Code conventions cheatsheet

The full set of Java code conventions, as documented by Sun Microsystems, is available at `http://java.sun.com/docs/codeconv/`

1 Classes

Class names should start with a capital letter, and should have a capital letter for the start of every new word in the name, e.g. `ThisIsAnExampleOfClassCapitalisation`. The file should have the exact same name, down to the capitalisation, i.e. `ThisIsAnExampleOfClassCapitalisation.java` - otherwise it will not compile. Class names should be reasonably concise and reflect what the class represents or does - e.g. `Bicycle.java` should represent a bicycle, whilst `SpeedCalculator.java` should be a class to calculate the speed of something. You should typically have one file for each class you write - multiple classes per file limits the reusability of the code.

2 Variables

Variable names should begin with a lowercase letter, and then have the first letter of each subsequent word capitalised, as with class names. For example, `thisWouldBeAWellCapitalisedVariableName`. You should also try and give variables concise but explicit names - the name `inputText` is much easier to read than `theInputTextIGotFromTheCommandLine` or `t` when buried in the middle of a code block. Variables should almost always be declared private, and have accessor (getter) and mutator (setter) methods in place to access them (if appropriate). They should generally be declared at the top of the code block in which they are used.

3 Methods

As with variable names, methods should start with a lowercase letter and capitalise the first letter of every following word. You should always prefix method declarations with either public, protected, or private. Methods should generally start with a verb or action, and describe what they do, e.g. `getMail()`, `calculateTax()`. Methods should be internally consistent - that is, they should not rely on anything else being called first. You should also avoid using `println()`statements in anything except a dedicated print method or main method, unless it is for testing purposes - in which case you should comment them out before submission of the code.

4 Comments

You should comment the start of your class and every method to specify what it does. You should also add comments to complicated blocks of code to explain what is happening in them. See Example 1 for appropriate comment formatting - note that in practice, calculateValue() would **not** need that much commenting.

5 Formatting

When adding a new code block, signified by parentheses , you should indent your code such that each level of parentheses indents one step further. For instance, the two classes given in Examples 1 and 2 are functionally identical, but the first is much easier to read and understand than the second.

Example 1

```
/**
 *This is a test class for storing and returning a number
 */
public class Test{

    private int number;

    /**
     *Constructs an instance of Test storing the specified number
     */
    public Test(int n){
        number = n;
    }

    /**
     *Returns the number stored in this instance of Test
     */
    public int getNumber(){
        return number;
    }

    /**
     *Calculates the sum of every integer from 1 to number (exclusive),
     *except those divisible by 5
     */
    public int calculateSum(){

        //We start with a sum of 0
        int sum = 0;

        /*In this loop, sum will store the total of each i that is not
         *divisible by 5. It will terminate when i reaches number
         */
        for(int i = 1; i < number; i++){

            //If i is not divisible by 5, add it to the sum
            if(i % 5 != 0){
                sum = sum + i;
            }
        }

        //sum now contains the total of all values not divisible by 5
        return sum;
    }
}
```

Example 2

```
public class Test{
private int x;
Test(int n){
x=n;}
int number(){
return x;}
int sum(){
int s=0;
for(int i=1; i<x; i++)
if(i % 5 == 0) continue;
else s=s+i;
return s;
}
}
```