CPSC2150 – Project Exam 1

Open: 2-27 @ 8:00am

Closed: 2-29 @ 11:59pm

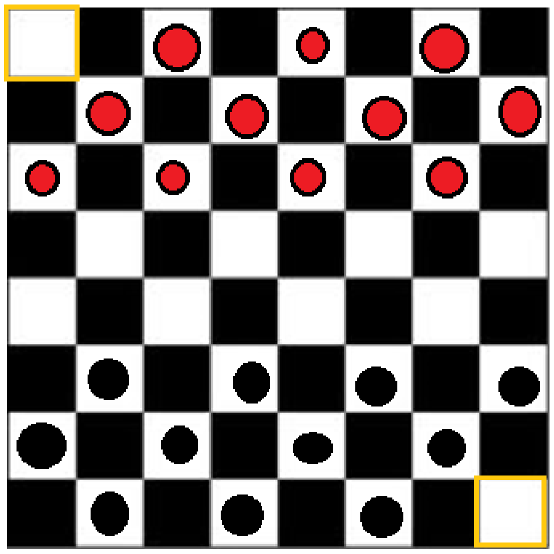
You have 3 hours to complete this exam, starting from the moment you access it.

This is to be done individually! Do not go to lab to complete this exam.

Your name:

Background:

Let's consider an alternative version of checkers called "Kings Corner". In Kings Corner Checkers, both players only begin with 11 pieces, missing a piece in the two playable corners, called the "King's Corners", of the board. This means our starting board looks like this:



For regular pieces and regular kings, the same normal rules of moment of standard checkers apply. Regular pieces can only move in the two diagonal directions towards their opponent, one space at a time. Regular kings can move in all four diagonal directions, one space at a time.

However, in Kings Corner Checkers, any checkers piece that makes it to their opponent's "King Corner" is given the title of "King of all Kings". "King of all Kings" pieces have the following special properties that makes them different from regular kings:

- Each player can only have 1 "King of all Kings" piece at a time.
- "King of all Kings" pieces **can only be jumped by king pieces and other "King of all Kings" pieces.**
- "King of all Kings" pieces can move in all 4 diagonal directions and jump pieces normally.

- Jumping one of your own regular, non-kinged, pieces with a “King of all Kings” piece causes that jumped piece to become a regular King piece.

Now, imagine we are updating our projects to include this “Kings Corner” version as an alternative version to the regular checkers the players can play. All the input and output of the game (that you should have seen in the Example Output PDF from project 1 – part 1) should be the same. However, instead of jumping right into the game, the program would first ask the user to enter “regular” or “kings corner” to select which version of the game they wish to play.

Question 1 (30%):

Write 3 new user-story form functional requirements that we would have to add to our project if we implemented “Kings Corner” into the requirements.

Functional Requirement #1:

As a player, I can enter the version of the game so that I can play the version with the settings I wish to play

Functional Requirement #2:

As a player with a King of all Kings piece, I need to be able to jump my own pieces so that I can make them become kings

Functional Requirement #3:

As a player facing a King of all Kings piece, I need to be able to jump the opponent’s piece with my own king or King of all Kings piece so that I can remove their King of all Kings piece from play

---

Question 2 (15%):

Write 3 non-functional requirements we would need to add to our project if we implemented “Kings Corner” into the requirements.

Non-Functional Requirement #1:

Jumping friendly pieces to make them kings should not add significant lag or calculation time to the game

Non-Functional Requirement #2:

Getting to the opponent’s Kings Corner should reliably turn your piece into a King of all Kings

Non-Functional Requirement #3:

King of all Kings movement patterns should be consistent with a normal king's movement with the addition of jumping friendly pieces

---

Question 3 (15%):

Download and unzip the student.zip from canvas. Inside you'll see a PNG of a student UML diagram and a Student.java class.

Identify **at least three** mistakes the student UML Makes.

Mistake 1: major is capitalized in the UML when the field is lower case in the class

Mistake 2: name has a + when the field is private, so it should be -

Mistake 3: birthdate has a + when the field is private, so it should be -

Mistake 4: Student constructor should be + instead of - because it is a public constructor

Mistake 5: Student constructor is lowercase in the UML when the constructor is uppercase

more:

studentsStaticMethod is static and should be underlined

setName, setBirthdate, setID, setMajor are all also public but have a - in the UML and should be +

Question 4 (40%):

Write the contracts for:

- The student class itself
- The parameterized student constructor
- studentsStaticMethod(String)

This class contract and 2 method contracts should be done in the student.java file.

Submission:

Place student.java and this doc (**SAVED AS A PDF!!!!!!!)** In a folder and zip it up and submit it to **both** canvas **and** gradescope.

To clarify you are submitting a zip file that contains:

- **student.java**
- **this document, with your answers on it to questions 1-3, as a PDF**

If we see anything other than what is above, which includes but is not limited to:

- student.java and this document as a .doc or .docx or .anythingElse
- only student.java
- only this document
- nothing (????)

It's an auto **ZERO** for being unable to follow directions.

If you submit only to gradescope and not to canvas, it's an auto **ZERO**.

If you submit only to canvas and not to gradescope, it's an auto **ZERO**.