Object-oriented thinking is like a checkers game system.

1. Description of Functional and Non-functional Requirements for the Selected Sys

Functional Requirements:

Game Board:

An 8x8 board with alternating black and white squares.
Initial setup of checkers: each player has 12 checkers.

Game Rules:

Moves of checkers: moving diagonally one square forward.
Moves of kings: moving diagonally any number of squares.
Capturing checkers: jumping over an opponent's checker to an empty square.
Becoming a king: a checker reaching the opponent's last row.

User Interface:

Display of the board and checkers.
Ability to select and move checkers.
Display of the current turn.

Playing against AI or another player:

Option to play against the computer or another player.
Different difficulty levels of AI.

Game State Management:

Starting a new game.
Saving and loading the current game state.
Undoing the last move.

Non-functional Requirements:

Performance:
Fast and responsive game without delays when moving checkers.
Reliability:
Protection against invalid moves and rule violations.

Usability:
Intuitive user interface.
Cross-platform Compatibility:
Ability to play on different devices and operating systems.

2. Developing Use Cases According to the Requirements

Use Cases:

Start a New Game:
Actors: Player 1, Player 2 (or AI).
Description: Players start a new game, checkers are placed in their initial positions.
Make a Move:
Actors: Player 1, Player 2 (or AI).
Description: A player selects a checker and moves it to a valid position.
Capture an Opponent's Checker:
Actors: Player 1, Player 2 (or AI).
Description: A player jumps over an opponent's checker, capturing it.
Become a Queen:
Actors: Player 1, Player 2 (or AI).
Description: A checker reaching the opposite end of the board becomes a Queen.
End the Game:
Actors: Player 1, Player 2 (or AI).
Description: The game ends when one player has no valid moves left.

3. Identifying Objects, Classes, and Relationships in the System

Objects and Classes:

Game (Game):
Attributes: current board state, current turn.
Methods: start a new game, make a move, check game over.
Board (Board):
Attributes: board cells.
Methods: initialize board, validate move, update state.
Cell (Cell):
Attributes: coordinates, checker (if any).
Checker (Piece):
Attributes: color, is it a king.
Methods: valid moves, become a king.

Player (Player):

Attributes: checker color, player type (human or AI).

Methods: make a move, select a checker.

Объектно-ориентированное мышление похоже на систему игры в шашки.

1. Описание функциональных и нефункциональных требований для выбран

Функциональные требования:

1. Игровое поле:

Доска 8x8 с чередующимися чёрными и белыми клетками.
Расстановка шашек в начале игры: по 12 шашек у каждого игрока.

2. Правила игры:

Ходы шашек: движение по диагонали на одну клетку вперёд.
Ходы дамок: движение по диагонали на любое количество клеток.
Взятие шашек: прыжок через шашку противника на свободную клетку.
Превращение в дамку: достижение шашкой последней горизонтали доски прот

3. Интерфейс пользователя:

Отображение доски и шашек.
Возможность выбирать и перемещать шашки.
Отображение очередности хода.

4. Игра против ИИ или другого игрока:

Возможность выбора игры против компьютера или другого игрока.
Различные уровни сложности ИИ.

5. Управление состоянием игры:

Начало новой игры.
Сохранение и загрузка текущего состояния игры.
Отмена последнего хода.

Нефункциональные требования:

1. Производительность:
Быстрая и отзывчивая игра без задержек при перемещении шашек.
2. Надёжность:
Защита от некорректных ходов и нарушений правил игры.

3. **Удобство использования**:
   Интуитивно понятный интерфейс.
4. **Кроссплатформенность**:
   Возможность игры на различных устройствах и операционных системах.

2. Разработка вариантов использования (use cases) в соответствии с требован

Варианты использования:

1. **Начало новой игры**:
   Акторы: Игрок 1, Игрок 2 (или ИИ).
   Описание: Игроки начинают новую партию, шашки расставляются на стартовы
2. **Сделать ход**:
   Акторы: Игрок 1, Игрок 2 (или ИИ).
   Описание: Игрок выбирает шашку и перемещает её на допустимую позицию.
3. **Взятие шашки противника**:
   Акторы: Игрок 1, Игрок 2 (или ИИ).
   Описание: Игрок совершает прыжок через шашку противника, взяв её.
4. **Превращение в дамку**:
   Акторы: Игрок 1, Игрок 2 (или ИИ).
   Описание: Шашка, достигшая противоположного края доски, превращается в д
5. **Завершение игры**:
   Акторы: Игрок 1, Игрок 2 (или ИИ).
   Описание: Игра заканчивается, когда у одного из игроков больше нет допустим

3. Определение объектов, классов и их связей в системе

Объекты и классы:

1. **Игра (Game)**:
   Атрибуты: текущее состояние доски, очередь хода.
   Методы: начать новую игру, сделать ход, проверить конец игры.
2. **Доска (Board)**:
   Атрибуты: клетки доски.
   Методы: инициализация доски, проверка допустимости хода, обновление состо
3. **Клетка (Cell)**:
   Атрибуты: координаты, шашка (если есть).
4. **Шашка (Piece)**:
   Атрибуты: цвет, является ли дамкой.
   Методы: допустимые ходы, превращение в дамку.

5. Игрок (Player):
Атрибуты: цвет шашек, тип игрока (человек или ИИ).
Методы: сделать ход, выбрать шашку.

ной системы

ıе позиции.

ıых ходов.