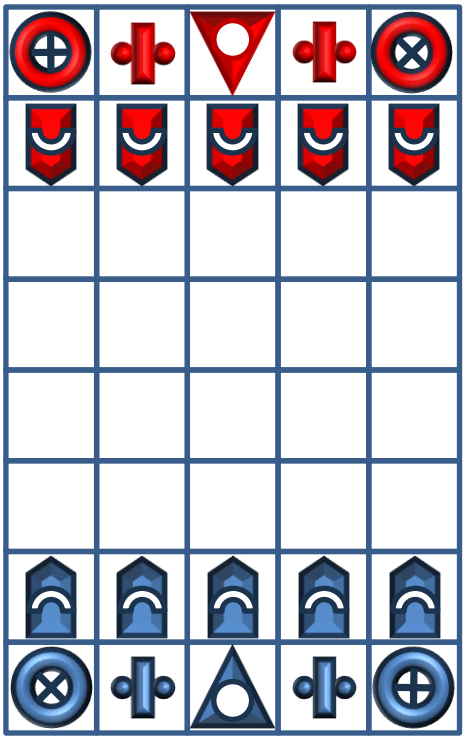
Assignment for CCP6224 Object Oriented Analysis and Design

Trimester 2410 ☉ Teams should be of 3 or 4 people.

The project: Kwazam Chess

You are to implement a Kwazam 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.

Kwazam Chess is played on a 5x8 board, like this:



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

	The Ram piece can only move forward, 1 step. If it reaches the end of the board, it turns around and starts heading back the other way. It cannot skip over other pieces.
	The Biz piece moves in a 3x2 L shape in any orientation  (kind of like the knight in standard chess.) This is the only piece that can skip over other pieces.
	The Tor piece can move orthogonally only but can go any distance. It cannot skip over other pieces. However, after 2 turns, it transforms into the Xor piece.
	The Xor piece can move diagonally only but can go any distance. It cannot skip over other pieces. However, after 2 turns, it transforms into the Tor piece.
	The Sau piece can move only one step in any direction. The game ends when the Sau is captured by the other side.

None of the pieces are allowed to skip over other pieces, except for Biz.

After 2 turns (counting one blue move and one red move as one turn), all Tor pieces will turn into Xor pieces, and all Xor pieces will turn into Tor pieces. (This makes Kwazam chess different from 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.

You **must** use the MVC pattern since this is a program with a GUI. This means you must not mix code that tells how the game works with the GUI code. This means that if you later put this game on another GUI system, e.g. Android,

you will not need to change any model code – you will only have to replace the GUI code. So, your view and controller classes must not have any of the game logic – all the game logic must be in the model classes only.

You must use design patterns in your code, and you must identify what design patterns you use and where. As this is a GUI program, you **must** MVC. You also may use Singleton and/or Template Method, but if you do, you need to use at least one other design pattern as well.

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. For save and load game, the game should be saved into a **text file** so that it's **human-readable**.

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 meet us, 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.

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.

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

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 header like this:

> **TCP2201 Project**
> Trimester 2310
> *by <<TEAM NAME>>*
>
> Team Leader: Name, phone number, email
> Team members:
> Name, phone number, email
> Name, phone number, email
> Name, phone number, email

    A. Compile and run instructions – 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.) **If you do not include these instructions, you will lose marks.**
    B. UML Class diagram – also indicate which classes are participating in which design patterns and what their roles in the design patterns are.
    C. Use Case diagram – show the main functions.
    D. Sequence diagrams – for **each** of the use cases from the Use Case diagram.
    E. User Documentation – A user manual on how to use the program.

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. **Do not have a zip file inside another zip file** – it causes the lecturer extra work so if you do it we'll deduct marks for not following instructions. **Each group submits one project in eBwise.** If you submitted before the due date and want to resubmit, just upload a new version in eBwise. Any student in the group submitting will be considered by eBwise to have been submitted by the whole group.

In addition to the main submission above, each group member should submit the "Assignment Peer Review" form *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 your lecturer your project unless eBwise 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. So, for example, if your mark is 30 out of the possible 40, and it's 2 days late, it will be $30 \times 0.8 = 24$.

Warning about plagiarism

We also have a code plagiarism checker, which can identify code copied from others outside your team. If you copy code from anyone else, we will give you **zero**. In the past, we have given many zeros because we 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. If you learn from an online source or past year projects, that's fine, but **write your own code**. Even if you type it in yourself, if you're using their code, it's considered cheating. **Many people have failed this course in the past because of such plagiarism, so don't repeat their mistake!**

Due Dates

- Peer review #1 due Monday, 23 December 2024.
- Peer review #2 due Monday, 6 January 2025.
- Project Due Monday, 20 January 2025.
- Late policy applies until Friday, 24 January 2025.
- Peer review #3 should be submitted immediately after you submit your project.

