

Overview document

This version of the overview document should focus on the parser, evaluator, game state, and user interface.

If your document will mostly follow the last version, you can highlight changes clearly therein, so that we can give feedback a bit faster.

- **design: discuss how you plan to implement the project**

- [architecture](#) : identify important classes, packages, or components that make up your design

- **Game State** : แสดงสถานะทั้งหมดของเกม

Variable => board , players, currentTurn , gameStatus, playerBudget

Methods => initialize , updateState(move) , isGameOver, getPlayerBudget(player), getOpponentExpression(minion), getAllyExpression(minion), getNearby(minion, direction)

Variable	Purpose
HexMapGraph board	กระดานหกเหลี่ยม ที่ใช้ในเกม
List<Player> players	รายการของผู้เล่นทั้งหมด (ทั้งคนเล่นและบอท)
Player player1, player2	ตัวแทนของผู้เล่นแต่ละฝ่าย
Player currentPlayer	ติดตามผู้เล่นที่กำลังเล่นอยู่ในปัจจุบัน
int turnCounter	ตัวนับรอบ (เริ่มที่ 1 และเพิ่มขึ้นทุกเทิร์น)

int maxTurns	จำนวนรอบสูงสุดของเกม
GameMode gameMode	โหมดเกม ที่เลือก (Duel , Solitaire , Auto)

Method	Description
setupGame(HexMapGraph map)	กำหนดค่าเริ่มต้นของเกม เช่น โหมดเกม, ผู้เล่น, และโหลดข้อมูลจากไฟล์
startGame()	เริ่มเกมและจัดการลำดับการเล่น
takeTurn(Player player)	ควบคุมการเล่นของแต่ละผู้เล่นในแต่ละรอบ เช่น การรับเงิน, ซื้อช่อง, และวางแผนมินเนี่ยน
isGameOver()	ตรวจสอบว่าเกมจบลงแล้วหรือไม่ (เช่น ครบเทิร์น สูงสุด หรือ ผู้เล่นหมดมินเนี่ยน)
declareWinner()	ประกาศผลแพ้ชนะ โดยดูจากบประมาณที่เหลือ
handleMinionSpawning(Player player)	จัดการการวางแผนมินเนี่ยนของผู้เล่น
chooseMinionTypeForPlayer(Player player)	ให้ผู้เล่นเลือกประเภทมินเนี่ยน ที่ต้องการวางแผน
handleBudget(Player player)	จัดการงบประมาณของผู้เล่น (รวมรายได้ต่อรอบ และดอกเบี้ย)

executeMinionStrategies(Player player)	ดำเนินกลยุทธ์ของมินเนียน ตามสคริปต์ที่กำหนด
handleHexPurchase(Player player)	ให้ผู้เล่นซื้อ hex

- Player แสดงสถานะของผู้เล่น

Variable	Purpose
private final String name	ชื่อของผู้เล่น
public int budget	งบประมาณปัจจุบันของผู้เล่น
public final Set<HexTile> ownedHexes	รายการของช่องหกเหลี่ยม (HexTile) ที่ผู้เล่นเป็นเจ้าของ
private Queue<Minion> minions	คิวของมินเนียนที่ผู้เล่นเป็นเจ้าของ
private final Map<String, Object> context	ข้อมูลบริบท (Context) เช่น งบประมาณที่ใช้ในกลยุทธ์
private boolean hasSpawnedThisTurn	ตรวจสอบว่าผู้เล่นได้วางมินเนียนในรอบนี้แล้วหรือไม่

Method	Description

<code>public String getName()</code>	คืนค่าชื่อของผู้เล่น
<code>public int getBudget()</code>	คืนค่างบประมาณปัจจุบันของผู้เล่น
<code>public void adjustBudget(int amount)</code>	เพิ่มหรือลดงบประมาณของผู้เล่น (แต่ไม่เกินค่าสูงสุด)
<code>public Map<String, Object> getContext()</code>	คืนค่าบันทึกของผู้เล่น เช่น ข้อมูลงบประมาณ
<code>public void startTurn()</code>	รีเซ็ตสถานะการวางแผนมินเนี่ยนเมื่อต้นเทิร์น
<code>public boolean canSpawn()</code>	ตรวจสอบว่าผู้เล่นสามารถวางแผนมินเนี่ยนในรอบนี้ได้หรือไม่
<code>public void spawnMinion(HexTile hex, MinionType minionType)</code>	สร้างมินเนี่ยนและวางลงบนช่องหกเหลี่ยมที่กำหนด
<code>public void initializeHex()</code>	กำหนดช่องเริ่มต้นของผู้เล่นตามตำแหน่งบนกระดาน
<code>public Queue<Minion> getMinions()</code>	คืนค่ารายการมินเนี่ยนของผู้เล่น
<code>handleHexPurchase(Player player)</code>	ให้ผู้เล่นซื้อ hex
<code>public Set<HexTile> getOwnedHexes()</code>	คืนค่ารายการช่องหกเหลี่ยมที่ผู้เล่นเป็นเจ้าของ
<code>public void executeStrategyForMinion(Minion minion)</code>	ดำเนินกลยุทธ์ของมินเนี่ยนที่เลือก

public void setHasSpawned(boolean b)	กำหนดค่าการวางมินเนี่ยนในรอบปัจจุบัน
--------------------------------------	--------------------------------------

