

# (Final) Design Overview Document

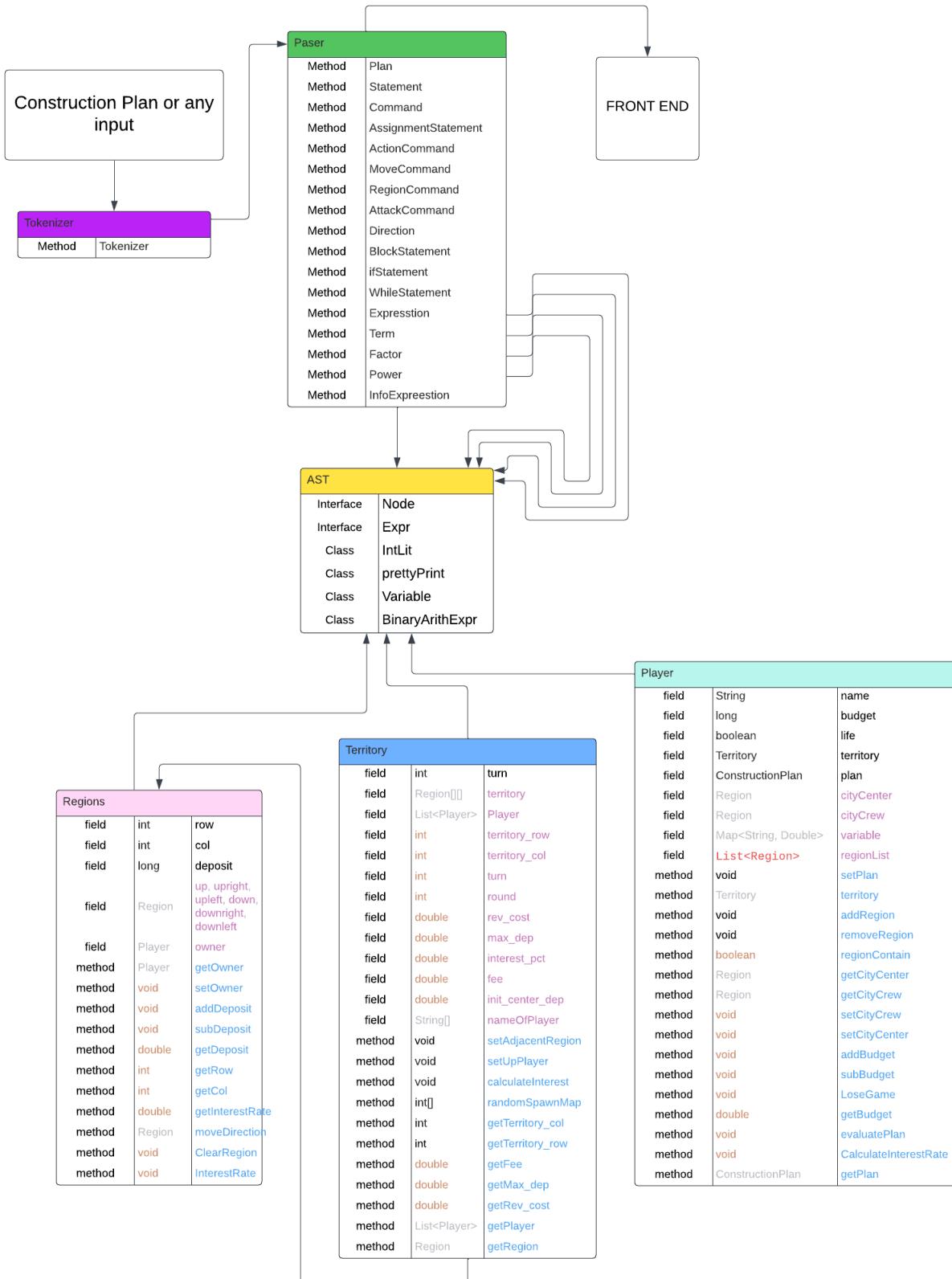
---

## An overview of your server-client implementation

เรื่องแรก จะให้ผู้เล่นใส่ชื่อและสร้าง Config File ที่เป็นห้องของตัวเกมส์ จากนั้นจะทำการยิง API ไปหลังบ้านเพื่อให้สร้าง Map ของเกมส์และส่งกลับมาให้หน้าเว็บ ถ้าผู้เล่นอยู่ในเกมส์แล้ว จะทำการส่ง Construction Plan ที่เป็น String ไปให้หลังบ้าน หลังจากการประมวลผลของ Construction Plan นั้น ๆ แล้ว ก็จะทำการดำเนินการต่าง ๆ เช่น shoot, relocate, move และส่งมาแสดงผลบนหน้า UI

