# **MTA Coding Standards**

This document provides coding conventions for developing automated test scripts under Mobile Test Automation lab

1. **Eclipse Project entity naming conventions**

Project Names should follow the syntax ‘PRJ<XXXXXX\_SectionName>’

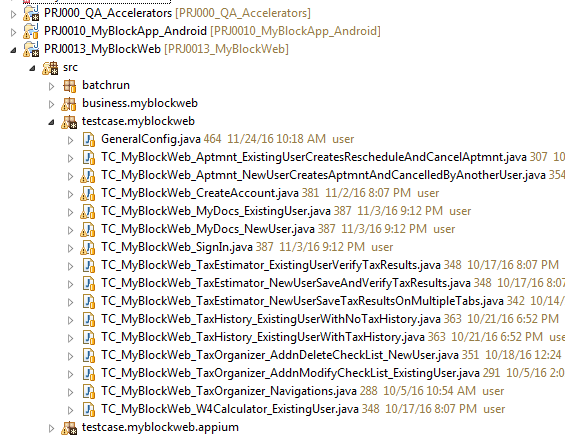
* Business logic classes are placed under a package with a syntax

‘business.<SectionName>’

* Testcase classes are created under the package

‘testcase.<SectionName>’

* Testcase classes should prefix with ‘TC\_’



1. **Beginning Comments**

All source files should begin with a c-style comment that lists the class name, version

information, date, and copyright notice:

1. **Package and Import Statements**

The first non-comment line of most Java source files is a package statement. After that,

import statements can follow. For example:

package java.awt;

import java.awt.peer.CanvasPeer;

1. **Intendation**

Four spaces should be used as the unit of indentation. The exact construction of the indentation

(spaces vs. tabs) is unspecified. Tabs must be set exactly every 8 spaces (not 4).

1. **Comments**
   1. Block Comments

A block comment should be preceded by a blank line to set it apart from the rest of the code.

/\*

\* Here is a block comment.

\*/

1. Single-Line Comments

A single-line comment should be preceded by a blank line. Here’s an example

of a single-line comment in Java code:

if (condition) {

/\* Handle the condition. \*/

...

}

1. Trailing Comments

Here’s an example of a trailing comment in Java code:

if (a == 2) {

return TRUE; /\* special case \*/

} else {

return isPrime(a); /\* works only for odd a \*/

}

1. **Initialization**

Try to initialize local variables where they’re declared. The only reason not to initialize a

variable where it’s declared is if the initial value depends on some computation occurring first.

1. **Class and Interface Declarations**

When coding Java classes and interfaces, the following formatting rules should be followed:

* No space between a method name and the parenthesis “(“ starting its parameter list
* Open brace “{” appears at the end of the same line as the declaration statement
* Closing brace “}” starts a line by itself indented to match its corresponding opening

statement, except when it is a null statement the “}” should appear immediately after the “{“

class Sample extends Object {

int ivar1;

int ivar2;

Sample(int i, int j) {

ivar1 = i;

ivar2 = j;

}

int emptyMethod() {}

...

}

* Methods are separated by a blank line

1. **Statements**
   1. **Simple Statements**

Each line should contain at most one statement. Example:

argv++; // Correct

argc++; // Correct

argv++; argc--; // AVOID!

* 1. **Compound Statements**

Compound statements are statements that contain lists of statements enclosed in braces

“{ statements }”. See the following sections for examples.

* + The enclosed statements should be indented one more level than the compound statement.
  + The opening brace should be at the end of the line that begins the compound statement; the closing brace should begin a line and be indented to the beginning of the compound

statement.

* + Braces are used around all statements, even single statements, when they are part of a

control structure, such as a if-else or for statement. This makes it easier to add

statements without accidentally introducing bugs due to forgetting to add braces.

* 1. **Return Statements**

A return statement with a value should not use parentheses unless they make the

return value more obvious in some way. Example:

return;

return myDisk.size();

return (size ? size : defaultSize);

* 1. **Return Statements**

The if-else class of statements should have the following form:

if (condition) {

statements;

}

if (condition) {

statements;

} else {

statements;

}

if (condition) {

statements;

} else if (condition) {

statements;

} else {

statements;

}

* 1. **Return Statements**

A for statement should have the following form:

for (initialization; condition; update) {

statements;

}

An empty for statement (one in which all the work is done in the initialization, condition, and update clauses) should have the following form:

for (initialization; condition; update);

* 1. **do-while Statements**

A do-while statement should have the following form:

do {

statements;

} while (condition);

* 1. **switch Statements**

A switch statement should have the following form:

switch (condition) {

case ABC:

statements;

/\* falls through \*/

case DEF:

statements;

break;

case XYZ:

statements;

break;

default:

statements;

break;

}

Every time a case falls through (doesn’t include a break statement), add a comment where the break statement would normally be. This is shown in the preceding code example with the

/\* falls through \*/ comment.

Every switch statement should include a default case. The break in the default case is

redundant, but it prevents a fall-through error if later another case is added.

* 1. **try-catch Statements**

A try-catch statement should have the following format:

try {

statements;

} catch (ExceptionClass e) {

statements;

}

A try-catch statement may also be followed by finally, which executes regardless of whether or not the try block has completed successfully.

try {

statements;

} catch (ExceptionClass e) {

statements;

} finally {

statements;

}