Assignment for TCP2201 Object Oriented Analysis and Design

Trimester1, 2018/2017

You may have a maximum of 4 people per team.

In your code, you must put comments documenting each method (function) as to who wrote that method. (If more than one person worked on a method, you may list all their names.)

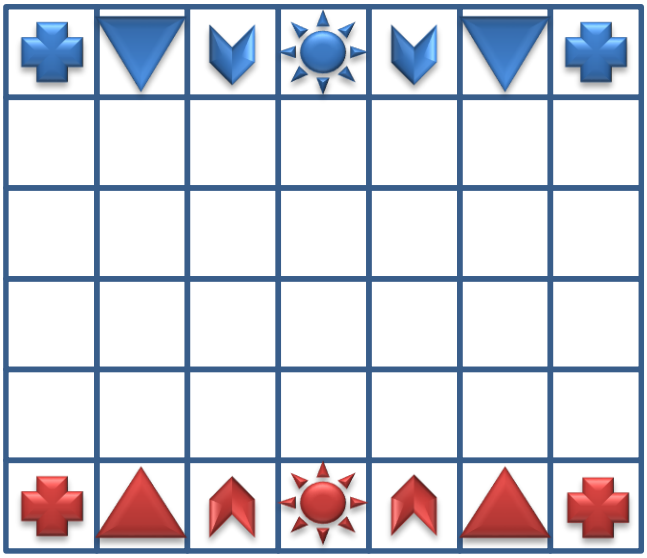
During evaluation, I may call any student to come and modify some code to do something differently in front of me to prove that they actually wrote the sections that their name appears on.

I also have a code plagiarism checker, which can identify code copied from others outside your team. If you copy code from anyone else, I will give you **zero**. In the past, I have given many zeros because I detected their plagiarism, so please avoid this heartache for yourself. **If you give your code to someone else to copy, it is also considered cheating and you can also get zero** so do not give your code to anyone else to copy.

The project: Myrmidon Chess

You are to implement a Myrmidon Chess game as a GUI-based Java Application. You do not have to implement a computer player – it can just be a two human player game.

Myrmidon Chess is played on a 6X7 board, like this:



The above layout is the initial position of the game pieces.

The Plus can move vertically or horizontally, up to 2 steps in any direction in a straight line.

The Triangle can move diagonally, both up and down, up to 2 steps in any direction in a straight line.

The Chevron must move in an L shape, exactly two steps then 1 step right or left. (This is the only piece that can jump over other pieces.)

The Sun can move only 1 step in any direction. The game ends when the sun is captured by the other side.

After 3 turns, a Plus will turn into a Triangle, a Triangle will turn into a Chevron, and a Chevron will turn into a Plus. (This makes Myrmidon chess different from all other chess games, because the pieces will transform like that.)

You must use good object-oriented design concepts in designing your program. Subclassing, delegation, composition and aggregation should be used where appropriate.

In **addition** to MVC, you must use design patterns in your code, and you must identify what design patterns you use and where. For example, the board might be a Singleton. The behaviour of the chess pieces might be implemented as a Strategy or State. You might use an Abstract Factory, a Factory method, or a Prototype to create the pieces. (These are only ideas; you do not have to use these particular design patterns, and are encouraged to think about which design patterns might be suitable.) You may come to see the lecturers to discuss your design.

You should make your program user friendly, with suitable menus, save game, resizable windows, flipping the screen when it is the other player's turn, etc.

You must document every class and method. You must practice proper indentation. **Marks will be deducted** if you do not do this.

Please see us early if you have problems – whether it is problems understanding what you need to do, or problems with team members who are not doing their work. The earlier you see me, the better chance you'll have of solving the problems. If you wait till the last week before the due date, it'll likely be too late to fix the problems.

Project Team Registration

You create your group on MMLS. If there is any discrepancy between the team members listed in the peer review Excel files and the MMLS team members, you will be liable for any loss of marks.

If you find as the project progresses that some people are not contributing or problematic in any other way, **please contact your lecturer immediately** so that remedial action can be taken before it gets too late. If you do not do so, you will be liable for any loss of marks.