## Architecture:

### module dependency diagram



# Class hierarchy for your game state (model)

## ▼ Player

### field

```
public String name  
public boolean life  
private Territory territory  
private ConstructionPlan plan  
public final Map<String, Double> variable  
private final List<Region> regionList  
private Region cityCenter  
private Region cityCrew;  
  
private double budget = 0;
```

### method

```
public Player(String name, Territory territory, Region cityCenter)  
  
public void setPlan(List<String> text)  
  
public Territory territory()  
  
public void addRegion(Region region)  
  
public void removeRegion(Region region)  
  
public boolean regionContain(Region region)  
  
public double getVariable(String varName)  
  
public void setVariable(String varName, double value)  
  
public Region getCityCenter()  
  
public Region getCityCrew()  
  
public void setCityCrew(Region cityCrew)  
  
public void setCityCenter(Region cityCenter)
```

```
public double getBudget()  
  
public void addBudget(double n)  
  
public void subBudget(double n)  
  
public void LoseGame()  
  
public void evaluatePlan()  
  
public void CalculateInterestRate()
```

## ▼ Regions

### field

```
private int row;  
private int col;  
private double deposit;  
public Region up = null, upright = null, upleft = null, down = null, downright = null,  
downleft = null;  
private Player owner = null;
```

### method

```
public GameState.Player getOwner()  
  
public void setOwner(GameState.Player owner)  
  
public double getDeposit()  
  
public void addDeposit(double n)  
  
public void subDeposit(double n)  
  
public int getRow()  
  
public int getCol()  
  
public double getInterestRate()  
  
public Region moveDirection(String direction)
```

```
public void ClearRegion(Player player)

public void InterestRate()
```

## ▼ Territory

### field

```
private int territory_row
private int territory_col

private double init_budget
private double init_center_dep

private double rev_cost
private final double fee = 1

protected double max_dep
protected double interest_pct;
protected int turn = 1

private int round = 0

private List<Player> Player
protected Region[][] territory

String[] nameOfPlayer
```

### method

```
public Territory(int m, int n, long init_budget, long init_center_dep, long rev_cost,
long interest_pct, long max_dep, String[] nameOfPlayer)
private void setAdjacentRegion()
private void setUpPlayer()
private int[] randomSpawnMap()
public void calculateInterest(Player user)
public int getTerritory_col()
public int getTerritory_row()
public double getFee()
public double getMax_dep()
public double getRev_cost()
public List<GameState.Player> getPlayer()
public Region getRegion(int row, int col)
```

## Design patterns

- Factory pattern
  - ใช้ใน class Territory ใน randomSpawnMap() โดยรวมการสร้างโลจิกในการสุ่มตำแหน่ง spawn
- Observer pattern
  - ใช้ใน class Player ทำหน้าที่เป็น subject ที่ observes state ต่าง ๆ ของเกม และเปิดการทำงานต่าง ๆ เช่น evaluatePlan() ตามการเปลี่ยนแปลง ทำให้จัดการ game state ได้ง่ายขึ้น

## Rep Invariants

- **Player Class:**
  - ตรวจสอบให้แน่ใจว่างบประมาณของ player จะไม่สามารถเป็นค่าที่ติดลบได้
  - จัดการ regions ที่ player เป็นเจ้าของ
- **Region Class:**
  - ควบคุม row และ column อย่างถูกต้อง
  - ตรวจสอบให้แน่ใจว่า deposit จะไม่สามารถเป็นค่าที่ติดลบได้
  - ตรวจสอบให้แน่ใจว่าเจ้าของ region ถูกต้อง
- **Territory Class:**
  - ตรวจสอบให้แน่ใจว่าจำนวนเงินที่ถอนจาก region ทั้งหมด อยู่ในชุดจำกัดที่กำหนดไว้
  - ตรวจสอบให้แน่ใจว่าขอบเขตไม่ใช้ค่าลบและมีความสมเหตุสมผลสำหรับ row และ column
  - ตรวจสอบให้แน่ใจว่า player เป็นเจ้าของ region

