Reviewer Finder

The application include three major components: UI, QueryEngine(QE) and BackEnd(BE).
UI is built with JavaFX, QE and BE are built with Hibernate on top of MySQL.

Installation instructions:

1. Pre-compiled JAR with MySQL running on our AWS instance.

    - We have a JAR packaged with all dependencies under ./dist

2. Compiling from source code.
    - Current directory is an Eclipse Maven project.
        - source directories include src/ resources/
        - classpath lib/*
        - Main class of the project is com.neu.reviewerfinder.AppStarter
        - In order to create the MySQL DB on your own intance, simply run the jar using the following commands:

          ```
          java -cp <location of reviewerfinder.jar> com.neu.reviewerfinder.backend.Parser <dblp xml path> <committees folder path> <parser flag>
          ```

          where <parser flag> is a boolean which represents whether to parse the data or not (true for initial data loading only)
          This will create dblp database (if it doesn't exist) and several tables under it which support extensive range of queries.
        - Above command will parse DBLP and committees data only. In order to add User data, run the following command:

          ```
          java -cp <location of reviewerfinder.jar> com.neu.reviewerfinder.backend.UserParser <user file path>
          ```

          where <user file path> represents location of the file containing user information for importing in to the database. `users.txt` is a sample <user file> for reference which contains username and password (tab-separated).
    - Modify database connection settings in src/main/resources/hibernate.cfg.xml to use your own MySQL server. Make sure to reflect these changes in test/resources/hibernate.cfg.xml as well.

Execution Instructions:

1. Using pre-compiled JAR:

    - Double-click on the jar file to start the application
    OR
    - Go to the directory containing jar file from command line and run `java -jar reviewerfinder.jar`

2. Using source code:
    - Import the project into Eclipse and run `com.neu.reviewerfinder.AppStarter` to start the application

Test Suite:

```
-----------

1. We have used TestFx with JUnit for testing JavaFX and Java code
2. From Project root directory, use `mvn test` to run all test cases or `mvn
test -Dtest=TestName` to run individual test, where TestName is replaced by an
existing JUnit (Example: `LoginPageTest`)
    - The overall statement and conditional coverage for the test cases is:
.....

Team Infrastructure to Improve Code Quality:
--------------------------------------------

1. We use JIRA and Confluence to make development plans and track issues.
2. Git commit-msg hook can be found under ./tools. It makes sure team members
   include JIRA issue number in their Git commit message so that all commits can
   be tracked on JIRA.
3. Jenkins CI jobs are configured so that
    - JUnit tests are run daily as well as when commits are pushed to GitHub.
    - lint job is run after tests to check Java coding style
    - doclint job is run after tests to remind us of properly documenting code.
4. All Jenkins build status/results are sent to our slack channel.
5. We use EclEmma plugin for Eclipse to ensure good code coverage from tests.
6. In Order to test UI functionalities as well, we have used TestFX plugin.
```