# ITC205 Code Style Guidelines.

The code style guidelines listed below are arbitrary. They are intended purely to generate code changes which will result in conflicts that need to be resolved in version control, and standards which can form the basis of a static code review.

Please Note: these guidelines are NOT being put forward as enforcing ‘good’ coding practices. If you wish to add additional guidelines to these to improve code quality, please do so with the following provisos:

1. Any additional guidelines must be in addition to the stated guidelines. You may not substitute guidelines and you may not introduce guidelines that state that the provided guidelines can be ignored.
2. You MUST state your additional guidelines and provide your complete modified guidelines as part of your code repository for the assessment.
3. Your modified guidelines MUST be applied consistently across all code.
4. You MUST explicitly review all code against your modified guidelines during the static code reviews.

The following guidelines MUST be enforced:

1. Final class, method, and variable names MUST conform with the specification given in the accompanying class diagram. (Also, see below)
2. All variable names are to be meaningful. Variable names should be nouns or noun phrases.

All variable names are to start with a lowercase letter and to be in camelBack. Underscore\_separated variable names or names starting with a capital are not acceptable. E.g.:

int bookId;

not

int BookID;

nor

int bid;

nor

int book\_id;

1. All method names are to be meaningful. Method names should be verbs or verb phrases. E.g. isAvailable() rather than available().

All method names are to start with a lowercase letter and to be in camelBack. Underscore\_separated method names or names starting with a capital are not acceptable.

E.g.:

ArrayList<Book> books = library.getBooks();

not

ArrayList<Book> books = library.books();

nor

ArrayList<Book> books = library.Get\_Books();

nor

ArrayList<Book> books = library.get\_Books();

1. All class (and enum) names are to be meaningful. Class (and enum) names should be nouns or noun phrases.

All class (and enum) names are to start with an uppercase letter and to be in CamelBack. Underscore\_separated class (and enum) names or names starting with a lowercase letter are not acceptable.

E.g.:

public class Library {

not

public class library {

and

private enum ControlState {INITIALISED, READY, RESTRICTED};

not

private enum CONTROL\_STATE {INITIALISED, READY, RESTRICTED};

1. The names of variables used as constants (static final variables) should be meaningful, all upper case, and underscore separated. E.g.:

static final int LOAN\_LIMIT = 3;

Not

static final int LOANLIMIT = 3;

Nor

static final int loanLimit = 3;

1. All variables and operators MUST be separated by white space. E.g.:

a = b + c + d;

not

a=b+c+d;

1. Anywhere curly brackets MAY be used, they MUST be used. E.g.

if (condition) {

some single line of optional code;

}

further non optional code;

Not

if (condition)

some single line of optional code;

further non optional code;

1. There MUST be a SINGLE consistent bracketing style used throughout the code: Either:

if (condition) {

some single line of optional code;

}

further non optional code;

or

if (condition)

{

some single line of optional code;

}

further non optional code;

but not both, even in different files

1. Indentation increments MUST be 4 spaces for every level of nesting. Not tabs, not 8 spaces, not 2 spaces; 4 spaces.
2. The argument list of any method calls MUST NOT contain any other method calls. E.g.

someVar = someMethod(x, y, z);

someOtherVar = someOtherMethod(someVar, p, q, r);

Not

someOtherVar = someOtherMethod(someMethod(x, y, z), p, q, r);

1. Argument lists to method calls should start immediately after the opening bracket, be separated by whitespace, and the closing bracket should follow the last argument without whitespace. E.g.:

someMethod(x, y, z);

Not

someMethod(x,y,z);

Nor

someMethod( x, y, z );

Nor

someMethod( x,y,z );

1. Singletons MUST return their sole instance through a method called getInstance() E.g.

Library library = Library.getInstance();

