Razvan Hogea – Additional Explanations

To solve the Automation exercises provided by the recruiter, I decided to utilize Java as a programming language, Selenium as the automation framework, Junit for defining the tests and JSON simple to decode the response of the Card Game API's.

The format of the project is as follows:

- The "pages" folder contains the page objects for both the "Checkers Page" and "Deck of Cards Page". The page objects have the role to store the page URL, page elements stored via xpath or id as seen in the respective annotations (i.e. @Findby(xpath="…")), static elements, as well as various methods and functions.
- The "tests" folder contains the two test cases for each page: "Checkers Test" and "Deck of cards test".

1. The Checkers Game

As mentioned previously, the checkers game solution is split into two files: "CheckersPage.java" and "CheckersTests.java".

The five most important functions in the Checkers Page which will aid in our solutions for the test are:

- isPageLoaded() which checks if the current url of the driver is equal to the url of the page
- canMakeMove() which uses the "Make move." text on the screen to ensure that the game is not over
- getCurrentBoard() which builds a matrix of the current board based on the images on the screen
- makeAnyMove() which checks if there is an empty spot in the matrix either on the top right diagonal or top left diagonal
- didGetBlue() which returns true if the orange managed to get a blue. The function loops through the matrix and tries to find a blue piece followed by an empty space on the diagonal in a greedy manner and makes the move if there is an available spot.

The solutions to the exrecises can be seen in the "Checkers Tests" file, and are implemented as follows:

- to check if the page is loaded, we verify that the current url is correct, that the page title is available, as well as if the document state is complete

- Next, we start of by making an initial move such that the game starts, and we can utilize the “make move” message.
- For the rest of the moves, we will use a while loop to try to get a blue piece every move or make any move otherwise. The loop will finish when we made over 5 moves and got a blue piece or when the game is over as indicated by the message presented on the page. The test would pass only if we made over 5 moves and got a blue piece as well.
- After restarting the game by clicking the button we check if the message is back to the initial message of “Select an orange piece to move” and if the current board equals to the initial board. If both conditions are fulfilled the test passes.

Results:

- For the UI automation we can see the results in the attached video: https://drive.google.com/file/d/1szLhkZNqY6i1acUcI8tzvuOD6oWEMNgk/view?usp=sharing

Possible improvements:

Additionally, we can have a greedy approach and take as many blue pieces as possible until eventually we win the game.

2. The Deck of Cards

As mentioned previously, the checkers game solution is split into two files: “DeckOfCardsPage.java” and “DeckOfCardsTests.java”.

The most important functions in the “DeckOfCardsPage.java” which will help with our solution are:

- isCardPageLoaded() which checks if the current url of the driver is equal to the url of the page
- getNewDeck() which utilizes the API for getting a new deck and updates the deck id of the object if successful to later be used in the rest of the API’s, without the need of calling the API again
- shuffleDeck() uses the provided API as well as the deck id to shuffle the deck and updates the isShuffled and isShuffledWithSuccess parameters if the API response was a success
- dealThreeCards() which takes the player number as a parameter and uses the API to deal 3 cards from the existing deck. The function also uses a helper method “getCardValue” which assigns a value of 10 for face cards, corresponding numbers to numbered cards, and 11 or 1 to the ACE based on the current value of the player. The function assumes that the player

is forced to get the 3 cards and does not know what card comes next when deciding the ACE value, just the previous card.

Results:

- A demo of the cards game can be seen here: https://drive.google.com/file/d/18TiKhK8xqYfKNIDDagJ_5nkftMCz71B9/view?usp=sharing

Improvements:

One possible improvement is to create a function which takes the API url and returns the JSON response since some of the lines are duplicated. Another possible solution is to create separate classes for each player and for the deck of cards.