Deliverables

1. Java Source code for the entire project
    a. Every class must be commented to show what the purpose of that class is. If the class is part of a design pattern, document what part it plays.
    b. Every method must be commented to show who wrote that method, and if it is not obvious, the purpose of that method.
    c. All the code must be properly indented.
2. Report containing
    a. A header like this:

> **TCP2201 Key Collector Project**
> Trimester 1, 2018/2019
> *by <<TEAM NAME>>*
>
> Team Leader: Name, phone number, email
> Team members:
> Name, phone number, email
> Name, phone number, email
> Name, phone number, email

    b. Instructions how to compile & run your program **from the command line**, and user documentation on how to use your program. **This is especially important if you developed your code using an IDE.** The lecturer marking might not have your IDE, or a different version of your IDE. **You are responsible for any loss of marks if the lecturer has trouble compiling and running your code**. (Especially be careful

of capitalization of file names – Windows ignores the capitalization of file names, but LINUX and Mac do not. Some of the lecturers might be marking on LINUX or Mac.)

c. UML Class diagram – also indicate which classes are participating in which design patterns and what their roles in the design patterns are.
d. Use Case diagram – show the main functions
e. Sequence diagrams – for **each** of the use cases from the Use Case diagram.

The documentation, UML Class Diagram, Use Case Diagram and Sequence diagrams **must** reflect the version of the code submitted or marks will be deducted.

Zip up all the source code files together with the report and submit to the MMLS Assignment submission system by 6pm of the due date. **Each group submits one project, according to the MMLS group.**

In addition to the main submission above, each group member should submit the "Assignment Peer Review.xlsx" file *individually* to the "Assignment peer review" project submissions. There will be **three** peer reviews at intervals during the project. This peer review submission is secret – your team members will not see it, so you can be completely honest about how they performed.

- If the person is a sleeping partner or contributed almost nothing, you can evaluate them 0-1
- If they contribute very little effort, can rate them 2-3.
- If they are seriously involved and actively contribute, you may rate them 4-5. These marks are private and confidential. Kindly submit to us on MMLS in the "Assignment peer review" submission.

For the first review, you should rate your teammates from the start of the project till the first review. For the second review, you should rate your teammates from the first review until the second review. For the last review, you should rate your teammates from the second review till the project submission.

Note: **All** students must be involved in the programming. You cannot say "This student just did the documentation" or "this student just did the UML diagram."

Do not email me your project unless MMLS Assignment Submission is not working.

Late policy: 10% will be deducted if the project is submitted on 1 day late. 20% will be deducted if it is 2 days late. 30% will be deducted if it is submitted 3 days late. 40% will be deducted if it is submitted 4 days late. No submissions will be accepted after that.

Due Dates

- Peer review #1 due Monday 27 August
- Peer review #2 due Monday 10 September
- Project Due Monday 24 September 2018.
- Late policy applies until Friday, 28 September 2018.
- Peer review #3 should be submitted immediately after you submit your project.

If you have submitted a version, but then change it, you can re-submit until the cut-off date for each of these items, and the new version will replace the old version.

TCP2201 Project Evaluation Form (30%)

Tutorial Section:	
Team Name:	
Group Leader:	
Member	
Member	
Member	

Prototype and Presentation (20%)

Item	Maximum marks	Actual Marks
Comments, indentation, following proper Java naming conventions, other Java style issues.	2	
Object-oriented concepts like subclassing, delegation, composition, aggregation, polymorphism, etc.	3	
Appropriate use of Design Patterns	3	
User friendliness and appropriate GUI components used, windows resize properly and the board scales properly, menus still work during game play, etc.	4	
Functional requirements fulfilled, e.g. the board is set up correctly, players can play through a game properly, all the pieces move and transform correctly, winner is declared, save game, load saved game, etc.	8	
Total:	20	

Report (10%)

Item	Maximum marks	Actual Marks
UML Class Diagram done and is coherent with the implementation	3	
Use Case Diagram done and is coherent with the implementation	2	
Sequence Diagrams done and is coherent with the implementation	3	
User Documentation done and is coherent with the implementation	2	
Total:	10	

Note: Individual marks will be adjusted after the lecturer evaluates the “Assignment Peer Review.xlsx” each student submits individually. Your teammates will not be able to see what you wrote about them there, so you can be honest in your evaluation of your teammate’s efforts.