Coding Conventions

Welcome to the TheRiskConquest

Coding conventions-

1) Naming Conventions:

for Classes: Class names must start with the Capital Letters. In case, If class names has multiple words, inner words must start with capital Letters

Example- String, StringBuffer

Using meaningful names for global variables instead of just cryptic abbreviations makes the code easier to read and understanding. When variable names contains more than two words, we should use camel case.

Example- name instead of n and computerScience instead of cs.

Interface names start with Capital letters whereas package names start with lower case letters.

Static Varaiables - Every subsequent letters are capitalized and every other word is separated with an underscore sign \_ .

Example- NUM\_GEARS = 10

2) Testing Conventions

Follow the rules mentioned below for testing.

Testing class will contain suffix as Test.

For ex - Consider a class BankImplementation,

Now its corresponding test class would contain test as suffix in the class name mandatorily.

BankImplementationTest

Also, methods will end with test. For Example, corresponding test method borrowItem() shall be written as borrowItemTest() method. @Test annotation is included before the test method. Testing makes no sense

without a single assert class. So we use one of the available asser classes - assertEquals, assertTrue, assertFalse,assertNull,assertNotNull, assertSame, asserNotSame, Fail or assertArrayEquals.

3) Opening parentheses should be placed at the end of the same line as the declaration statement whereas the closing parentheses } starts a line indented to match its corresponding opening statement.

Example-

class Player extends Object {

int noOfPlayers;

Player(int p) {

noOfPlayers = p;

}

Using // for comments after a variable to explain it

Example- dateOfBirth; // date of birth in mm/dd/yyyy format

4. :Use block comments for method description (i.e /\*\* \*/ )

Example-

/\*

\* Here is a block comment.

\*/

5. Indentation -

Using proper indentation makes the program more human readable and gives an indication of scope without affecting the program directly.

A. if (a < 10 && b < 20 || c < 60) { return true; } else {System.out.println (“incorrect”) ; return false; } // Not suggested

B. if (a < 10 && b < 20 || c < 60) // correct way

{

return true;

}

else

{

System.out.println(“incorrect”);

return false;

}