Object-oriented thinking is like a chess game system.

1. Describe the requirements for the chosen system: functional and non-functional.

1.1 The functional requirements for a chess game system:

Game Start:

The system should allow the user to start a new game.
Система должна предоставлять пользователю возможность начать новую игру.

It should provide an option for selecting the color of pieces (white or black) for the player.
Должна быть возможность выбора цвета фигур (белых или черных) для игрока.

Piece Movement:

The system should enable players to move chess pieces according to the rules of chess.
Система должна разрешать игрокам выполнять ходы фигурами согласно правилам шахмат.

Validating moves: The system should check whether a move is valid for the selected piece.
Валидация ходов: система должна проверять, является ли ход допустимым для выбранной фигуры.

Turn Switching: *Смена хода:*

The system should switch turns between white and black players after each move.
Система должна переключать ходы между белыми и черными игроками после каждого хода.

Check and Checkmate Detection: *Проверка на шах и мат:*

The system should detect if a king is under check from the opponent's pieces.
Система должна проверять, находится ли король какой-либо из сторон под шахом.

If necessary, the system should determine checkmate and end the game.
При необходимости, система должна определить мат и завершить игру.

Save and Load Game: *Сохранение и загрузка игры:*

The system should provide the capability to save the current state of the game and load a saved game later.
Система должна предоставлять возможность сохранить текущее состояние игры и загрузить сохраненную игру позже.

Board and Piece Display: *Отображение доски и фигур:*

The system should display the chessboard and pieces in their current state during the game.
Система должна отображать игровую доску и фигуры в текущем состоянии игры.

Game End:

The system should announce the game results, such as victory for one of the players, a draw, or the option to continue playing.
Система должна объявлять результаты игры, такие как победа одного из игроков, ничья или продолжение игры.

Multiplayer Mode (Optional): *Многопользовательская игра (опционально):*

If multiplayer functionality is included, the system should support multiple players playing the game over a network.
Если планируется многопользовательская игра, система должна поддерживать игру нескольких игроков через сеть.

1.2 The non-functional requirements for a chess game system:

Performance: *Производительность:*

The system should provide smooth and responsive gameplay, ensuring that piece movements and game state updates happen instantaneously or with minimal delay.
Система должна обеспечивать плавную и отзывчивую игровую динамику, гарантируя, что перемещения фигур и обновления игрового состояния происходят мгновенно или с минимальной задержкой.

Security: *Безопасность:*

For online games, if applicable, the system should ensure secure communication between players and protect against potential attacks or fraud.
Для онлайн-игр, если таковые есть, система должна обеспечивать безопасное общение между игроками и защиту от потенциальных атак или мошенничества.

Data Protection: User personal data, if stored, should be safeguarded against unauthorized access.
Защита данных: Личные данные пользователей, если они хранятся, должны быть защищены от несанкционированного доступа.

Reliability: *Надежность:*

The system should be stable and reliable, minimizing potential crashes or errors in the application.
Система должна быть стабильной и надежной, минимизируя возможные сбои или ошибки в работе приложения.

Scalability: *Масштабируемость:*

If a large number of concurrent games or numerous players are anticipated, the system should be able to scale to provide quality service to all players.
Если предполагается большое количество одновременных игр или множество игроков, система должна быть способной масштабироваться, чтобы обеспечивать качественное обслуживание всех игроков.

Compatibility: *Совместимость:*

The application should be compatible with various devices and browsers if it is intended to work in a web browser.
Приложение должно быть совместимо с разными устройствами и браузерами, если оно предназначено для работы в браузере.

User-Friendliness: *Удобство использования:*

The game interface should be intuitive and user-friendly for players of all ages and experience levels.
Интерфейс игры должен быть интуитивно понятным и удобным для пользователей всех возрастов и

Design and Graphic: *Дизайн и графика:*

The game's graphical design should be visually appealing and in line with the theme of chess.
Графическое оформление игры должно быть привлекательным и соответствовать тематике шахмат.

Language Localization: *Языковая локализация:*

If multi-lingual support is planned, the system should accommodate different languages.
Если предполагается многолокализационность, система должна поддерживать разные языки.

Loading Speed: *Скорость загрузки:*

The game should load quickly, especially when starting a new game.
Игра должна быстро загружаться, особенно при старте новой игры.

Cross-Platform Compatibility (Optional): *Совместимость с разными платформами (опционально):*

If you want the game to be available on different platforms (e.g., Android, iOS, Windows), specify this requirement.
Если вы хотите, чтобы игра была доступна на разных платформах (например, Android, iOS, Windows), укажите это требование.

2. Design use cases for the system based on the requirements.

1. Use Case: Start a New Game / *Варианты использования: Начать новую игру:*

Actor: Player

Description: The player initiates a new game.

Actions:

- The player selects to start a new game.

- The system provides an option to choose the color of pieces (white or black).

2. Use Case: Make a Move / Выполнить ход:

Actor: Player

Description: The player makes a move with a chess piece.

Actions:

- The player chooses a piece to move.
- The player selects a destination for the move.
- The system validates the move's validity.
- The system updates the game state.

3. Use Case: Check and Checkmate Detection / Проверка на шах и мат:

Actor: System

Description: The system checks if a king from either side is in check or checkmate.

Actions:

- The system analyzes the current state of the board and chess pieces.
- The system determines if there is a threat of check to the king.
- The system identifies if this constitutes a checkmate condition.

4. Use Case: Save and Load Game

Actor: Player

Description: The player saves the current game state and loads a previously saved game.

