ASSIGNMENT 5: IMPLEMENTATION OF OOAD FOR TIC-TAC-TOE GAME

1. **ANALYZE**: Write minimum requirements for app:

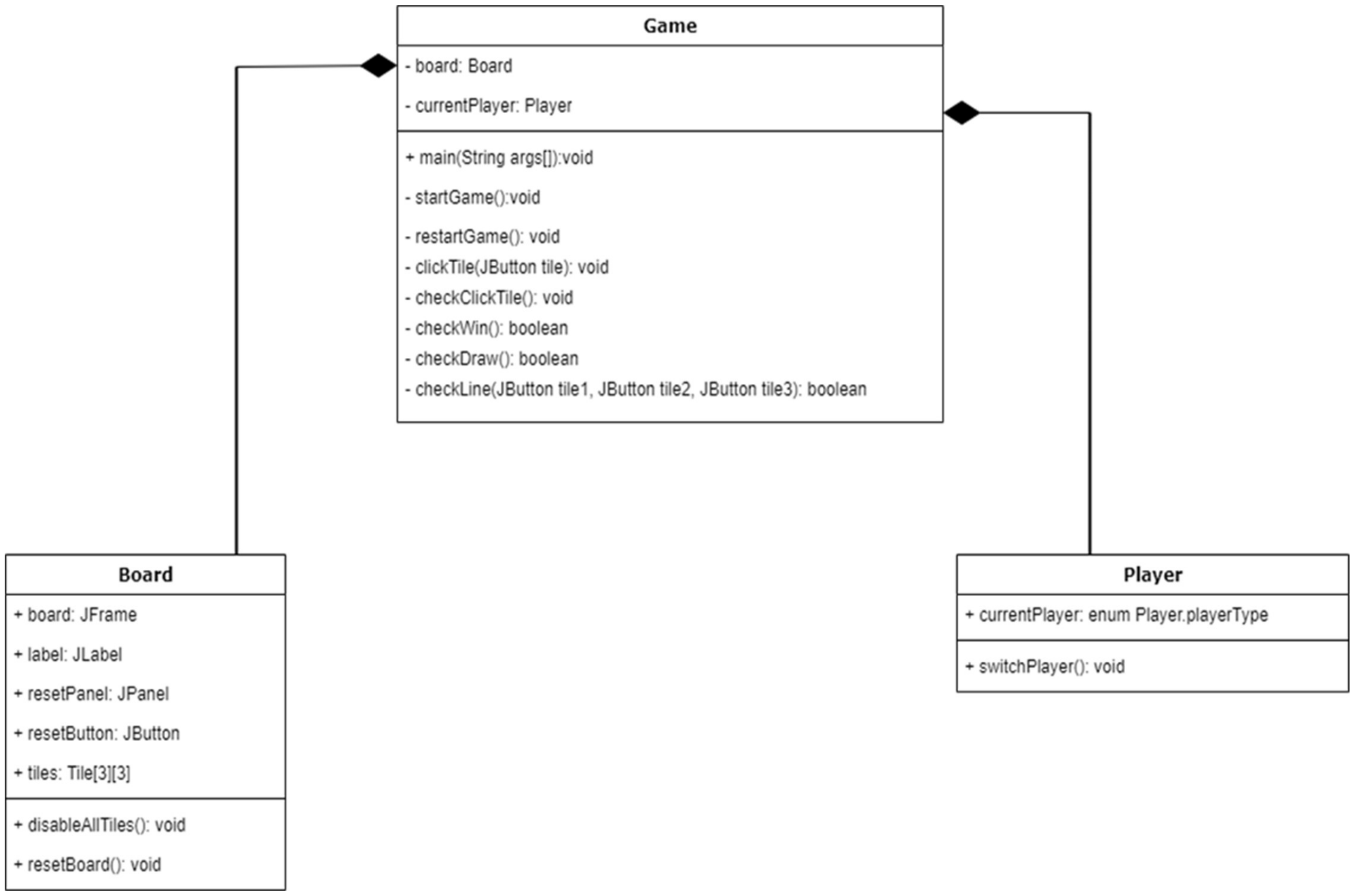
Play a 3x3 grid **Tic Tac Toe game** in GUI. Two-**player** mode where both **players** take turns.
Game UI has a **board**, which includes **title**, **restart button** and 3x3 **tiles** for gaming play.
Starting **game** by initialize 3x3 blank **tiles**.

First **player** is 'X', second **player** is 'O'.
When current **player** is in turn, clicking **tile** will display 'X' or 'O' and change to next player's turn.
After each clicking **tile**, check win conditions.

If win conditions is true, and display which **player** is winner.
Else if, check draw conditions is true, then display 2 players are draw.
Else, display next **player** turn.

Click **restart button** to replay **game**.

2. **DESIGN**: Make UML Class diagram



3. Pros and cons of OOAD?

Pros:

- Improved modularity: OOAD encourages the creation of small, reusable objects that can be combined to create more complex systems, improving the modularity and maintainability of the software.
- Better abstraction: OOAD provides a high-level, abstract representation of a software system, making it easier to understand and maintain.
- Improved communication: OOAD provides a common vocabulary and methodology for software developers, improving communication and collaboration within teams.
- Reusability: OOAD emphasizes the use of reusable components and design patterns, which can save time and effort in software development by reducing the need to create new code from scratch.
- Scalability: OOAD can help developers design software systems that are scalable and can handle changes in user demand and business requirements over time.
- Maintainability: OOAD emphasizes modular design and can help developers create software systems that are easier to maintain and update over time.
- Improved software quality: OOAD emphasizes the use of encapsulation, inheritance, and polymorphism, which can lead to software systems that are more reliable, secure, and efficient.

Cons:

- Complexity: OOAD can add complexity to a software system, as objects and their relationships must be carefully modeled and managed.
- Overhead: OOAD can result in additional overhead, as objects must be instantiated, managed, and interacted with, which can slow down the performance of the software.
- Steep learning curve: OOAD can have a steep learning curve for new software developers, as it requires a strong understanding of OOP concepts and techniques.
- Complexity: OOAD can be complex and may require significant expertise to implement effectively. It may be difficult for novice developers to understand and apply OOAD principles.
- Time-consuming: OOAD can be a time-consuming process that involves significant upfront planning and documentation. This can lead to longer development times and higher costs.
- Rigidity: Once a software system has been designed using OOAD, it can be difficult to make changes without significant time and expense.
- Cost: OOAD can be more expensive than other software engineering methodologies due to the upfront planning and documentation required.