- Minion แสดงสถานะของ minion ของผู้เล่น

Variable	Purpose
private final MinionType type	ประเภทของมินเนี่ยน (กำหนดโดย MinionType)
private int health	ค่าพลังชีวิตของมินเนี่ยน
private int defense	ค่าพลังป้องกันของมินเนี่ยน (ขึ้นอยู่กับ MinionType)
private final Player owner	เจ้าของมินเนี่ยน (ผู้เล่นที่เป็นเจ้าของ)
private final Map<String, Object> context	ข้อมูลบริบทที่ใช้ในกลยุทธ์ (เช่น งบประมาณของผู้เล่น)
private int row, col	ตำแหน่งของมินเนี่ยนในกระดาน (แถวและคอลัมน์)

Methods => move(newPosition), attack(target), getPosition(), getStrength(), getHP(), getOwner(), takeDamage(damage), isAlive()

- Command แสดงคำสั่ง (done, move, shoot)

Variable => commandType

Methods => execute(), isValid()

- Parser เป็นตัวประมวลผลคำสั่ง input

Variable	Purpose
private final List<Token> tokens	ชื่อของผู้เล่น
private int currentPosition	ตำแหน่งปัจจุบันของโทเดินที่กำลังถูกพาร์ส
private static HexMapGraph board	กระดานเกมที่ใช้ตรวจสอบตำแหน่งมินเนี่ยนและเงื่อนไขต่างๆ
public Minion currentMinion	มินเนี่ยนที่กำลังดำเนินกลยุทธ์อยู่ในปัจจุบัน

Method	Description
public static HexMapGraph getBoard()	คืนค่ากระดานเกมปัจจุบัน
public Minion getCurrentMinion()	คืนค่ามินเนี่ยนที่กำลังประมวลผลอยู่
private Token getCurrentToken()	คืนค่าโทเดินที่กำลังถูกพิจารณาในขณะนี้
private void consume()	เลือนตำแหน่งไปยังโทเดินถัดไป
private boolean expect(TokenType type)	ตรวจสอบว่าโทเดินปัจจุบันตรงกับประเภทที่คาดหวังหรือไม่
public List<Statement> parse()	เริ่มต้นการแปลงโทเดินเป็นชุดคำสั่ง (AST)

private Statement parseStatement()	แปลงโทเค็นเป็นโครงสร้างคำสั่งตามประเภทของคำสั่ง
private Statement parseAssignment()	แปลงโทเค็นเป็นคำสั่งกำหนดค่า (Assignment Statement)
private Statement parseIfStatement()	แปลงโทเค็นเป็นคำสั่งเงื่อนไข (if-then-else statement)
private Statement parseWhileStatement()	แปลงโทเค็นเป็นคำสั่งลูป (while statement)
private Statement parseActionCommand()	แปลงโทเค็นเป็นคำสั่งดำเนินการ เช่น move, shoot, done
private String parseDirection()	แปลงโทเค็นเป็นทิศทาง (DIR_UP, DIR_DOWN, ฯลฯ)
private Statement parseBlockStatement()	แปลงโทเค็นเป็นบล็อกคำสั่ง { ... }
private Expression parseExpression()	แปลงโทเค็นเป็นนิพจน์ (Expression) ทางคณิตศาสตร์
private Expression parseTerm()	แปลงโทเค็นเป็นตัวประกอบของนิพจน์ (+, -, ถูกประมวลผลหลัง *, /, %)
private Expression parseFactor()	แปลงโทเค็นเป็นตัวประกอบที่อาจมี ^ (ยกกำลัง)
private Expression parsePrimary()	แปลงโทเค็นเป็นค่าหลัก (ตัวเลข, ตัวแปร, นิพจน์ใน

	วงเล็บ)
private Expression parseInfoExpression()	แปลงโทเค็นเป็นคำสั่งตรวจสอบสภาพ (<code>opponent</code> , <code>ally</code> , <code>nearby</code>)
private boolean isActionCommand(Token token)	ตรวจสอบว่าโทเค็นเป็นคำสั่ง (<code>move</code> , <code>shoot</code> , <code>done</code>) หรือไม่
private boolean isDirection(Token token)	ตรวจสอบว่าโทเค็นเป็นทิศทาง (<code>DIR_UP</code> , <code>DIR_DOWNLEFT</code> ฯลฯ) หรือไม่
private boolean isInfoExpression(Token token)	ตรวจสอบว่าโทเค็นเป็นคำสั่งตรวจสอบสภาพ (<code>opponent</code> , <code>ally</code> , <code>nearby</code>) หรือไม่

- Evaluator คำนวณ minion strategies และ ผู้เล่น

Variable => strategyRules

Methods => evaluateMinion(minion), evaluatePlayer(player), generateReport()

