Умова

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. Опишіть вимоги до обраної системи: функціональні та нефункціональні.
2. Спроектуйте варіанти використання системи на основі вимог.
3. Визначте об’єкти, класи та зв’язки в системі. За бажанням оформіть картки CRC.
4. Створіть діаграми класів, що відображають класи, їхні атрибути та зв’язки в системі.
5. За бажанням можна конвертувати діаграми в код Java.
6. Надішліть такі результати: вимоги, випадки використання, CRC (необов’язково), діаграми класів, код Java класів.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

*1.Опишіть вимоги до обраної системи: функціональні та нефункціональні.*

**Функціональні вимоги**

**1.Реалізація базових правил гри**

**Правила руху фігур:**

Кожна фігура (пішак, кінь, слон, тура, ферзь, король) має свої унікальні правила руху.

Система має перевіряти коректність ходу згідно з правилами шахів.

**Ходи гравців:**

Користувач може вибрати фігуру, яку хоче перемістити, і вказати цільову клітинку. Система визначає валідність ходу та виконує його.

**Спеціальні ходи:**

* **Рокіровка:** довга і коротка.
* **Перетворення пішака:** при досягненні останньої горизонталі.

**Визначення кінця гри:**

* **Шах:** система визначає, чи знаходиться король під загрозою.
* **Шах-мат:** гра завершується, якщо король не може уникнути шаху.
* **Пат:** гра завершується в разі відсутності можливих ходів у гравця.
* **Нічия:** гра завершується у випадках, передбачених правилами шахів (наприклад, повторення позиції тричі).

**2. Моделі гри**

Гра проти AI: Система має рівні складності AI (початковий, середній, просунутий)

**3. Збереження і відновлення гри**

**Збереження стану гри:** Можливість зберегти гру з усіма поточними даними (позиції фігур, історія ходів, активний гравець).

**Відновлення гри:** Завантаження збереженої гри для продовження.

**4. Історія ходів**

* Відображення всіх ходів гравців у форматі шахової нотації.

**5. Взаємодія з користувачем**

**Повідомлення:** Сповіщення про шах, шах-мат, пат або помилковий хід.

**6. Аналітика:**

Підрахунок статистики гри: Час, витрачений на хід, загальний час гри.

### ****Нефункціональні вимоги****

#### ****1. Швидкодія****

* Система має обробляти кожен хід менш ніж за 0.5 секунди.
* Швидка перевірка шаху, шах-мату або пату.

#### ****2. Надійність****

* Захист від втрати даних: Якщо програма аварійно завершує роботу, стан гри повинен зберігатися.
* Перевірка введення користувача для уникнення некоректних дій.

#### ****3. Масштабованість****

* Система повинна підтримувати розширення функціоналу:
  + Наприклад, додавання нового рівня складності AI або додавання нової фігури

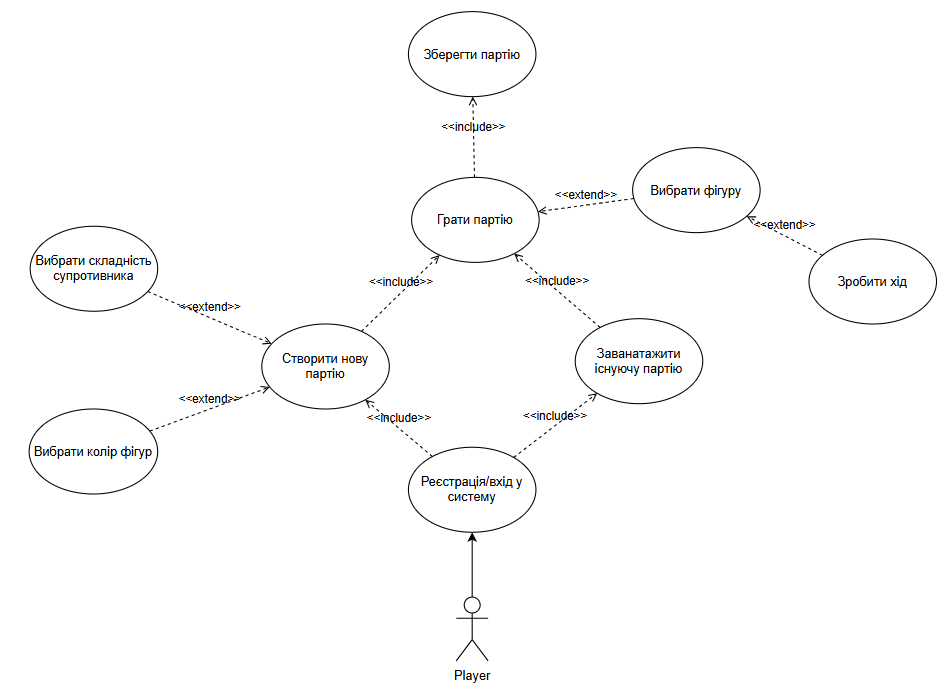
#### ****4. Кросплатформеність****

* Система повинна працювати на Windows, macOS, Linux.
* Використання універсальних технологій (наприклад, Java).

