



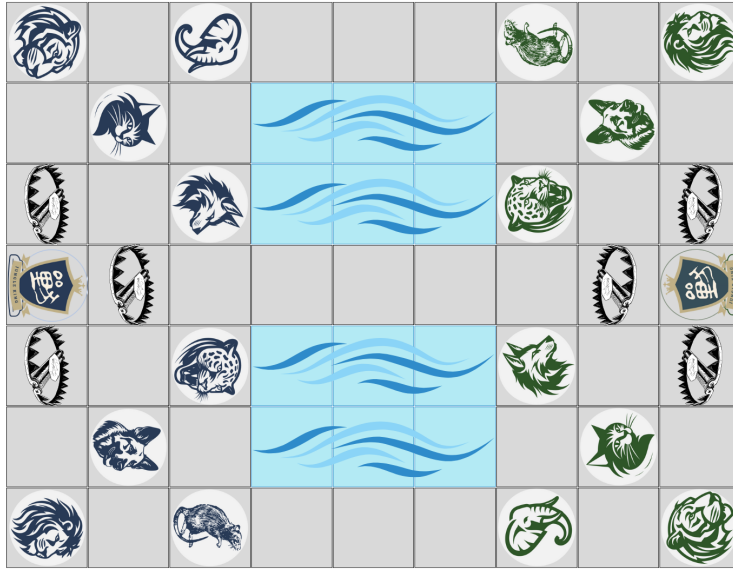










Jungle King is a turn-based two-player board game. The board is a 9×7 grid . On each side of the board are the home bases  surrounded by traps . There are two small lakes  in the middle of the board.



1 The Game

Each player controls 8 animal pieces, namely elephant , lion , tiger , leopard , wolf , dog , cat  and rat . The initial positions of the pieces is shown in the figure above.

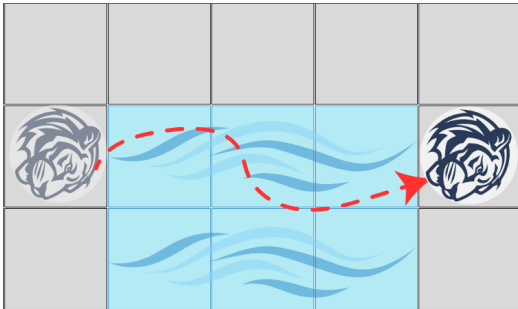
1.1 Player Move

On each turn, the player can choose to move one of his pieces. By default, each piece can move forward, backward, to the left, or to the right, onto an empty square immediately next to it.

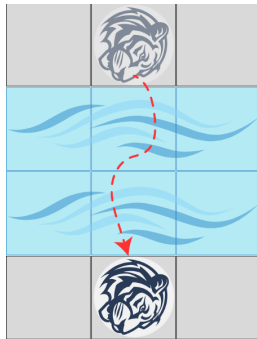
The lion, the tiger, and the rat have additional moves allowed.

Lions and tigers can cross the lake. The lion and the tiger may cross the lake, horizontally or vertically, and land on the square immediately after the lake. This square is either empty or occupied by a capturable opponent's piece. If an opponent's piece is on the other side of the lake, it will be captured after the lion or the tiger crosses the lake.

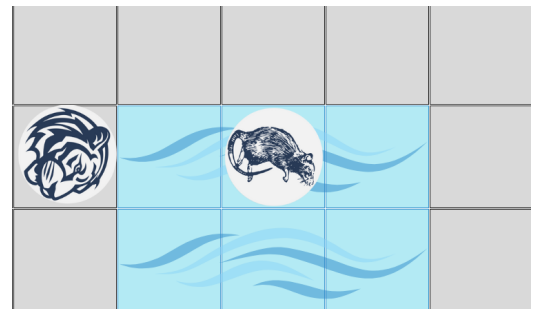
The lion or the tiger cannot cross the lake if a rat is along the way.



The tiger can cross the lake.



The tiger can cross the lake.











The tiger cannot cross the lake. The rat is along its way.

Rats can swim on the lake. The rat may swim on the lake. Each time, the rat swims one square. When the rat is on the lake, animals on the land cannot capture it. The rat from the lake cannot capture the elephant on the land. However, the rat may capture the opponent's rat on the lake.

1.2 Capture

Generally, a stronger animal captures the equally strong or weaker animal. Captured pieces are removed from the board.

The arrangement of animals from strongest to weakest is as follows: elephant , lion , tiger , leopard , wolf , dog , cat  and rat . This means, the elephant can capture the lion, the tiger, etc.; the lion can capture the tiger, the leopard, etc.; and so on.