Board Package เป็นส่วนของการแสดง Hex 8x8 บน Terminal ประกอบด้วย

Direction Enum, HexMapGraph Class, HexTile Class

Board:	(1,2)[-]	(1,3)[-]	(1,4)[-]	(1,5)[-]	(1,6)[-]	(1,7)[-]	(1,8)[-]
(1,1){Cora}{P1}	(2,2){P1}	(2,3)[-]	(2,4)[-]	(2,5)[-]	(2,6)[-]	(2,7)[-]	(2,8)[-]
(2,1)[-]	(3,2)[-]	(3,3)[-]	(3,4)[-]	(3,5)[-]	(3,6)[-]	(3,7)[-]	(3,8)[-]
(3,1)[-]	(4,2)[-]	(4,3)[-]	(4,4)[-]	(4,5)[-]	(4,6)[-]	(4,7)[-]	(4,8)[-]
(4,1)[-]	(5,2)[-]	(5,3)[-]	(5,4)[-]	(5,5){P2}	(5,6)[-]	(5,7)[-]	(5,8)[-]
(5,1)[-]	(6,2)[-]	(6,3)[-]	(6,4)[-]	(6,5)[-]	(6,6)[-]	(6,7)[-]	(6,8)[-]
(6,1)[-]	(7,2)[-]	(7,3)[-]	(7,4)[-]	(7,5)[-]	(7,6)[-]	(7,7)[-]	(7,8)[-]
(7,1)[-]	(8,2)[-]	(8,3)[-]	(8,4)[-]	(8,5)[-]	(8,6)[-]	(8,7)[-]	(8,8){Cora}{P2}
(8,1)[-]							

Minion Package ประกอบด้วย

Minion Class, MinionType Class

Parser Package ประกอบด้วย

AST Package, Parser Class, SyntaxError Exception

AST Package ประกอบด้วย

ActionCommand Class, AssignmentStatement Class, ASTNode Class, AttackCommand Class, BinaryExpression Class, BlockStatement Class, Command Class, DoneCommand Class, Expression Class, IdentifierExpression Class, IfStagement Class, InfoExpression Class, MoveCommand Class, NumberExpression Class, Statement Class, WhileStatement Class

Player Package ประกอบด้วย

Bot class, Player Class

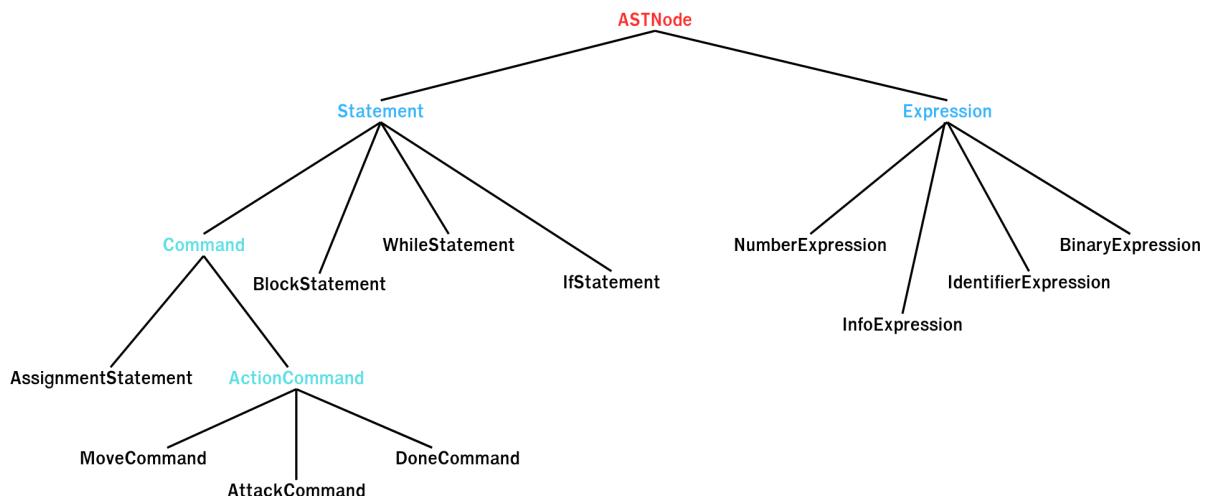
Tokenizer Package ประกอบด้วย

Token Class, Tokenizer Class, TokenType Enum

Class อื่นๆ ที่ไม่ได้อยู่ใน Package ใดๆ

ConfigLoader Class, GameSetup Class

- show how classes are composed and interact to accomplish design goals
(a module dependency diagram is suggested)



ASTNode handle the evaluation.

