Planning Tic-Tic-Toe Game

UML Use-Case Diagrams

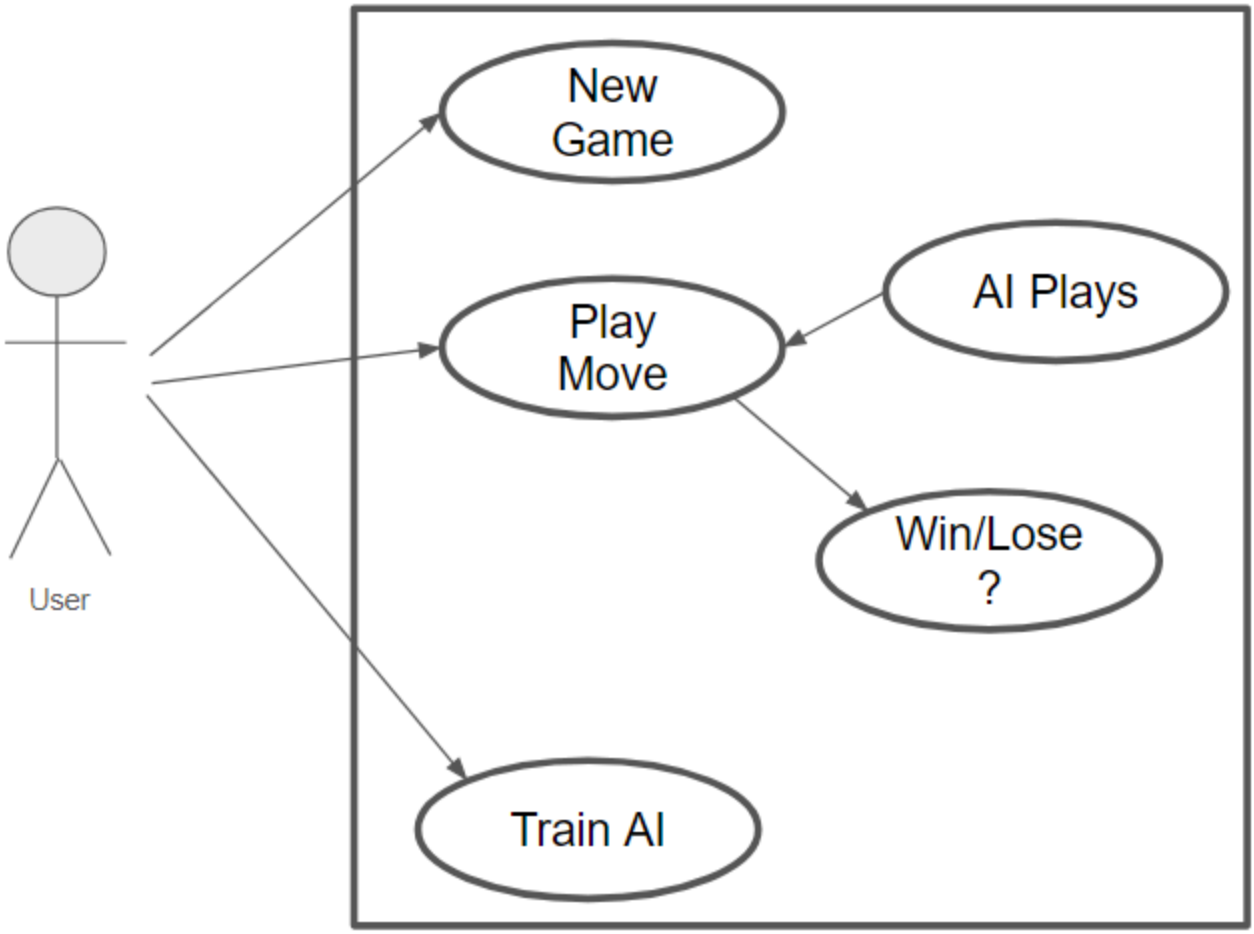
As we begin to plan a program to play Tic-Tac-Toe it is important to consider how we want our **users** (i.e. humans) to be able to interact with it. When planning how a user will interact with a system, a UML "Use-Case Diagram" is a handy tool.

A UML (Unified Modeling Language) Use-Case diagram is a visual representation that illustrates the interactions between actors (individuals or systems) and a system under consideration. It primarily focuses on the functionality provided by a system from the user's perspective.

In a Use-Case diagram, actors are depicted as stick figures or symbols representing external entities interacting with the system. Use cases, represented as ovals, describe specific interactions or functionalities offered by the system. Lines connecting actors and use cases indicate relationships or interactions.

The diagram helps to identify and clarify system requirements by outlining various use cases and how actors engage with them. It provides a high-level overview of the system's functionalities and user interactions, aiding communication between stakeholders, designers, and developers. Use-Case diagrams are valuable tools in software development, assisting in the visualization and understanding of system behavior and facilitating the design process by capturing essential user-system interactions.

