ASSIGNMENT 5
LEGENDS OF VALOR
DESIGN DOCUMENTATION

TEAM :
OWEN MARIANI (U74333523)
JIGAR KANAKHARA (U21762912)
ALI ALNASEEB (U87307609)

1. Purpose and Scope

This project implements two console-based, turn-based RPG games in Java:

- Heroes and Monsters
- Legends of Valor

The main goal was to reuse a shared object-oriented framework (board, game loop, pieces, figures, items, data loading, and UI utilities) while adding Legends of Valor–specific mechanics such as a lane-based board, movement legality rules, teleport/recall, terrain bonuses, and lane-based turn order.

This document focuses on design evaluation: why specific implementations were chosen, what alternatives existed, and whether the final choices achieved the intended results.

2. Architecture Overview (High-Level)

The codebase is split into three main layers:

A) Common (Reusable Framework and Systems)

- Core framework: Game, Board, Square, Player/Piece, Effect
- Shared gameplay systems: Figure, Party, Hero, Monster, Items/Inventory
- Data-driven loading: TextDataLoader + LoadableFromText
- Utilities: input handling + console UI helpers + validators

B) HeroesAndMonsters (Base Game Mode)

- HMBoard / HMSquare / HMSquareType
- HMGame + HMGameState
- Battle and market systems used by the base RPG loop

C) LegendsOfValor (Extension Game Mode)

- LVBoard / LVSquare / LVSquareType (8x8 lane board)
- LVGame (lane-based turns, movement/teleport/recall legality rules)
- LV menus: LVMainMenuOptions, LVMovementMenuOptions

Why this split worked:

- Maximizes code reuse (items, heroes, monsters, data, UI) across both game modes
- LV-only logic stays isolated and does not complicate Heroes & Monsters

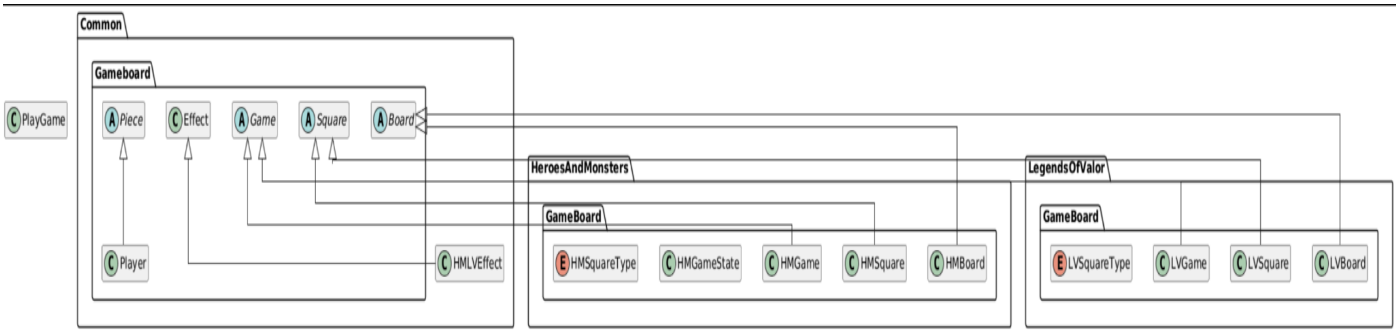
3. Key Design Decisions and Evaluation

- **Shared framework with game-specific extensions**:The project uses a reusable core in Common and builds HeroesAndMonsters and LegendsOfValor on top of it. This kept shared systems (figures, items, inventory, data loading, UI) consistent across both games while allowing Legends of Valor to introduce lane-based rules without disrupting the base game.
- **Generic board/square types for safety and reuse**: The Board<T extends Square> and Square<TType> design ensures each game mode uses the correct square type at compile time (e.g., LV cannot accidentally use HM squares). This improved maintainability and made it easier to extend the framework to new variants.
- **Centralized movement legality in Legends of Valor**: Movement, teleport, and recall legality checks are enforced in one place in the LV logic, ensuring consistent rule enforcement (walls/lane separation, occupancy rules, and "cannot move behind monsters" constraints). This reduced edge-case bugs and simplified special actions.
- **Lane-based turn order for Legends of Valor**:Turns are processed lane-by-lane (Hero then Monster per lane). This matches the tactical structure of the board, is easy for players to follow, and makes the implementation/debugging straightforward.
- **Data-driven configuration for heroes/monsters/items**:Heroes, monsters, spells, weapons, armor, and potions are loaded from text files via TextDataLoader. This allows new content to be added without changing Java code, at the cost of requiring consistent file formats and directory paths.

