MISCELLANEOUS JAVA TOPICS

JAVADOC

Algonquin Documentation Standards

File Header

The file comments must contain the following information:

- **File name**: [*YourMasterpiece.java*]
- **Author:** [*Student name, ID#*]
- **Course:** CST8132 – OOP
- **Assignment:** [*number*]
- **Date:** [*the date of the final version of the file*]
- **Professor:** [*place here the name of your lab professor*]
- **Purpose:** [*brief description of the contents*]
- **Class list:** [*include this only if there is more than one class in the file*]

Java Documentation Standards

Class Doc Comments

Each class definitions must have a ***Java doc comment (starts with /** ends with */)*** containing the following information:

- [Brief description of the purpose of the class]
- **@author** [*your name if you wrote the class*]
- **@version** [*version number*]
- **@see** [*package name or a class name used in the class– for example, java.io, java.lang.String*]
- **@since** [*the version number of the Java compiler which was used to compile the class*]

Java Documentation Standards

Method Doc Comments

Each of the class method definitions must have a ***Java doc comment*** containing at least the

- following information:
- [Brief description of the purpose of the method]
- **@param** [*name description*] – one per parameter and only if there are parameters
- **@return** [*type description*] – used only if there is a return type different from void

Java Documentation Standards

Field Doc Comments

Each of the class fields (data members) must have a ***Java doc comment***, which describes

- briefly the purpose of the field (variable). If the field is final, the **{@value}** tag must be used.

Java Documentation Standards

Implementation Conventions and Comments (non-javadoc)

- All local variables must be commented. The comment must explain the use of the variable.
- All function segments must be commented (a segment is a sequence of related statements (i.e. loops, switches, if-else ladders and sequence of linear statements performing some distinctive task.)
- Each important line of code must be commented. A student should use their judgment to decide whether the line is important. Important lines: testing some special conditions; complex calculations; conversions and so on. Do not overly comment your programs – do NOT comment every line of code.
- The coding style and the naming should follow the Java coding conventions.

Generating JavaDoc documentation

- JavaDocs will only be generated based on comments within the /** */ frame.

<u>From command line:</u>

- After you have completed all your testing, and are ready to submit your assignment, create a folder named **Assign4**. This will be your deployment folder. Inside create a folder named **docs***.*
- Copy all **Assign4** related ***.java*** files into the **docs** folder. All your files must have a package statement.
- Open a command (DOS) window, make the **docs** folder current, and run the *javadoc* utility with the following options:

```
javadoc -author -version -private  *.java
```

- If *javadoc* generates some warnings or errors related to the *@see* tags which are referring to the Java API classes, ignore them. Pay attention to all other errors and warnings. Correct the errors and generate the documentation again.
- Check the documentation for correctness and completeness. Correct and generate again.
- Delete the *.java* files from the **docs** folder.

Generating JavaDoc documentation

- JavaDocs will only be generated based on comments within the /** */ frame.

Eclipse:

- To generate the documentation with Eclipse you have to use the *File>Export>Java>Javadoc* option. The *Javadoc command* must point to javadoc.exe. For example

  C:\Program Files\Java\jdk1.8.0_20\bin\javadoc.exe
- You must also select the *Private* option of *Create Javadoc for members with visibility:*

- Reference for JavaDoc: http://media.pearsoncmg.com/ph/esm/deitel/javahtp_8/WebAppendices/jhtp8_appM_UsingJavadoc.pdf

  (you need to register to the site first…..)