Coding Style Guide

In real-world software development, it is paramount to create readable and easily maintainable code. That is typically achieved through the use of style and commenting guidelines. Python is a language that forces good indentation practices, while Java does not. Therefore, we expect you to follow the following style and commenting conventions. We have also provided an Eclipse preferences file that sets up many of these formats automatically and will adjust your code using the `Format` command in the `Source` menu. (See the tutorial at the end of this document for more information.)

- **Filename:** The filename must match the containing class name and be `TitleCased`.
- **Casing:** Class names must be `TitleCased`; variable, field, and method names must be `camelCased`; constants should be `ALL_CAPS`.
- **Indentation:** Code blocks must be indented **4** spaces using "soft tabs" (specifically, no hard `tab` characters). Our style file will automatically update Eclipse so that the `tab` key on your keyboard inserts 4 spaces automatically.
- **Curly braces:** Curly braces must open at the end of the line preceding the code block and end on a line of their own unless followed by an `else` or `catch` statement, such as:

```
if (true) {
    // do something
}

if (test) {
    // do something
} else {
    // do something else
}
```

- **Header:** All files must contain the following heading, matching the homework number, your name, your userid, and listing any sources to be cited. Please remember the academic integrity policy: do not share, view, or copy any other student code!

```
/**
 * Homework N
 * Your Name, mst3k
 *
 * Sources: TA office hours, Big Java book, ...
 */
```

- **Method and Field Comments:** Each field and method must be commented with information about that field or method, using the format below. You must also include the attribute information for parameters, `@param`, and return values, `@return` as shown.

```
/**
 * Short description of the field title
 */
private String title;

/**
 * Brief description of the method and what it should be
 * doing.
 *
 * @param x Describe what x is
 * @param y Describe what y is
 * @return What gets returned: what does this boolean mean?
 */
public boolean methodName(int x, float y) {...}
```

- **Inline Comments:** If there is a portion of your code that is difficult to understand, or you've received (collaboration policy appropriate) help on a portion of code, you should describe that in comments within your code.

```
// Check that the new title is not null (edge case)
if (newTitle != null) {   // Asked TA about edge cases
    this.title = newTitle;
}
```

Sample File: Book.java

```java
/**
 * Homework 2
 * Your Name, mst3k
 *
 * Sources: TA office hours, Big Java book
 */

public class Book extends OtherClass implements Comparable<ExampleClass> {

    /**
     * Holds the title of the book
     */
    private String title;

    /**
     * Constant containing max number of pages
     */
    public final int MAX_NUMBER_PAGES = 1000;

    /**
     * Set the title of the book
     *
     * @param newTitle The updated title of the book
     */
    public void setTitle(String newTitle) {
        // Check that the new title is not null (edge case)
        if (newTitle != null) {   // Asked TA about edge cases
            this.title = newTitle;
        }
    }

    /**
     * Count to maximum number of pages while printing out the
     * page number given.  Returns true if the given page is less
     * than the limit.
     *
     * @param pageNumber The page number to print
     * @return Whether or not the page number is within the limit
     */
    public boolean printPageNumber(int pageNumber) {
        for (int i = 0; i < MAX_NUMBER_PAGES; i++) {
            if (pageNumber == i) {
                System.out.println(i);
            }
        }
        if (pageNumber <= MAX_NUMBER_PAGES) {
            return true;
        }
        return false;
    }
}
```


BY NC SA