- present key interfaces provided by classes, and discuss any design patterns that you plan to use and why you choose them

- discuss any important rep invariants maintained by classes

GameState:

ความถูกต้องของ Board (Board Consistency)

- board != null : ในการสร้าง Board จะต้องเป็น null หลังจากเริ่มเกม

- HexMapGraph : จะต้องไม่สามารถเปลี่ยนแปลงได้

Player Management

- players.size() == 2 : ต้องมีผู้เล่น 2 คนเสมอ ไม่ว่าจะอยู่ Mode ไหน (Duel, Solitaire, Auto)

- player1 != null && player2 != null : ผู้เล่นทั้ง 2 ฝ่ายต้อง set up ค่า Minion ก่อนเริ่มเล่นเกม

- currentPlayer : เป็นการขี้ไปยัง player1 หรือ player2 ตลอดการเล่นเกม

Turn Management

- turnCounter >= 1 : เป็นตัวนับรอบที่มีค่ามากกว่าหรือเท่ากับ 1 และต้องเพิ่มขึ้นอย่างถูกต้องหลังจบรอบของแต่ละผู้เล่น

- turnCounter <= maxTurns : เป็นตัวนับรอบที่ต้องไม่เกินจำนวนรอบสูงสุด (maxTurns) ที่กำหนดในไฟล์ config

ความถูกต้องของ Mode (Game Mode Integrity)

- gameMode != null : ต้องมีการกำหนดโหมดเกมที่ถูกต้องเสมอ (Duel, Solitaire, Auto)

ความถูกต้องของ Minion และ HexTile (Minion and HexTile Consistency)

- Minion ทุกด้วยต้องเป็นของผู้เล่นที่ถูกต้องและอยู่บน HexTile ที่ผู้เล่นเป็นเจ้าของ

- ห้ามมี Minion มากกว่า 1 ตัวอยู่ใน HexTile เดียวกัน

- HexTile.isOccupied() ต้องเป็น false ก่อนทำการ Spawn Minion ใหม่

- HexTile.isBought() == false : ผู้เล่นสามารถซื้อ Hex ไดเฉพาะที่อยู่ติดกับ Hex ที่ตัวเองเป็นเจ้าของและยังไม่ถูกซื้อ

ความถูกต้องของ Budget (Budget Consistency)

- player.budget >= 0 : Budget ของผู้เล่นต้องไม่ติดลบหลังจากซื้อ

- player.budget <= ConfigLoader.get("max_budget") : Budget ของผู้เล่นต้องไม่เกินค่าสูงสุดที่กำหนดไว้ในไฟล์ config

- เงื่อนไขจบเกม (Game Over Conditions)

- เกมจะจบเมื่อ : turnCounter > maxTurns หรือผู้เล่นคนใดคนหนึ่งไม่มี Minion เหลืออยู่ (player.getMinions().isEmpty())

GameSetup:

- กำหนดชนิด Minion

- $1 \leq \text{minionTypes.size()} \leq 3$: ต้องมีชนิด Minion 1 ถึง 3 ตัวตามที่ผู้เล่นเลือก

- แต่ละ MinionType ต้องมีชื่อ, defenseFactor ≥ 0 , Strategy (strategy != null && !strategy.isEmpty())

- ตั้งค่าผู้เล่น (Player Initialization)

- player.getOwnedHexes().size() ≥ 1 : ผู้เล่นแต่ละคนต้องเริ่มเกมโดยเป็นเจ้าของ Hex อย่างน้อย 1 ช่อง

- ผู้เล่นต้องมี Minion อย่างน้อย 1 ตัวในการ Spawn ก่อนเริ่มเกม

HexMapGraph:

- Board Structure

- board != null : ในการสร้าง Board จะต้องเป็น null หลังจากเริ่มเกม

- board.length == 8 และ board[0].length == 8 : Board ต้องมีขนาด 8x8 เท่านั้น

- Tile Initialization

- $0 \leq \text{row} < 8$ และ $0 \leq \text{col} < 8$: HexTile ทุกช่องต้องมีการกำหนดพิกัดที่ถูกต้อง

- Valid Position Check

- isValidPosition(row, col) : ตำแหน่งต้องคืนค่าเป็น true เมื่อ Minion อยู่บน Board แต่ถ้าไม่อยู่จะเป็น false

- Ownership Rules

- hex.isBought() == false : ผู้เล่นไม่สามารถซื้อ Hex ที่มีเจ้าของอยู่แล้ว

- ผู้เล่นสามารถซื้อเฉพาะ Hex ที่อยู่ติดกับ Hex ที่ตัวเองเป็นเจ้าของเท่านั้น

HexMapGraph:

- Tile Identity

- แต่ละ HexTile ต้องมีพิกัด (row, col) ที่ไม่ซ้ำกันใน Board เดียวกัน

- Minion Occupancy

- HexTile หนึ่งช่องสามารถมี Minion ได้ 1 ตัวเท่านั้น (`isOccupied() == true` คือมี Minion อยู่แล้ว)

Minion:

- Basic Attributes

- `owner != null` : ทุก Minion ต้องมีเจ้าของที่ชัดเจน (Player)

- HP

- `health >= 0` : HP ต้องไม่ติดลบ
- `health == 0` : ถ้า HP เท่ากับ 0 ถือว่าตาย

- Defense

- `defense >= 0` : Defense ต้องเป็นค่าบวกหรือมากกว่าเท่ากับ 0 ขึ้นไป

Parser:

- Token Handling

- `tokens != null` : ต้องมี Token ที่ถูกต้องเพื่อนำไป Parsing
- Token สุดท้ายต้องเป็น EOF เพื่อระบุจุดสิ้นสุดของโคด

- Parsing Validity

- Statement ที่ถูก Parse ต้องถูกต้องตาม Grammar
- การ Parse if, while, assignment ต้องมีวงเล็บและเครื่องหมายเท่ากับครบถ้วนตามเงื่อนไข