4. UML Diagrams

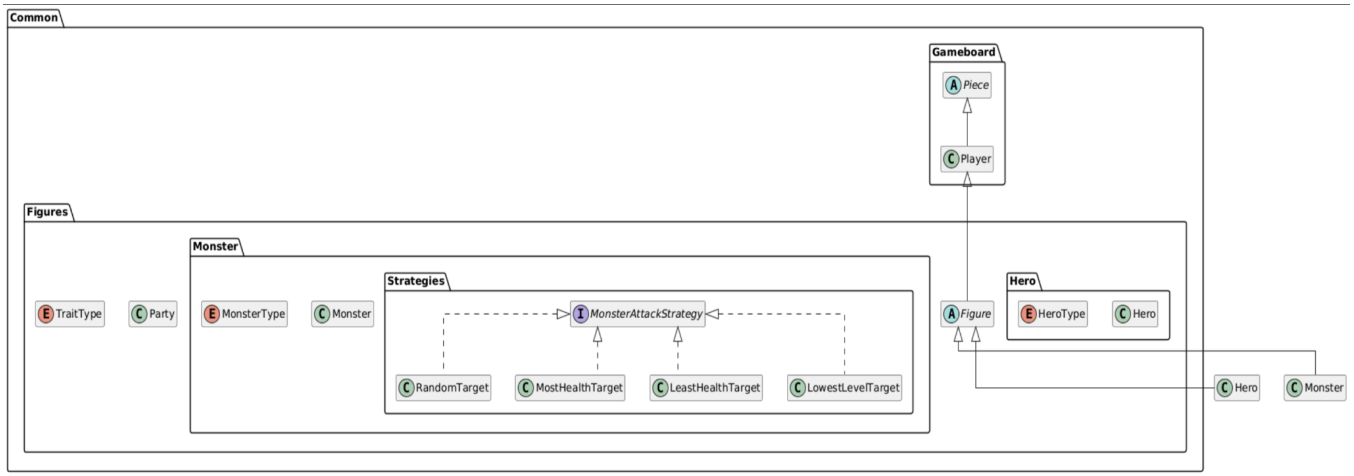
4.1 Diagram 1 — High-Level Framework + Game Modes

- The shared base framework in Common.Gameboard (Game, Board, Square, Piece, Player, Effect)
- The two game modes extending the framework:
    - HMGame, HMBoard, HMSquare, HMSquareType
    - LVGame, LVBoard, LVSquare, LVSquareType



4.2 Diagram 2 — Figures + Monster Strategy

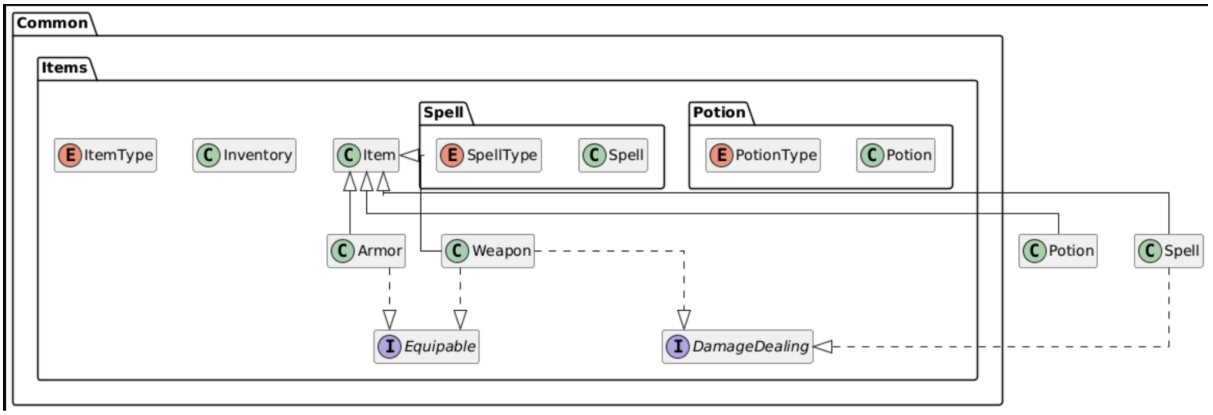
- Figure as the shared base for playable/AI-controlled entities
- Hero and Monster extending Figure
- Party representing collections of figures
- Monster targeting behavior implemented with the Strategy Pattern:
    - MonsterAttackStrategy interface
    - Strategies such as RandomTarget, MostHealthTarget, LeastHealthTarget, LowestLevelTarget