## Code Design:

### Data Structures

- Array 2D ในการเก็บ Region แต่ละอันโดยมีขนาดตาม Map m \* n
- Map เอาไว้เก็บตัวแปล ต่างๆที่เกิดขึ้น ใน Construction String —> double
- List เอาไว้เก็บว่ามี Player กี่คนและเอาไว้เก็บข้อมูลที่ใส่มาใน Construction plans

### Tradeoffs

- โดยการรวม เราเก็บ Region เป็น Array ทำให้สามารถเข้าถึงได้ง่าย และเขียน Code ในการทำงานง่าย
- แต่อาจจำทำให้ Code เขียนยากขึ้นเล็กน้อยในการเดินระหว่าง Region และข้อมูลมีการเก็บที่แยกกันทำให้ต้องทั่งใจใช้อาจจะ ยุ่งยากในการเขียน Code

## Tools:

- Notion วางแผนกำหนดการและการทำงาน
- Discord ใช้ติดต่อสื่อสารกับสมาชิกในกลุ่ม
- IntelliJ IDEA ใช้สำหรับการ coding
- Github ใช้สำหรับการอัพเดท Code ของ Project
- Spring Framework
- Postman เช็ค API
- Lucid chart เอาไว้ทำ module dependency diagram

## What did your group learn from designing and implementing parts

เราควรวางแผนในแต่ละส่วนให้ดีก่อนเริ่มต้นการทำงาน เพื่อให้การทำงานเป็นไปได้อย่างราบรื่นและลดโอกาสที่จะเกิดปัญหาต่าง ๆ ลง เช่น การวางแผนว่าจะใช้อะไรในการเก็บผู้เล่น เก็บ city center จะทำการ Parse อย่างไร จะออกแบบ UI อย่างไรให้ดูน่าสนใจ

---

## Git Repository Screenshot

# Testing

## ▼ Expr Test

ในการ test `BinaryArithExprTest` เราได้ทำการทดสอบโดยการสร้าง `setUpGame` เพื่อจำลองโครงสร้างของเกม แล้วนำมา test โดยส่วนแรก เราได้ test การคำนวนบวก ลบ คูณ หาร mod ยกกำลัง หารสุบบิร์

โดย variable ที่เราใช้ในการ test จะมี 0, 1, -1, 1000, 1001, -1000, -1001

- `Plush()` test การบวก
- `SubT()` test การลบ
- `MutiP()` test การคูณ
- `Divi()` test การหาร
- `ModPer()` test การmod
- `Power()` test การยกกำลัง
- `DivisionByZero()` test การหารด้วย 0

- `CombinationOfOperations()` test การใช้หลาย Operations

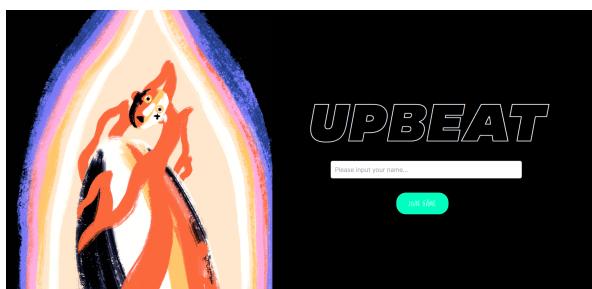
ส่วนที่สอง เราได้ทำการ `testConstructionPlan` โดยการจำลอง `setUpGame` และนำ Construction Plan มา test เช่น `t = t + 5` , `("d = deposit / 4")` , `dir = random` และดูค่า expected ว่าตรงกับไม่