- Direction Validity

- ทิศทางที่รับเข้าต้องมีเหล่านี้ UP, DOWN, UPLEFT, UPRIGHT, DOWNLEFT, DOWNRIGHT

- Strategy Parsing

- Strategy ที่ถูก Parse ต้องไม่เป็น null และต้องคืนรายการของ Statement ที่ถูกต้อง

AST:

- BinaryExpression

- `left != null && right != null` : ต้องมี Expression ฝั่งซ้ายและขวาที่ถูกต้อง
- operator ต้องเป็นหนึ่งใน +, -, *, /, %, ^ เท่านั้น

- การหารต้องไม่เกิดการหารด้วย 0

- $\text{Long.MIN_VALUE} \leq \text{result} \leq \text{Long.MAX_VALUE}$: ผลลัพธ์ของ evaluate() ต้องอยู่ในขอบเขตของค่า Long

- Evaluate Integrity

- ทุก Expression ต้องคืนค่าที่ถูกต้องตามประเภท เช่น NumberExpression ต้องคืนค่าเลขคงที่ตรงกับ Token ที่ถูก Parse

Player:

- Budget Validity

- $\text{budget} \geq 0$: Budget ของผู้เล่นต้องไม่ติดลบ

- $\text{budget} \leq \text{ConfigLoader.get("max_budget")}$: Budget ต้องไม่เกินค่าที่กำหนดใน Config

Token:

- Token Type Validity

- Token ที่เป็น NUMBER ต้องมีค่าที่เป็นตัวเลขที่ถูกต้อง

- Token ที่เป็น IDENTIFIER ต้องเป็นตัวอักษรหรือตัวเลขที่สามารถเป็นตัวแปรได้

Tokenizer:

- Input Validity

- String input ต้องไม่เป็น null

- Tokenization Process

- ค่าที่อ่านจาก input ต้องถูกกำหนดประเภท Token ที่ถูกต้อง เช่น ตัวอักษร → IDENTIFIER, ตัวเลข → NUMBER, อักษรพิเศษ → OPERATOR หรือ SYMBOL

- Whitespace

- ไม่สนใจช่องว่าง และไม่ควรคืนค่า Token ที่เป็นช่องว่าง

- peek()

- peek() ต้องไม่เปลี่ยนค่าตำแหน่ง currentPosition

- peek() ต้องคืนค่าที่เหมือนกับ nextToken() แต่ไม่ทำให้ตำแหน่งเปลี่ยน

- consume()

- consume(TokenType) ต้องตรวจสอบว่า Token ที่อ่านตรงกับประเภทที่คาดหวัง

- หาก Token ไม่ตรงกัน ต้องโยน RuntimeException และแสดงข้อความที่ถูกต้อง

- End of Input Handling

- hasNextToken() ต้องคืนค่า false เมื่อทำ input จบ

- what did your group learn from designing and implementing the parser, evaluator, and game state?

ในการ parser ต้องคิดว่าจะทำยังไงให้อ่านง่าย และเมื่อมี error จะต้อง debug ง่าย ส่วน evaluator ทำให้รู้และเข้าใจถึงโครงสร้างของ Expression ที่ถูกต้อง ส่วน game state ได้รู้ถึงการเชื่อมโยงข้อมูลของเกม

- code design

- describe data structures that will be used

ใช้ HexMapGraph ในการทำ โดยใช้ 2D Array เพื่อกีบข้อมูล Board แบบ 8x8 และ ใช้ List<HexTile> ในการจัดเก็บตำแหน่ง Hex ที่อยู่รอบ ๆ

- discuss any tradeoffs you made between different design goals, such as between simplicity of code and efficiency of execution

การใช้ List<Statement> แทน Tree ใน AST มีข้อดีคือง่ายกว่าและทำให้ Parsing อ่านง่าย แต่มีข้อเสียคือทำให้วิเคราะห์บางอย่างได้ไม่เต็มประสิทธิภาพเท่าที่ควร

และใช้ Queue<Minion> แทน List<Minion> ในการจัดการลำดับการทำของ Minion มีข้อดีคือเหมาะสมกับการทำเกมประเภทนี้ แต่มีข้อเสียคือต้องค้นหา Minion แบบ O(n)

- tools

- discuss any tools you have used or plan to use for the design and implementation of the project

