Software Development Notes and FAQ

Java Formatting Standards

See this document for full details: Java Formatting Standards
You should already be applying all of these aspects into your current programming.

Maven & POM Dependency Modifications

There have been a couple of modifications to the pom.xml file which is needed either for the testing framework or for platform compatibility. Below I have outlined the dependencies and the changes made, as well as if you need to add or modify anything in the pom in the directory you are working on.

JavaFX Version Change

Due to the current version of JavaFX we are using not working on MAC, we have modified the javafx version to be `17.0.10` rather than originally using `17.0.6`, you should just need to change the number in all sections of the pom.

JUnit Version Change

We have now updated to using JUnit 5 over JUnit 4 to provide better functionality.
We are now using version `5.10.2` rather than `4.13.2`. Please update all version occurrences.

JUnit extra dependency

Due to the way JUnit works with Maven, there is an extra plugin required. Please add in the Surefire plugin to your pom.xml if it isn't already in there. (In the appropriate place i.e. within the <plugins> tags) (p.s. Version 2.22.2 may be required rather than 3.2.5 - needs checking to confirm)

```
<plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.2.5</version>
</plugin>
```

Running JUnit Tests

In order to run the JUnit tests, you need to run them via maven, by running the following command. If using eclipse you can do this by right clicking on your current project (in the left sidebar). Find "Run As", then click "Maven Build…" (note the … here).

Now a new window should popup, and enter the following into the “goal” text box. Everything else is fine as it is.

```
clean test
```

For running the program again after doing this the first time, you can click the little drop down arrow next to the “run” button at the top, and there will be a list of recent build commands. (The maven one you did above, should be the top one in that list)

Git and GitHub Reminder

Please remember at the end of every session where you are adding info to any files, make sure to commit them to the branch you are working on and push them to the origin (GitHub) so that others wanting to merge or view anything you have done will see the most up to date version.
Additionally, when working with the libraries, the structure we are using is as follows:

`Lib-Dev` - The main development branch (includes both implementations and tests)

`Lib-Implementation` - The implementation for the methods in the library

`Lib-Testing` - The branch for all tests of the library methods

Lib-Implementation and Lib-Testing can be merged between one another to keep them up to date for implementing methods or to write new unit tests.

Lib-Dev is then used for a more complete version of both the implementation and testing. And will always have both implementations and testing merged into it.

At the end of producing a library, the final resources such as the README, and other resources required for the library will be pushed onto the main branch as the final “product”.

Installing a Jar Library to your local Maven Repository

During the development of the main application, due to the way we have designed our product to use multiple modular libraries, we will each need to install these libraries to our local maven (.m2) repository. This needs to be done so that you will be able to import the libraries into the main application project and utilise the functionality.
In order to do this, you will need to run the following steps on *EACH* of the libraries, (otherwise you wont have the libraries installed properly and will have errors).

1. Make sure Apache Maven is installed on your computer (should be already but IDE’s do not install it automatically…)
    a. Download Maven here! Make sure to get the binary not the source! (Unless you want to compile it yourself of course! 😉
    b. Follow these instructions for how to install Maven once downloaded!
2. Now navigate to the root directory for the library you want to install. It will look something like this (windows example):
   `C:\Users\username\Documents\SWEng\TextLibrary\`
3. Open this directory in a terminal window or command prompt

4. Check that maven is setup and working correctly by running the following (It should print the version in the terminal): `mvn -v`
5. Run `dir` for Windows or `ls -l` for Linux/UNIX (Includes MacOS) and check there is a `pom.xml` file.
6. You will now “deploy” the library to a local-maven-repository located in the base directory of the project.
7. Make sure before deploying the library that it isn’t already in the local directory, (unless you are updating the library with a newer version)
8. If the library you are going to deploy is already in the “local-maven-repository” directory.. Then you do not need to deploy it! Just check the each of the folders as they will be in reverse name by group id. E.g. if the XML library is in the repository there will be a set of folders called: `com/docduck/xml`
9. Run the following command.. But make sure to specify the correct file paths!!!
   The ids and version details for the jar library can be found within the pom.xml itself at the top of the file.
   (This is all one command and can be put on one line, it is separated into multiple to make it easier to read).

```
mvn deploy:deploy-file
-DgroupId=[GROUP]
-DartifactId=[ARTIFACT]
-Dversion=[VERS]
-Durl=file:{pathToProjectToDeployTo}/local-maven-repo/
-DrepositoryId=local-maven-repo
-DupdateReleaseInfo=true
-Dfile=[FILE_PATH_OF JAR LIBRARY]
-Dpackaging=jar
```

10. Add the repository and dependencies to your pom.xml now! Use the following structure and format and fill in all the dependency details: (the repository section is already complete as it is)

```
<dependency>
    <groupId>{groupID}</groupId>
    <artifactId>{artifactID}</artifactId>
    <version>{version}</version>
</dependency>

<repositories>
    <repository>
        <id>local-maven-repo</id>
        <url>file:///${project.basedir}/local-maven-repo/</url>
    </repository>
</repositories>
```

See start of next page once reached here!

Tick the checkbox to indicate you have read up to here and UNDERSTAND it!!!!

If you don't understand it, please ask me, William Betteridge !

- [x] ~~William Betteridge~~
- [ ] James Stevenson
- [x] ~~Josh Bradley~~
- [x] ~~Rob Walker~~
- [x] ~~Zhihao Ma~~
- [ ] Jabez Cheung
- [ ] Luke Warbey
- [x] ~~Hari Mamman~~
- [x] ~~Noah Carter~~

11. **OLD INSTRUCTIONS BELOW (PLEASE IGNORE, FOR REFERENCE ONLY)**
12. Now comes the installation of the library.. Run the following maven command, and it should install the library to your local maven .m2 repository:
    (This is all one command and can be put on one line, it is separated into multiple to make it easier to read).
    The ids and version details for the jar library can be found within the pom.xml itself at the top of the file.

```
mvn install:install-file
 -Dfile=<path-to-file>
 -DgroupId=<group-id>
 -DartifactId=<artifact-id>
 -Dversion=<version>
 -Dpackaging=<packaging>
 -DgeneratePom=true

Where: <path-to-file>  the path to the file to load
      <group-id>      the group that the file should be registered under
      <artifact-id>   the artifact name for the file
      <version>       the version of the file
      <packaging>     the packaging of the file e.g. jar
```

https://gist.github.com/cleberjamaral/6c9b0a615e51e26c94ffe407a641f531
I will be aiming to do something like this when I get around to it! Because it will be much easier to manage one central repository.

Link for use by William.
https://stackoverflow.com/questions/4955635/how-to-add-local-jar-files-to-a-maven-project