#### ****5. Пам’ять****

* Програма не повинна споживати понад 200 МБ оперативної пам’яті.

*2. Спроектуйте варіанти використання системи на основі вимог.*



*3. Визначте об’єкти, класи та зв’язки в системі. За бажанням оформіть картки CRC.*

### Об'єкти та класи:

**Game**

Відповідає за управління грою: збереження, завантаження, перевірку статусу (шах, шах-мат, нічия тощо).

Зв’язки: Player, Board, MoveValidator, AIPlayer.

**Player**

Уособлює гравця

Зв’язки: Piece, Game

**AIPlayer**

відповідає за обчислення ходів для штучного інтелекту.

Зв’язки: Piece, Game

**Board**

Відповідає за відображення шахової дошки та її стану.

Зв’язки: Piece, Game.

**Piece**

Абстрактний клас для всіх шахових фігур.

Підкласи: Pawn, Rook, Knight, Bishop, Queen, King.

Зв’язки: AIPlayer, Player, Board

**MoveValidator**

Відповідає за перевірку правильності ходу із-за закінчення партії.

Зв’язки: Game

**MoveBehavior**

Відповідає за специфічну логіку руху фігури.

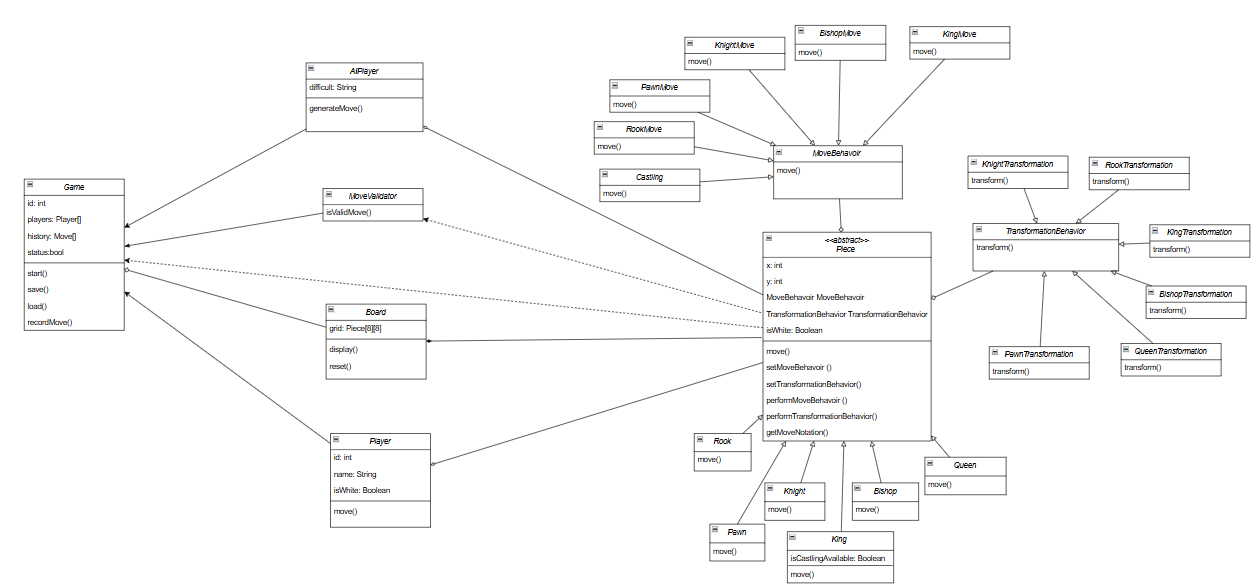
Підкласи: PawnMove, RookMove, KnightMove, BishopMove, KingMove, Сastling

**TransformationBehavior**

Обробляє перетворення пішака або інші зміни фігур.

Підкласи: RookTransformation, QueenTransformation тощо.

*4. Створіть діаграми класів, що відображають класи, їхні атрибути та зв’язки в системі.*



*5. За бажанням можна конвертувати діаграми в код Java*

<https://github.com/Furkalo/OOP_Chess_EPAM>