Design Tools

- Figma : ใช้ในการออกแบบ UI

Development Tools

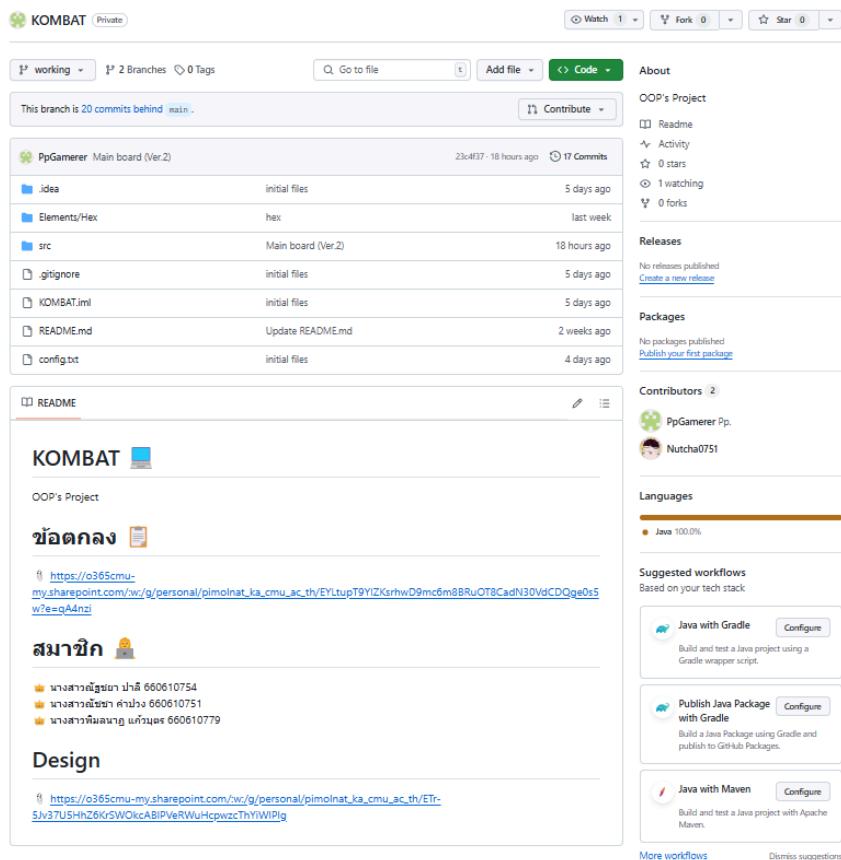
- IDE: IntelliJ IDEA : ใช้ในการเขียนโคด (backend)

- WebSocket : ใช้ในการเชื่อมต่อระหว่าง client และ server (browser and server)