Tic-Tac-Toe Use Case Diagram



In this case, the AI is not shown as an actor (stick figure) in the design, because it is part of the program being designed. However, a different Use-Case diagram COULD be designed showing the AI as a different actor if this helped to provide more clarity for the people designing the program. The goal of the Use-Case diagram is gain an understanding of the fundamental aspects of the program being designed, and make sure that the final product focuses on those aspects. To that end, this sketch gives an overview of the actions that we will need to provide with in the classes that we design for a game which plays Tic-Tac-Toe.

UML Class Diagrams

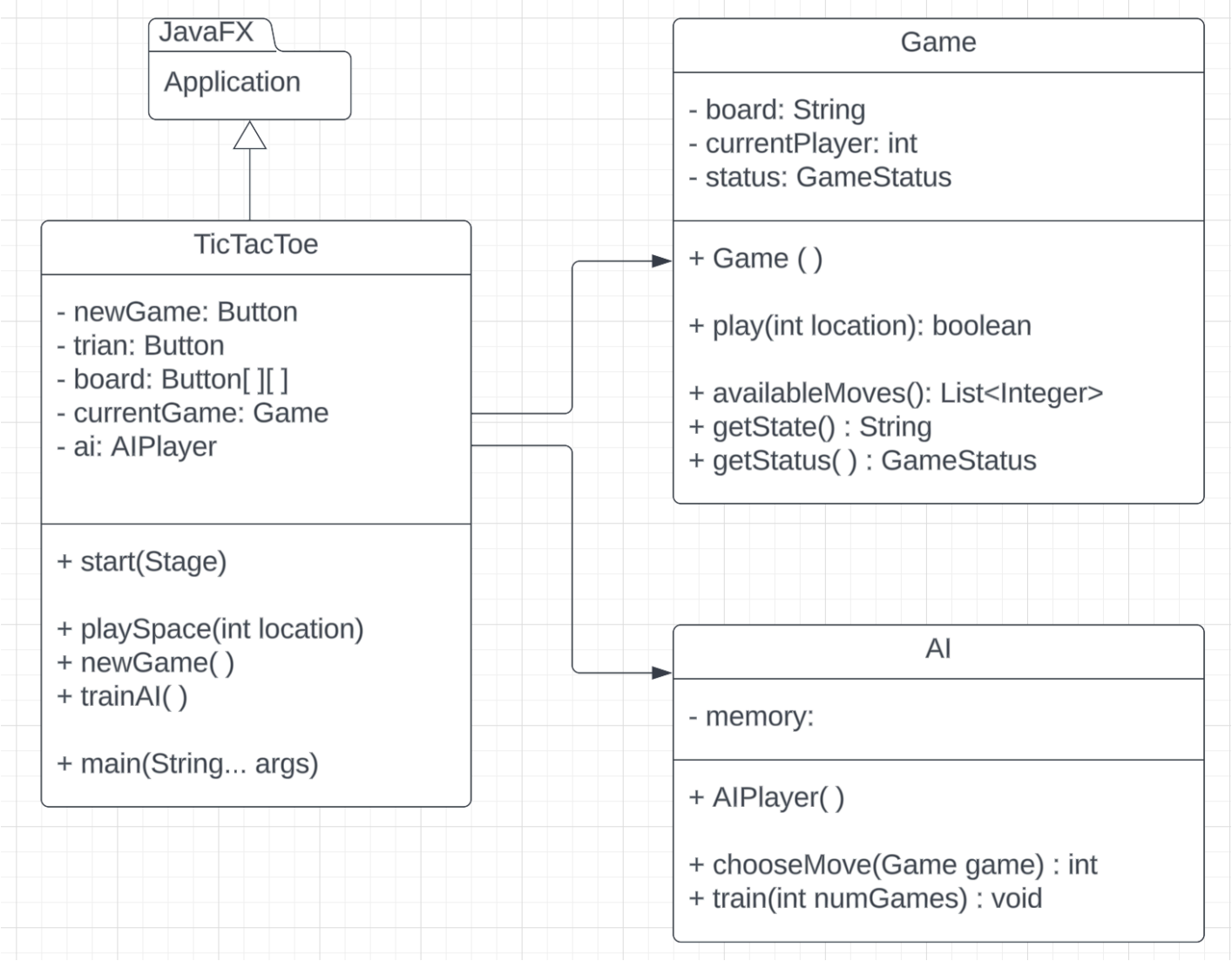
A UML (Unified Modeling Language) Class Diagram is a visual representation that illustrates the structure of a system by depicting classes, their attributes, relationships, and methods. It serves as a blueprint for the static aspects of object-oriented systems.

In a Class Diagram, classes are represented as rectangles with three compartments: the top compartment contains the class name, the middle one lists attributes, and

the bottom one details methods. Arrows connecting classes indicate associations, showcasing how different classes are related. Multiplicity notations on these connections define the cardinality of associations, specifying the number of instances involved.

