Grading for: hw1

Name: **tannerhuynh**

Repo URL: **https://github.ccs.neu.edu/cs5500-fse/tannerhuynh-F19**

Commit Date: **Fri Sep 20, 2019 09:45:17 ET**

Late: **No**

Grader: **Sibendu Dey**

Score: 94.0

Rubric Items

Question: Git Step 1: Configure git.

Good answer given

Question: Git Step 2: Clone the repo.

+0.25pts Good answer given and did bonus

net addition to score: 0.25 - nice job

Question: Git Step 3: Edit the README.md

Good answer given.

Question: Git Step 4: Create .gitignore

Good answer given

Question: Part 2-Q1: Review the code and write a critique

-3.75pts

1. Found issue overall: no documentation
2. Found issue overall: smell with package name
3. Found issue with App.java. Good. Person.java: Found Overrides equals() and should therefore also override hashCode().
   Found Line 4, java.util.Collections is not used. Should be removed.
   Found Line 89: String name should be private.
   Found line 72 – code smell: replace if cond then true with return cond.
4. Found MatchMaker.java: Lines 4& 5: Imports are unused and should be removed.
5. Found MatchMaker.java: makeMatches() lines 12-17: could be pushed to a helper to clarify the code. Why it is there in the first place is unclear.
6. Found MatchMaker.java: makeMatches() Line 13, 16, 19: Replace use of System.out or System.err by a logger. These are mainly debugging type messages and should have been cleaned up.
7. Found MatchMaker.java: findMatches() Line 26: for statement should be replaced by an iterator.
8. Found MatchMaker.java: findMatches() Lines 33, 43, 52, 57: System.out used when most of the messages are really tracing notes. Should be removed or moved into comments explaining points in the algorithm.
9. Found MatchMaker.java: findMatches() Line 43: could get a nullPointer. This adds to why this “comment”/trace print statement should be removed.
10. Missed MatchMaker.java: findMatches() Line 56: doesn’t actually add to the function of the code and should be removed (moved to a comment?).
11. Missed MatchMaker.java: findMatches(): A real candidate for refactoring. Overall, the code is a bit bulky.
12. Found MatchMaker.java: getList() line 100: should return an empty ArrayList, not null.

13. Missed TestMatch.java: Line 31: unused variable, Significant repetition of the instantiation of Person within the tests. Should be moved setup() and teardown().
14. Missed TestMatch.java: Significant repetition of the instantiation of Person within the tests. Should be moved setup() and teardown().
15. Found TestPerson.java: Inconsistent use of Logger – why is this the only place a logger is used?
16. Missed TestPerson.java: :Significant repetition of the instantiation of Person within the tests. Should be moved setup() and teardown().

total deduction taken: 3.75

Question: Part 2-Q1: Issues found above and beyond

2.50pts

1. Found MatchMaker.java Line 34: while test should be extracted into a method to reduce complexity of the conditional.
2. Found MatchMaker.java Lines 36, 38 should be extracted into a private method to promote clarity
3. Found MatchMaker.java Lines 46, 47 should be extracted into a private method to promote clarity.
4. Found Attributes.java: it's unclear whether all combinations of pairs are allowed. For example, female and male make sense as does Advisor and Student, but does female and student make sense? You can argue there should be some kind of match attribute aligner. Could be here or it could be policy in the MatchMaker.
5. Found Person.java: given there are three attributes that may be set on instantiation, some kind of builder or factory should have been used
6. Found Person.java: The code doesn't enforce that a person must have a name or be in a pool. While you can argue that putting a person in a pool isn't a requirement at first use, it's hard to argue about names not being set. Further, the uniqueness of the name isn't enforced.
7. Found MatchMaker.java that there is a curious combination of java 8 (functional) and not java 8 styles.
8. Found TestPerson that the use of the logger is fine, but it's not adding to the information set. If the intent is to set up a general stack trace to assist in debugging, that's fine. But the logger is cluttering the test and should be moved to the set up and tear-down activities.
9. Found Person.java on Property name: it's unclear if once set, you should be allowed to change it.
10. Missed the code does not follow the principle "Program to an interface, not implementation"
11. Found MatchMaker might store the proposers and proposees differently so there's a single container for either group and the get and set methods do not have to do the conditional on type. For example, you as a hashMap of > instead of two lists. Then setting up the lists or returning the lists wouldn't need the switch statements. The downside is if you call for the lists more than once, you will be iterating through the container more than once (the cost of setList right now).

net addition to score: 2.5 - nice job

Question: Part 2-Q2: Create a branch called refactor.

Good answer

Question: Part 2-Q3: Refactor the code

-5pts Issues identified in question 1 still remain (less than one-third of the total identified)

total deduction taken: 5.0

Question: Part 2-Q4: Commit the final draft of your branch and push the branch to origin.

Good answer

Question: Part 2-Q4: Merge refactor to master. Push master to origin.

Good answer

Question: Part 3-task-1: Git

Good answer

Question: Part 3-task-2: Refactoring the code

Good answer

Question: Part 3-task-3: Reflection

Good answer

Question: Additional Comments

Awesome job with the refactored code especially the stable match algorithm.
Suggestions:
1. Please try to follow the instructions carefully next time. All the turn-in tasks as well as critique tasks were asked to add under a single document.