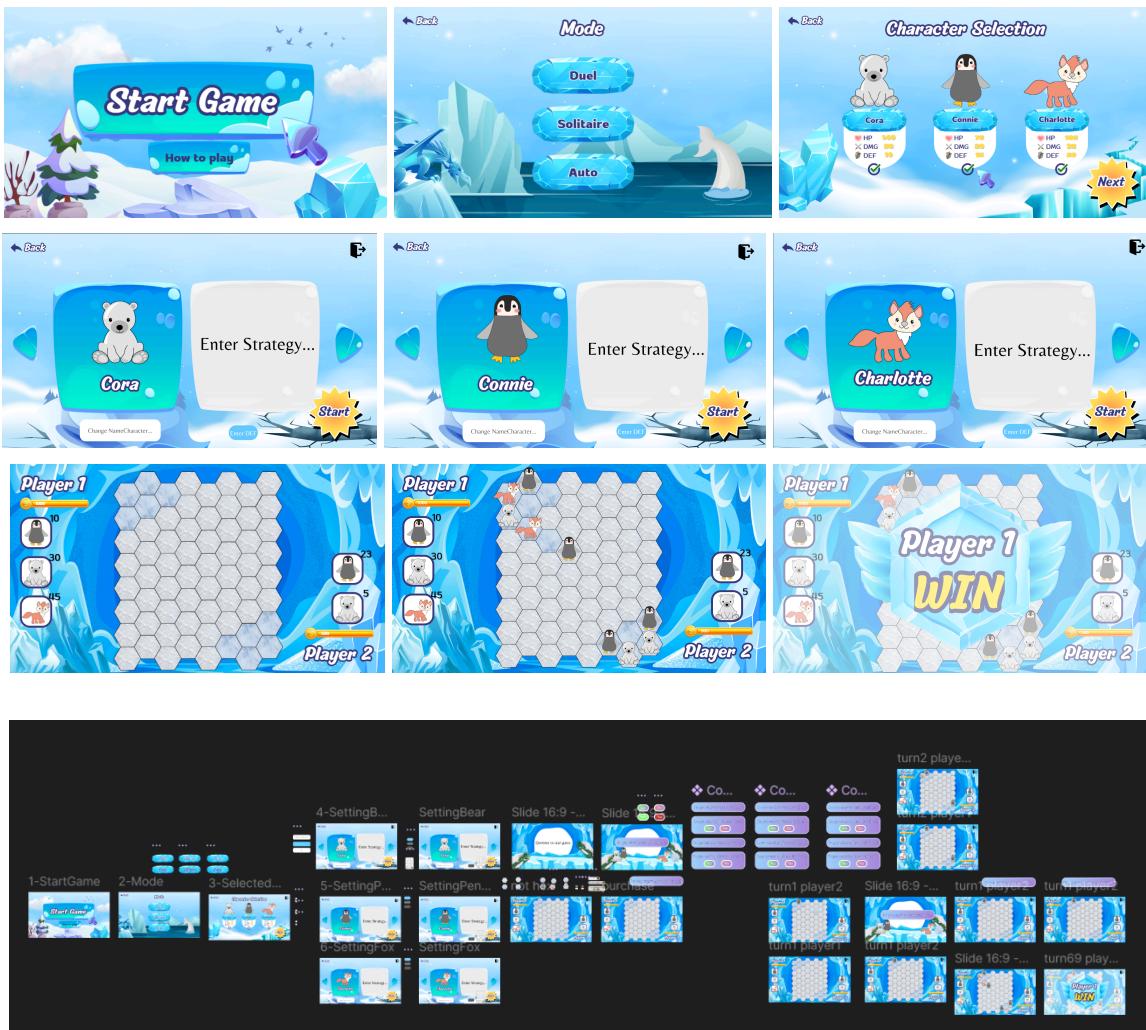
Testing Tools

- JUnit : ใช้ในการ test เพื่อทดสอบการทำงานฟังก์ชันของระบบโปรแกรม

A screenshot of git repository for your team



- a screenshot of your Figma design, and a link to your Figma project (if you decide not to use Figma, explain your reason; it is possible that you find another, more appropriate approach)



เริ่มแรกจะแสดงหน้าที่มีปุ่ม Start Game และ How to play (ยังไม่สามารถกดได้) เมื่อกดปุ่ม Start Game จะนำไปสู่หน้าเลือก Mode ซึ่งมี 3 Mode และเมื่อกด Mode แล้ว ก็จะไปหน้าให้เลือกชนิดของตัวละคร (Minion) หลังจากเลือกได้แล้วก็จะแสดงหน้า Set up ให้ตั้งชื่อตัวละคร (Minion) รวมถึง Strategy ต้า Set up เสร็จแล้วก็สามารถกด Start เพื่อเริ่มเกมได้เลย

Figma : <https://www.figma.com/design/>

- your job is to convince the course staff that you thought carefully about the construction for the project

ได้พยายามจัดระเบียบโครงสร้างให้ดีที่สุดเท่าที่ทำได้แล้ว แต่ก็อาจจะไม่ได้มีประสิทธิภาพเท่าที่ควร TT แต่จะพยายามแก้ไขต่อไปค่ะ

- testing: describe your testing process

- how did you test your parser, evaluator, and game state?

ทดสอบ parser, evaluator, and game state เพื่อให้แน่ใจว่าโค้ดทำงานถูกต้องในผลลัพธ์ต่าง ๆ ที่จะเกิดขึ้นว่ามีข้อผิดพลาดอะไรบ้าง หรือมีอะไรที่ถูกต้องแล้ว

- describe test cases you used for testing

Parser Test เทสในส่วนของ

- Simple Assignment (passed)

ทดสอบคำสั่ง $x = 5 + 3$

ตรวจสอบว่า x ได้ค่า 8 ตามที่คาดไว้

- If-Else Condition (passed)

ทดสอบ `if (1) then x = 10 else x = 20`

ตรวจสอบว่า x ได้ค่า 10 เมื่อเงื่อนไขเป็นจริง

- While Loop (failed)

ทดสอบ `while (x < 3) { x = x + 1 }` เริ่มต้นจาก $x = 0$

ตรวจสอบว่า x หยุดที่ 3 ตามที่คาดไว้

- Move Command (failed)

ทดสอบ loop ที่ให้ minion เคลื่อนที่ 3 ครั้งตามงบประมาณ

ตรวจสอบว่า loop ทำงานครบและบลดลงตามเงื่อนไข.

- Shoot Command (failed)

ทดสอบ loop ที่ให้ minion ยิง 2 ครั้ง (`shoot up cost`)

ตรวจสอบว่า $x = 2$ และบลดลงตามเงื่อนไข

- Nested If Conditions (passed)

ทดสอบ if-else ข้อนี้

ตรวจสอบว่าคำสั่งในเงื่อนไขที่ซ้อนกันทำงานถูกต้อง.

- Complex Strategy (failed)

ทดสอบการใช้ move, shoot และ loop ร่วมกัน

ตรวจสอบว่าลำดับการทำงานและค่าที่ได้ตรงตามที่คาดหวังไว้

Evaluator Test

Evaluator ถูกทดสอบทางอ้อมผ่าน การ execute คำสั่งที่ถูก parse ใน ParserTest

แต่ละการทดสอบช่วยให้มั่นใจว่าการคำนวณ, ลูป, เงื่อนไข, คำสั่ง move และ shoot ทำงานตามที่ตั้งใจไว้

Game State Test

- Initial Budget (passed)

ตรวจสอบว่า Player1 และ Player2 มีงบเริ่มต้น 10,000

- Minion Spawning

จำลองการเล่นของผู้เล่นเพื่อดูว่าด้านบนรอบเพิ่มขึ้นถูกต้อง

ให้ minion เกิดบน hex tile ตรวจสอบว่า tile ถูกครอบครองและผู้เล่นมี minion ในรายการของตน

- Budget Reduction After Spawning

ตรวจสอบว่า budget ลดลงหลังจากเกิด minion

- Maximum Turns Condition

จำลองเกมจนถึงรอบสูงสุด

ตรวจสอบว่าเกมจบลงเมื่อถึงจำนวนรอบที่กำหนด

- Minion Execution

ตรวจสอบว่ามี strategy สำหรับ minion และสามารถ execute ได้โดยไม่มี error

ตรวจสอบว่ากลยุทธ์ไม่เป็นค่าว่างและทำงานได้ตามที่คาด

- describe your testing plan for the remainder of the project

แก้ไขโค้ด และ Test ให้ผ่าน cases ทั้งหมด (Tokenizer ผ่านทั้งหมดแล้ว และ Parser, Evaluator, Game State ผ่านแค่บาง case)

- what did your group learn from testing?