If you have submitted a version of the project, but then change it, you can re-submit until the cut-off date, and the new version will replace the old version. If it is after the due date but before the cut-off date, please inform your lecturer not to mark it yet, and note that the late policy will apply.

Rubrics

Prototype and Presentation (25%)

Item	Maximum marks	Rubrics					
Style issues: • Proper commenting. • Proper indentation • Identifier names chosen make sense. • Proper case of identifier names.	1	1 – All fulfilled.	0.75 – Mostly fulfilled	0.5 – Some fulfilled	0.25 – Mostly not fulfilled		
Object-oriented concepts implemented correctly. • Subclassing • Delegation • Composition and/or aggregation • Encapsulation. • Polymorphism. • No unwarranted if/else/switch.	5	5 – All fulfilled.	4 – 1 criterion not fulfilled.	3 – 2 criteria not fulfilled.	2 – 3 criteria not fulfilled.	1 – more than 3 criteria not fulfilled.	0 – nothing fulfilled
Appropriate use of Model-View-Controller, and at least one additional Design Pattern beyond Singleton and Template Method.	5	5 – MVC and other patterns implemented correctly.	4 – MVC is correct and other patterns present, but have incorrectness	3 – MVC and several patterns present but incorrect.	2 – several patterns incorrectly present.	1 – Only 1 pattern partially present.	
User friendliness • Appropriate GUI components used • Windows resize properly • The board scales properly • Menus still work during game play • The board flips for each player • No other user-friendliness problems.	5	5 – All fulfilled.	4 – 1 criterion not fulfilled.	3 – 2 criteria not fulfilled.	2 – 3 criteria not fulfilled.	1 – more than 3 criteria not fulfilled.	

Functional requirements • The board is set up correctly • Players can play through a game properly • All the pieces move and transform correctly • Winner is declared correctly • Save game • Load saved game • Saved game file is a human-readable text file. • Other functional requirements.	9	9 – All fulfilled.	8 – 1 criterion not fulfilled.	7 – 2 criteria not fulfilled.	6 – 3 criteria not fulfilled.	5 – 4 criteria not fulfilled	4 – 5 criteria not fulfilled	3 – 6 criteria not fulfilled	2 – 7 criteria not fulfilled	1 – more than 7 criteria not fulfilled.	
Total:	25										

Report (15%)

Item	Maximum marks	Rubrics					
Accurate and complete command line compile and run instructions.	1	1 – Instructions given and they work fully.			0.5 – instructions given but they're not correct.		0 – nothing fulfilled
UML Class Diagram • Coherent with the implementation • Design patterns are clearly marked.	5	5 – All classes and relationships drawn correctly, and design patterns clearly marked, showing all the proper interactions.	5 – All classes and relationships drawn correctly, but the design patterns are not correctly and clearly marked, or not showing proper interactions.	3 – Some classes or relationships or design patterns not correctly and clearly marked or not showing proper interactions.	2 – The diagrams have some relation to the code but it's badly done with a lot of errors and/or omissions.	1 – The diagram is badly drawn, or doesn't correspond to the code at all.	
Use Case Diagram done and is coherent with the implementation.	2	2 – All use cases clearly shown and correctly drawn.	1.5 – All use cases clearly shown and drawn, most of them no mistakes but some have mistakes.	1 – There is a UCD but it's got lots of mistakes.	0.5 – Badly done UCD.		
Sequence Diagrams for each use case done and is coherent with the implementation.	5	5 – All done correctly.	4 – All the use cases have sequence diagrams, but a minority of use cases' sequence diagrams have some mistakes.	3 – All the use cases have sequence diagrams, but more than half the sequence diagrams have mistakes.	2 – Many use cases do not have sequence diagrams, but those that do mostly correspond to the code, though there may be some mistakes.	1 – Most use cases do not have sequence diagrams.	
User Documentation done and is coherent with the implementation.	2	2 – Fulfilled.	1.5 – Mostly fulfilled.	1 – Partially fulfilled.	0.5 – Mostly not fulfilled.		
Total:	15						

Note: Individual marks will be scaled after the lecturer interviews you, based on how much work each person did. If all students in a group did equal work, the scaling factor is 1 – i.e. the marks you get in the rubrics is exactly what you will each get. If someone did substantially more work than the other team members, their score will be scaled up by a factor greater than 1. If they did less work than the others, their score will be scaled down by a factor less than 1.