Class diagrams are instrumental in visualizing the overall system architecture, aiding in software design and documentation. They provide a clear and concise overview of the relationships among classes, facilitating communication among developers and stakeholders throughout the development process.

As an example, here is a possible class diagram of the Tic-Tac-Toe program.

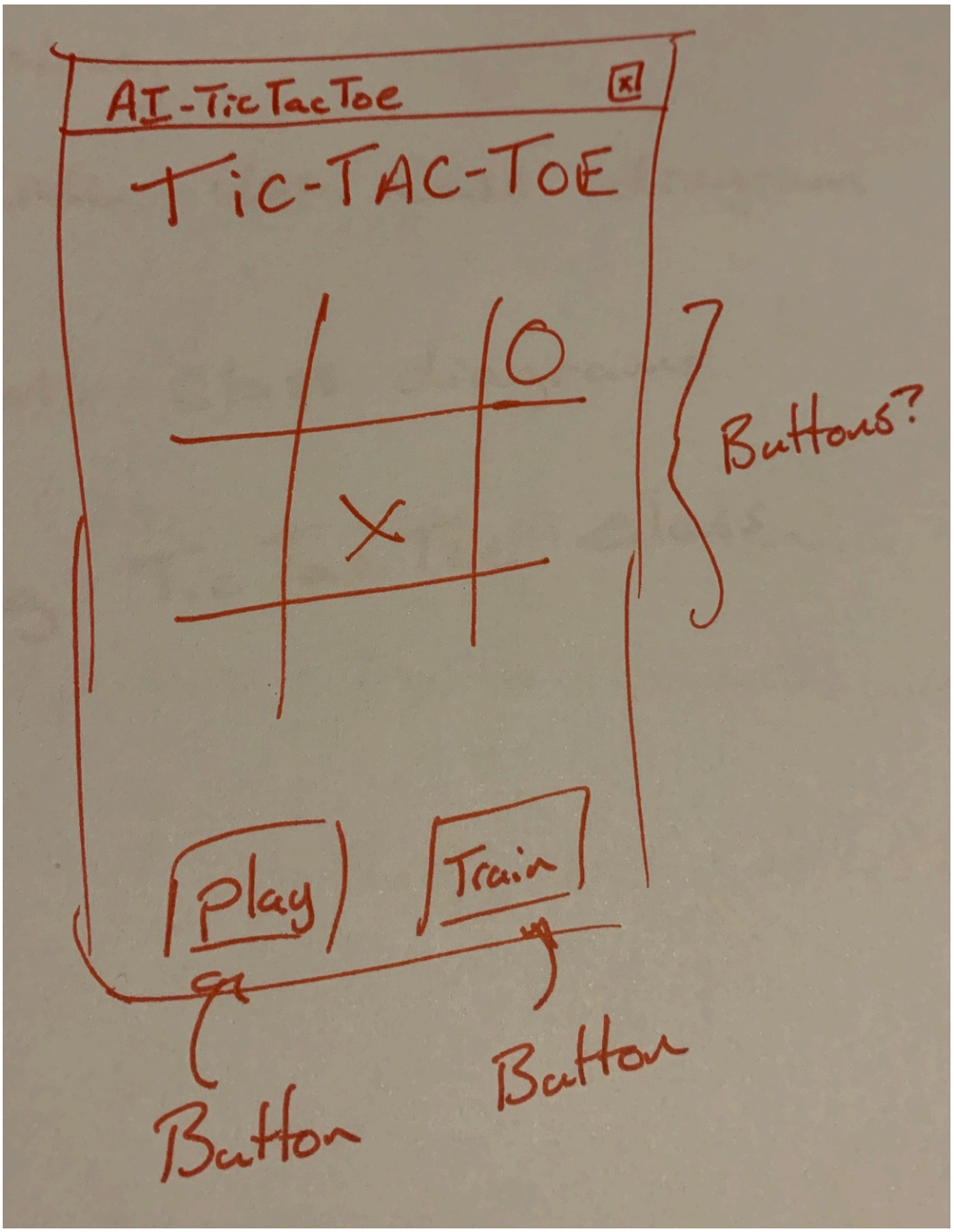


As a first draft these classes provide all of the functionality that would be required to create a functional Tic-Tac-Toe game that meets all of the Use Cases identified in the diagram above.

Class diagrams are meant to be living documents, meaning that as we work on the program, we can return to this class diagram and make changes and updates when we discover changes that need to be made to the program design.

GUI Plan

As a final planning step, it is a worth while to mock up a potential graphical user interface.



Again the goal is to make sure that all the identified use-cases are covered by the potential interface. As the project progresses, we may choose to add additional features, but at least for now all of the required functionality is present (create new games, play a game, train the AI).

Since the creation of GUIs was covered extensively in the first unit, creating this class will not be covered in detail in these lessons. A picture of the JavaFX version of the GUI planned above (running in Windows) is shown for reference.