ควรตั้งให้ครอบคลุม edge cases และดูเงื่อนไข input และผลลัพธ์ ของแต่ละส่วนดี ๆ ก่อน

- work plan

- explain the role of member(s) in your group responsible for this part of the project

- Did your group change the division of work? if so, why, what was the change, and how did you end up with that change?

มีการเปลี่ยนแปลง เนื่องจากงานที่ต้องทำมีค่อนข้างเยอะและยาก งานในครั้งนี้จึงจำเป็นต้องมีการแบ่งโดย Figma (UI) 1 คน และโปรแกรม 2 คน ซึ่งมีการแบ่งหน้าที่กันดังนี้

การแบ่งหน้าที่

- ณัฐชยา ปาลี : Figma (UI)
- ณัชชา คำปวง : Design the look of the game, HexMap, TokenizerTest
- พิมลนาฏ แก้วบุตร : Parser class, Evaluator class, Tokenizer, Minion class, Player class, ParserTest

การแบ่งหน้าที่กันทำแบบทำให้งานเดินໄວขึ้น เนื่องจากไม่ต้องรอนัดคนหนึ่งนาน

- what did your group learn from the process of dividing up the work, including changes along the way (if any)?

ได้เรียนรู้ว่าในการแบ่งงานตอนต้นอาจจะไม่ใช่การแบ่งงานที่ดีที่สุดในทุกๆ การทำทำงานแต่ละครั้ง และการแบ่งงานออกเป็นส่วนๆ ก็สำคัญ เพราะจะทำให้ทำงานง่ายขึ้น เป็นขั้นเป็นตอน ในระหว่างทางก็มีการเปลี่ยนแปลงการแบ่งงานด้วย เนื่องจากแต่ละคนเข้าใจในงานส่วนนี้มากกว่า

- known problems (optional)

- detail any known problems with your specification or design (if not already mentioned before)

การจัดระเบียบclasses ต่าง ๆ ให้ไม่ยุ่งยาก ค่อนข้างยาก, Figma มีข้อจำกัดบางส่วนที่ทำให้ไม่สามารถทำได้ตามที่หวัง

- for groups that did not finish this part on time: when do you plan to have this part completed, and why?

จะพยายามให้เสร็จให้เร็วที่สุดก่อนที่จะถึงการส่งงานครั้งหน้า (ไม่ควรเกิน 3 วันหลังจากส่งงานไป) เพื่อให้สามารถเริ่มทำงานที่จะต้องส่งครั้งไปให้เร็วที่สุด

- at this point, do you believe you will be able to finish the entire project on time? if not, why not, and when do you plan to have the entire project completed, and why?

คิดว่าทำเสร็จไม่ทัน เนื่องจากรายละเอียดที่ต้องทำค่อนข้างเยอะและยาก ถ้าจะทำให้เสร็จได้ภายในระยะเวลาประมาณ 1 เดือนเป็นไปได้ค่อนข้างยาก ซึ่งความรู้ความสามารถ รวมถึงประสบการณ์ของกลุ่มเรายังมีไม่มากพอ และถ้าทำไม่ทันที่กำหนดจริงๆ จะพยายามทำให้เสร็จทันภายในระยะเวลาที่ขยายให้ โดยจะพยายามให้มากขึ้น และอาจต้องมีที่ปรึกษา

- problems that are clearly identified and described here will be looked upon more favorably than problems that the course staff discovers later

parser, evaluator ไม่แน่ใจว่าจะทำงานได้ถูกต้องตามที่หวัง, การออกแบบ Game State ยังไม่ลึก入

- **comments (optional): express your opinions about the project**

- this section will not be graded

- you might address these questions:

- how much time did you spend on the design?

ประมาณ 7 ชั่วโมง

- what advice should we have given you before you started?

ภาพรวมใหญ่ ๆ ในการทำเกม ว่ามีอะไรที่ต้องทำบ้าง (ไม่มีประสบการณ์)

- what was surprising about the design?

รายละเอียดในสิ่งที่ต้องทำเยอะกว่าที่คิด

- what was hard about the design?

รายละเอียดในสิ่งที่ต้องทำเยอะกว่าที่คิด และต้องคิดว่าควรทำอะไรก่อนหลัง ทำอย่างไร

- what did you like or dislike about the project?

Like ได้นำสิ่งที่ได้เรียนรู้มาใช้

Dislike ระยะเวลาในการทำค่อนข้างน้อย(อาจเพราะมีภาระงานอื่นด้วย)

- what would you change about the project?

มีการเปลี่ยนแปลงในระหว่างการทำ เนื่องจากบางอย่างไม่เป็นไปตามที่คิด จึงจำเป็นต้องปรับแก้ให้สอดคล้องกับโค้ดส่วนอื่น