**Honor Homework 2 Assessment**

**Are there things that could be improved?**

Issues with implementing the Strategy Pattern:

The initial code had only created the computer player once in the beginning and did not have a factory for creating a new instance of a computer for a different strategy. Therefore, a factory has to be made which will handle the instantiation of new computer instances within the player selection menu. There also wasn’t an attribute inside the computer instance for the strategy it used so a strategy class and the attribute will need to be added. The nextcard() method for the computer exists in the PlayUno class which needs to be refactored and moved into the strategy class somehow in order for the computer to call upon it to determine the card it chooses to play depending on the strategy it uses.

Issues with implementing the Command Pattern:

The PlayUno’s playgame method has the execution of the rounds implemented using a while loop. Therefore, the implementation of doCommand() will be difficult. I believe I will save all the information that changed during the round in some kind of a game state class. The PlayUno class will hold a stack of commands, each of which will hold it’s game state information. When the player selects “undo”, the command will roll back until it’s the human’s turn again.

**Things that work well?**

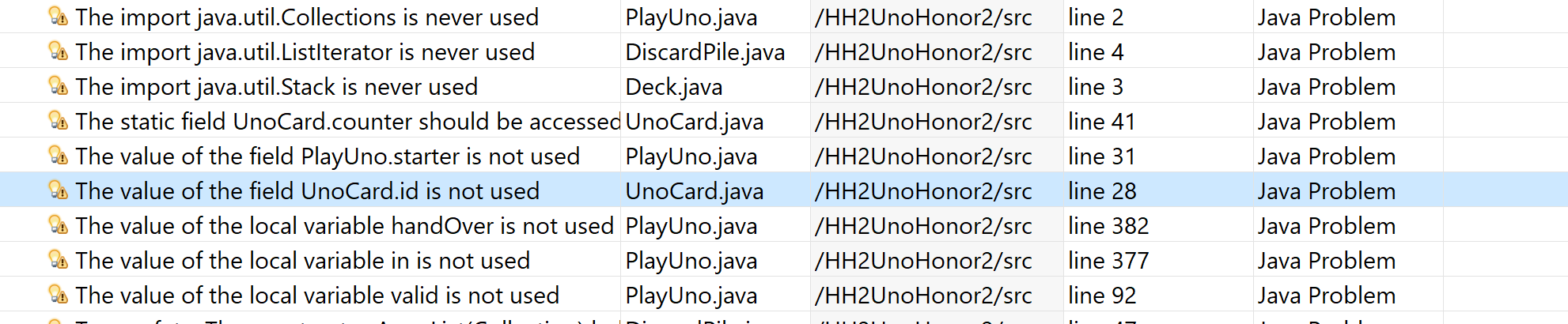
UnoCard implemented the Comparable interface allowing me to use the sorting method very easily from Collections.

**Fix any style bugs/missing Javadoc you might find and describe them in your write-up (hint: are any of the methods too long?).**

The playgame is very long. However, since the code is working currently and it’s so convoluted, I’ve decided to separate my own code with low coupling as much as possible from the playgame method so that I didn’t need to deal too much with it with high cohesion within my classes and methods.

**Is there any code that isn’t used?**

yes



**One of the tests fails – why?**

The UnoCard equals method was checking && for the cards’ ids which were different. But that’s changing the code. Since the .contains method and the equals method are doing there job correctly. The contains should return a false. Therefore, changing the asserts to false will fix the test.

**Are there any questionable design decisions or places where bugs could creep in depending on how methods are used? Is the visibility of attributes and methods appropriate?**

The playgame method has many if statements with convoluted structural choices. The cohesion for methods inside the PlayUno class is very low and the responsibility is very diffused. The visibility of attributes and methods are mostly appropriate. However, due to the high coupling between classes, there are many occasions where many “dots” for calling methods is a necessity.

**Third Strategy:**

Simply plays the lowest pointed card in the computer’s hand.