Actions:

- The player selects to save the game and specifies a save name.
- The player chooses to load a game and selects a previously saved game to continue from.

5. Use Case: End the Game

Actor: System

Description: The game concludes with a result.

Actions:

- The system determines the game result (victory for one side, a draw, etc.).
- The system displays the result and offers options to start a new game or exit the application.

3. Identify objects, classes, and relationships in the system. Optionally, design CRC cards.

3.1 Identify objects, classes in the system.

1. Player: A class representing a player who participates in the game. It can have attributes such as a name and the chosen color of pieces (white or black).
Класс, представляющий игрока, который принимает участие в игре. У него могут быть атрибуты, такие как имя и выбранный цвет фигур (белые или черные).

2. Board: A class representing the chessboard. It stores the state of all squares on the board and the pieces on them.
Класс, представляющий шахматную доску. Он хранит состояние всех клеток на доске и фигур на них.

3. Piece: A base class for all chess pieces (pawn, rook, queen, etc.). Specific classes for each type of piece can inherit from this base class.
Базовый класс для всех шахматных фигур (пешка, ладья, ферзь и т.д.). От этого класса могут наследоваться классы для каждой конкретной фигуры.

4. Pawn, Rook, Queen, etc.: Classes representing specific chess pieces. They can have unique attributes and methods for validating their moves.
Пешка (Pawn), Ладья (Rook), Ферзь (Queen) и т.д.: Классы, представляющие конкретные фигуры. Они могут иметь уникальные атрибуты и методы для проверки их ходов.

3.2 Relationships in the system.

1. "Has-A" Relationship: The "Board" class contains pieces on its squares. In other words, the "Board" class has an aggregation to the "Piece" classes.
Отношение "Содержит" (Has-A): Класс "Доска" содержит фигуры на клетках. То есть, у класса "Доска" есть агрегация к классам "Фигура".

2. "Belongs-To" Relationship: The "Piece" class belongs to a specific player, as each side has its own set of pieces.
Отношение "Принадлежит" (Belongs-To): Класс "Фигура" принадлежит конкретному игроку, так как каждая сторона имеет свои фигуры.

4. Optionally, design CRC cards. (Class-Responsibility-Collaboration)

1. Class: Player

Responsibilities: *Обязанности:*

- Select the color of pieces (white or black)
Выбрать цвет фигур: белые или черные.

- Make moves during the game
Делать ходы во время игры.

Collaborations: *Сотрудничество:*

- Collaborates with the Board class to execute moves
Сотрудничает с классом "Доска" для выполнения ходов.

2. Class: Board

Responsibilities: *Обязанности:*

- Maintain the state of all squares on the chessboard
Сохранять состояние всех клеток на шахматной доске.

- Validate moves made by players
Проверять ходы, сделанные игроками.

Collaborations: *Сотрудничество:*

- Collaborates with Piece classes for move validation
Сотрудничает с классами фигур для проверки ходов.

- Collaborates with the Player class to manage the game
Сотрудничает с классом "Игрок" для управления игрой.

3. Class: Piece (Base Class)

Responsibilities: *Обязанности:*

Define the basic attributes and behaviors common to all chess pieces
Определение основных атрибутов и поведения, общих для всех шахматных фигур.

Collaborations: *Сотрудничество:*

May have subclasses such as Pawn, Rook, Queen, etc., which inherit from it
Может иметь подклассы, такие как "Пешка", "Ладья", "Ферзь" и т.д., которые наследуют его.

4. Class: Pawn (Класс "Пешка")

Responsibilities: *Обязанности:*

Implement the specific rules for the pawn's movement
Реализация конкретных правил движения пешки.

Collaborations: *Сотрудничество:*

Collaborates with the Board class for move validation
Сотрудничает с классом "Доска" для проверки ходов.

5. Class: Rook (Класс "Ладья")

Responsibilities: *Обязанности:*

Implement the specific rules for the rook's movement
Реализация конкретных правил движения ладьи.

Collaborations: *Сотрудничество:*

Collaborates with the Board class for move validation Сотрудничает с классом "Доска" для проверки ходов.
6. Class: Queen (Класс "Ферзь")
Responsibilities: Обязанности:
Implement the specific rules for the queen's movement Реализация конкретных правил движения ферзя.
Collaborations: Сотрудничество:
Collaborates with the Board class for move validation Сотрудничает с классом "Доска" для проверки ходов.
7. Class: Bishop (Класс "Слон")
Responsibilities: Обязанности:
Implement the specific rules for the bishop's movement Реализация конкретных правил движения слона.
Collaborations: Сотрудничество:
Collaborates with the Board class for move validation
8. Class: Knight (Класс "Конь")
Responsibilities: Обязанности:
Implement the specific rules for the knight's movement
Collaborations: Сотрудничество:
Collaborates with the Board class for move validation
9. Class: King (Класс "Король")
Responsibilities: Обязанности:
Implement the specific rules for the king's movement
Detect and handle check and checkmate situations Обнаружение и обработка ситуаций "шах" и "мат".
Collaborations: Сотрудничество:
Collaborates with the Board class for move validation
10. Class: Check and Checkmate (Класс "Шах" и "Мат")
Responsibilities: Обязанности:
Identify when a king is in check and checkmate Определение, когда король находится под "шахом" и "матом".
Collaborations: Сотрудничество:
Collaborates with the King class to determine check and checkmate conditions Сотрудничает с классом "Король" для определения условий "шах" и "мат".