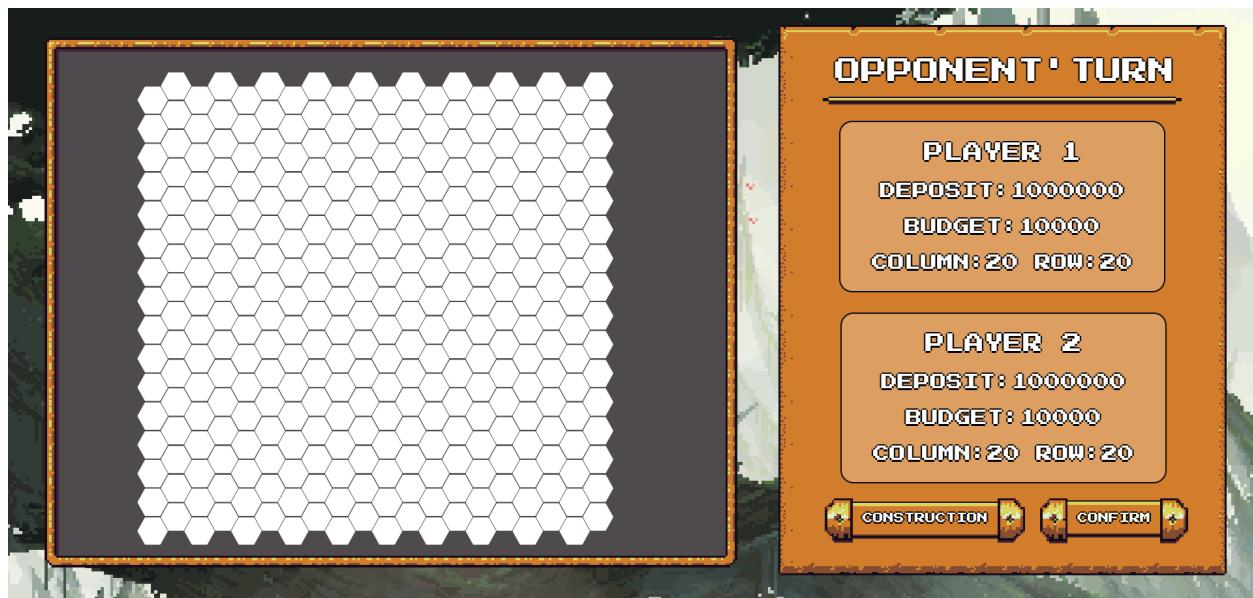
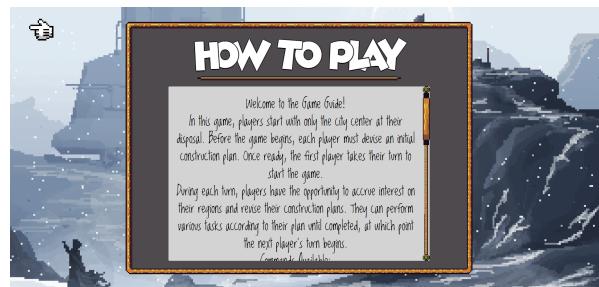
#### ▼ Construction Plan Parser Test

`ConstructionPlanParserTest` เราจะจำลองการเล่นโดยให้ Run `Construction Plan` ที่ได้จากหน้าเว็บที่ผู้เล่นส่งเข้ามา และลอง Parse และ Evaluate ดู output ว่ามันทำงานตามที่ต้องการหรือไม่ ส่วนแรกเป็นการ Test ที่ผู้เล่นจะมี action เช่น shoot, move, invest, relocate, deposit โดยทดสอบแยกๆ กันและอีกส่วนหนึ่งเป็นการ Test Statement เช่น if else , while , statement , block statement โดยมี Statement อยู่ด้านในเป็น action ที่เป็นเช่น shoot, move, invest, relocate, deposit ด้านใน รวมถึงเป็น block statement ที่มีหลาย action โดยดู output ที่ออกมาว่าถูกต้องตามที่ควรเป็นตาม Project specifications หรือป่าว

จากการทำ testing ทำให้เราได้เรียนรู้ว่าควรมี test cases อะไรบ้างที่จะทำให้ครอบคลุมงานของเรามากขึ้น โดยต้องอาศัยความอดทนในการแก้ไข test case ที่มีปัญหาไปทีละจุด เพื่อป้องกัน Bug ที่สามารถเกิดขึ้นได้ในอนาคต