4.3 Diagram 3 — Items + Inventory

- Item as a base class with subclasses:
    - Weapon, Armor, Spell, Potion

- Interfaces to model shared item capabilities:
    - Equipable (items that can be equipped)
    - DamageDealing (items that deal damage, such as spells/weapons)
- Inventory as the owner/container of items and the basis for trading systems



5. What Went Well

- Clear separation of concerns:
    - Shared framework is reusable and mostly game-agnostic
    - HM and LV logic are kept in separate packages
- Extensibility:
    - New heroes/monsters/items can be added through data files
    - New monster strategies can be added without modifying existing strategy code
- Maintainability:
    - Centralized movement legality reduced duplicated logic and prevented inconsistent rules

6. What We Would Improve

- Reduce coupling between UI/input flow and core game logic to make testing easier
- Add minimal automated tests for movement legality (especially LV edge cases)
- Document state transitions more explicitly (e.g., when battles trigger, when markets are accessible)

7. Contributions

Owen Mariani

1. Source of foundational codebase that we extended
2. Architected new project class structure
3. Built Game logic
4. Refactored and integrated all previous game logic into new game
5. Implemented win/lose condition
6. Implemented obstacles and how to remove them
7. Implemented all user menus
8. Implemented non-battle attack system
9. Implemented Monster spawning
10. Implemented Hero respawing & dying
11. Implemented Hero HP & MP regeneration
12. Implemented Team-based lane-sorted turn taking mechanics
13. Implemented Hero Lane-Choosing Logic
14. Wrote File Headers

Jigar Kanakhara

1. Implemented the Legends of Valor Board structure
2. Created the wall and lane logic
3. Implemented the initial base hero and monster movements(W/A/S/D)
4. Implemented the teleportation features
5. Introduced nexus , B/C/K squares and random probability in their placements.
6. Created the read me file and design document for the assignment.

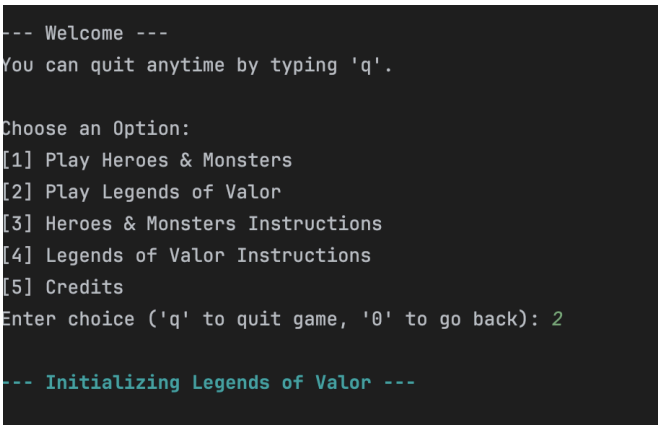
Ali AlNaseeb

1. Prepared the slides
2. Helped to Introduce nexus , B/C/K squares and random probability in their placements.
3. Source of foundational codebase that we extended
4. Architected new project class structure
5. Built Game logic
6. Refactored and integrated all previous game logic into new game

8. Example Output

8.1 Board

8.2 Game Instructions



8.3 Move Instructions

```
Choose an Option:
[1] Move
[2] Open Backpack
[3] See Statistics
[4] Pass
Enter choice ('q' to quit game, '0' to go back): 1
Choose an Option:
[1] Move Character
[2] Teleport
[3] Recall
Enter choice ('q' to quit game, '0' to go back):
```

8.4 Game Stats

```
---- Lane 3 Hero Turn ----
Hero: Skye_Soar (H3) at (6, 6)
Choose an Option:
[1] Move
[2] Open Backpack
[3] See Statistics
[4] Pass
Enter choice ('q' to quit game, '0' to go back): 3
Which group's statistics would you like to see?
Choose an Option:
[1] Monsters
[2] Heroes
Enter choice ('q' to quit game, '0' to go back): 1
Which statistics would you like to open?
Choose an Option:
[1] Casper
[2] Natsunomeryu
[3] Blinky
Enter choice ('q' to quit game, '0' to go back): 1
===== Monster STATUS =====
Name: Casper (Spirit)
Level: 1
HP: 100/100
Attack Strategy: Least Health
--- Attributes ---
Base Damage:  100
Base Defense: 100
Dodge Chance:   60
========================
```