Only the rat can capture the elephant (since it can sneak into the elephant's ears or nose).

1.3 The Traps

The traps limit the opponent's attack.

When an opponent piece lands on the trap, it becomes weak. This piece regains its strength once it leaves the trap.

When an opponent's piece is on the trap, any player piece occupying the neighboring square may capture the piece on the trap.

1.4 Before the Game

Eight animal pieces are shuffled, faced down, are presented to the players. Each player is asked to select one of the eight animal pieces. The player who selected the stronger animal will be the first player.

1.5 End Game

Game ends, when a player lands on the opponent's home base. The player wins the game.

Note that, the pieces of the player cannot land on his own home base.

2 Your task

Your task is to create an implementation of a GUI game **Jungle King** based on the description above. The implementation must follow the MVC-architecture.

You are required to apply the object-oriented concepts learned. You are required to create and use methods and classes whenever possible. Make sure to use object-oriented concepts properly. No brute force solution.

2.1 Deliverables

For each Phase

1. UML class diagram (**png** file, 300ppi);
2. Java source code files of the implemented classes, with internal documentation;
3. Meaningful program documentation generated via **javadoc**; and
4. Test case document.

MCO1 Requirements:

Due: Mar 10, 2025 (M) before 0800

1. Draw the UML class diagram to represent the system described.
2. Implement only the classes that allows the ____ and ____ pieces to move on the board with all terrains except *trap*. The program ends when the piece reaches the opponent's base.
3. Phase 1 implementation requires user to enter W, A, S, D (or any other keys to indicate direction), and displaying of essential information on the console (no GUI required yet).

MCO2 requirements:

Due: **Aug 1, 2025 (T) before 0800**

1. You must follow the MVC architecture.
2. Your final system **must be a GUI-based program**. All interactions will be on the GUI components. Console outputs will not be checked.
3. Complete UML class diagram.
4. Phase 2 implementation requires GUI.
5. Additional features may be implemented by the programmer. However, these should not contradict the requirement. Implemented additional features should be reflected in the UML class diagram too.

3 Important Notes

1. This project is done in **groups of 2 CCPROG3 students** from the same section. A student may opt to work alone.
2. A student cannot discuss or ask about design or implementation with other persons or entities such as AI chatbots, bulletin boards, etc., with the exception of the teacher and his/her group mate. Copying other entities' work or working in collaboration with another entity is not allowed and is punishable by a grade of 0.0 for the entire CCPROG3 course. A discipline case may be filed with the Discipline Office. In short, do not risk it; the consequences are not worth the reward.
3. The given description of the program is the basic requirement. Any additional feature will be left to the creativity of the student. Bonus points would be awarded depending on the additional implemented features. These additional features could include new types of game elements, including new relationships/restrictions among the game elements. Depending on the scale of the new feature, additional points will be awarded to the team. However, make sure that all the minimum requirements are met first. If this is not the case then no additional points will be credited despite the additional features.
4. For the minimum requirements of this MP, all the requirements written in this document should be implemented and working.
5. Do not forget to include internal documentation (comments) in your code.
6. You are required to create and use methods and classes whenever possible. Make sure to use Object-based (for MCO1) and Object-Oriented Programming concepts (for MCO2) properly. No brute force solution.
7. Submission of the project for each phase was announced during first day of classes, and indicated also in the Syllabus. Late submission, as indicated by AnimoSpace, will not be accepted, and will therefore result to 0 for that phase.
8. The students are responsible in submitting in the AnimoSpace assignment submission page the correct version. Only the last uploaded version by the deadline will be used as basis for assessment. As part of the requirement, the proponent/s should make pertinent back-ups of his/her/their own project.
9. During the MP demo, it is expected that the program can successfully be interpreted into bytecode file and will run. If the program does not run, the grade for that phase is 0. However, a running program with complete features may not necessarily get full credit, as design and implementation (i.e., code) may still be checked.
10. During the demo, all members of the group should be present. The group should know how to generate the bytecode file and to run the said file using CLI. Apart from question-answer, there is an Individual Programming Task that will be given to each member during MCO2. Each will have to work on and finish the given task during the demo period.
11. A student or a group who cannot answer questions regarding the design and implementation of the submitted project convincingly will incur a grade of 0 for that project phase.
12. All sources should have proper citations. Citations should be written using the APA format. Examples of APA- formatted citations can be seen in the References section of the syllabus.