## User Interface design





## Work Plan

สมชิก	หน้าที่
กิตติวัฒน์ ยะสารวรรณ	ทำงานในส่วนของ Backend

สมาชิก	หน้าที่
ไกเกอร์ แกนเนอร์	ทำงานในส่วนของ Frontend
ศุภกร สุวรรณภพ	ช่วยเหลือทำงานในส่วนของ Backend & Frontend โดยเน้นในส่วนของ Backend เป็นหลัก

#### ▼ Assessment of work that has occurred

การทำงาน Front-end และ Back-end เกือบเสร็จสมบูรณ์

- Front-end ออคแบบเซ็จสมบูรณ์ สวยงานตามที่ต้องการ
  - Back-end Parser, AST, GameState, Expr เซ็จสมบูรณ์ สามารถส่ง API ได้

ในตอนนี้เหลือเพียงแค่การเชื่อม Front-end และ Back-end โดยใช้ WebSocket ที่ยังมีปัญหาอยู่

Timeline	Deadline
ออกแบบ overview document, กำความเข้าใจหัวข้อที่กำหนด, แบ่งหน้าที่คร่าว ๆ	Wed 24 Jan
ออกแบบ overview document ของ game state และ user interface, กำ construction-plan	Wed 7 Feb
กำ Implementation of game state และ user interface, ออกแบบ overview document สำหรับ server-client implementation	Wed 21 Feb
กำ Server-client implementation, กำ Project report	Wed 6 Mar
กำ WebSocket และ API เชื่อมระบบ Front-end และ Back-end	Mon 25 Mar

จากการทำงานทำให้เรารู้ว่า เราควรศึกษาความรู้ในหลาย ๆ ส่วน เพื่อก่อหลังจากเราทำงานในส่วนของตัวเองเสร็จแล้ว เราสามารถไปช่วยหรือเช็คในส่วนอื่น ๆ ของสมาชิกในทีมได้ เพื่อให้งานมีความถูกต้องล่วงหากมีข้อบกพร่อง

## Known problems \*\*

- มีปัญหาเรื่องการเชื่อมระบบ Front-end และ Back-end ให้ได้ตามที่ต้องการ สามารถส่งข้อมูล API จาก Back-end ได้ แต่ไม่สามารถเชื่อมต่อผ่าน WebSocket ได้เสร็จสมบูรณ์

## Comments (optional)

- ຈະໄດ້ກ່ອຍາກໃຫ້ບອກຕ້ອນກ່ອນເຮັ່ມໂປຣເຈົກ:

- โปรเจคนี้ตอนอ่าน spec ครั้งแรกเข้าใจยากมาก อยากให้อาจารย์มีตัวอย่างคร่าว ๆ ว่าตอนเสร็จสมบูรณ์เป็นอย่างไร เล่นยังไง
  - สิ่งที่ไม่ชอบในโปรเจคนี้:
    - รู้สึกว่าโปรเจคนี้เป็นเกมที่ไม่ค่อยเป็นเกมเก่าໄหร້ ไม่รู้สึกว่าผู้เล่นจะได้รับความเพลิดเพลินอะไรจากการเล่น อยากให้เป็นโปรเจคที่น่าสนใจกว่านี้ เช่น เกม RPG
- 

## Line counts

C:\Users\Suhan\IdeaProjects\UPBEAT_Project>cloc-2.00.exe HEAD				
Language	files	blank	comment	code
JSON	5	0	0	5338
Java	42	327	92	2300
CSS	10	160	211	1115
JavaScript	10	52	11	886
Text	6	56	0	367
Bourne Shell	1	30	63	215
XML	10	0	0	191
DOS Batch	1	36	0	169
TypeScript	7	20	4	159
JSX	3	15	1	114
Maven	1	5	0	79
Markdown	2	13	0	24
Properties	2	0	1	3
SVG	2	0	0	2
SUM:	102	714	383	10962

## Commit summary report

```
PS D:\UPBEAT_Project> git shortlog -se
 42 Okinawa <fitthep@gmail.com>
 41 Suppakorn <110958686+Suwannapop@users.noreply.github.com>
 55 Suppakorn <suwannapop01@gmail.com>
 31 tigerlab <tiggers12@hotmail.com>
```