|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USE CASE** | **Description** | **Actor** | **Preconditions** | **Results** |
| Start Game | Starts the checkers game, initializes the board and pieces. | Player 1  Player 2 | None | The game board is displayed for both players |
| Move piece | Player moves a piece on the board. | Player 1  Player 2 | Move is valid | The piece moves to the new square |
| Capture piece | Player captures an opponent’s piece. | Player 1  Player 2 | Move is valid | Opponent's piece is removed from the board |
| King piece | A piece reaches the last row and is promoted to king. | Player 1  Player 2 | None | The piece is promoted to a king |
| End game | Game concludes when a player wins. | Player 1  Player 2 | None | The game announces the winner |

**Assignment 3**

**User Stories:**

Start a new game so that I can play checkers with another human player.

I need to move my piece when I go further.

Capture my opponent's pieces.

When I get to the other end of the board, I want my piece to become a “king” piece.

To know when to play next, let’s see whose turn it is.

Sometimes, see when the game is finished so that you tell if you have one or you have lost.

**Pros and Cons of Object Oriented Analysis Design:**

|  |  |
| --- | --- |
| **ADVANTAGES** | **DISADVANTAGES** |
| Uses UML diagrams, making it easier for different teams to communicate effectively. | More complex which may be unnecessary for smaller projects. |
| Inheritance and polymorphism allow systems to scale or modify easily without significant changes. | Time consuming design |
| Makes maintaining and updating the system more straightforward. Changes can be made to single objects without affecting others. | Requires careful design otherwise can cause issues later in the development process. |
| Protects data by bundling, reducing interactions between different parts of the system. | Overuse of abstraction layers may lead to reduced and less performance |
| Reuse of objects and classes across different parts of the system or even across projects. This saves development time and reduces redundancy. | Not ideal for all projects where procedure is more imporant. |

**OOAD:**

|  |
| --- |
| **Game** |
| players: Player  Status: int |
| coins left()  get game status()  get player names()  startGame().  switchTurn().  checkWinner(). |
|  |

|  |
| --- |
| **Board** |
| lenght: int  breadth: int |
| get coin()  movePiece()  make jump()  movable status()  Display the board() |
|  |

|  |
| --- |
| **Player** |
| name: String  Remaining Pieces: int  Turn: int  Mode type: char  Decision to play: Char |
| get name()  get color()  get mode()  view player()  get turn() |
|  |

|  |
| --- |
| **Pieces** |
| color:String(black/red)  IsKing: Boolean  Coins position: array |
| fix position()  get position()  view coins()  set status()  store moves()  store jumps()  update coins position()  promoteToKing(). |
|  |