letter. The same applies to Interface names.

```
Account  
Point2D
```

6. Constant identifiers are coded in all uppercase.

---

<sup>1</sup>Based somewhat on Google's guide: <https://google.github.io/styleguide/javaguide.html>

```
public static final double PI = 3.142;
public static final int MAX_SIZE_NAME = 30;

public enum Colour { RED, GREEN, BLUE};
```

7. Code multiword identifiers for constants by separating the words with underlines.

```
PI_SQUARED
```

8. Type parameter identifiers are written as a single uppercase character. Ex: R, S and T in

```
List<T>
Map<R, S>
```

9. Begin each class with a very brief description of what the class does (not of how the class works.) Use javadoc.

Writing a class description that is wrong or misleading is even worse than not writing a description at all.

10. Begin each public method with a very brief description. Possible exceptions are getters and setters and methods annotated with `@Override`. Use javadoc.

Include any preconditions and post-conditions.

11. Code only one declaration per line (encourages commenting).

```
int i, j; // no

int i;
int j;
```

12. Code meaningful identifiers for variables, constants, types, and functions.

```
closingSymbol
PI
NodeType
sort()
```

Do not use long vague identifiers such as `subscript`. Choosing a misleading identifier name is even worse than coding a meaningless name.

When an identifier is not meaningful (ex: `i`) add a comment that describes the identifier's purpose.

```
int i;    // index to `client` array
```

13. Do not use “magic numbers and String literals”. Use constants instead.

```
boolean classTooSmall = students > 30;           // bad
boolean classTooSmall = students > MAX_CLASSROOM_SIZE; // good
```

14. Avoid initializing fields when declaring them, i.e.: outside of a constructor.

```
class Point {

    int x = 0;
    int y = 0;

    Point() {}
}
```

vs.

```
class Point {

    int x;
    int y;

    Point() {
        x = 0;
        y = 0;
    }
}
```

15. Class members are grouped together in the following order:

- (a) constants,
- (b) fields,
- (c) constructors, and
- (d) methods.

16. Method and constructor overloads are grouped together in the class.

17. Fields are private.

18. Methods and constructors are always declared public or private

19. Static fields that behave like global variables are prohibited (unless they are declared `final`, i.e.: constants).