Design Document: Legends of Valor

Project Overview

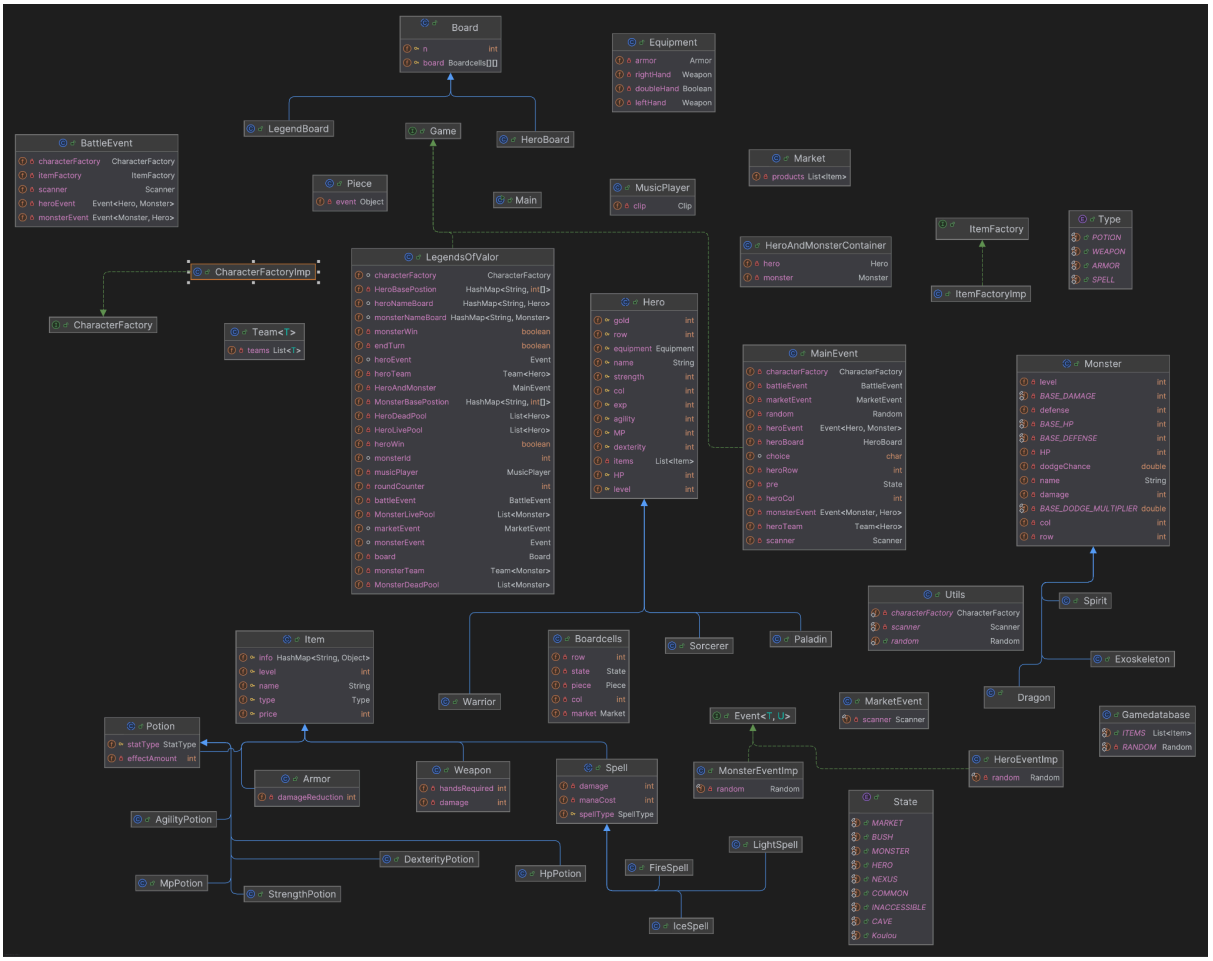
Legends of Valor is a turn-based role-playing game where heroes and monsters face off in a strategic battle environment. The game implements various design patterns to ensure scalability, flexibility, and reusability while providing an immersive gameplay experience. It features board-based navigation, dynamic character interactions, and rich item mechanics.

Authors

- Jiahao He (jiahhe@bu.edu)
- Bhavya Surana (surana@bu.edu)

UML Diagram

The UML diagram provides a detailed representation of the project's architecture, illustrating the relationships between core components like Heroes, Monsters, Board, Items, and events.



System Design

1. Core Features

- Turn-Based Gameplay: Heroes and monsters take turns performing actions like attacking, moving, or using items.
- Dynamic Board Mechanics: The 8x8 game board supports different cell states (e.g., Nexus, Bush, Cave) influencing gameplay mechanics.
- Character Classes: Heroes and monsters have unique abilities defined by their respective classes (e.g., Warrior, Paladin, Dragon).
- Market System: Heroes can purchase or sell items in Nexus markets.
- Event-Driven Design: Encounters like battles, market transactions, and hero actions are handled via specific event implementations.

2. Class Design

Controller Package

1. BattleEvent
    - Manages hero and monster battles.
    - Rewards heroes with gold and experience after defeating monsters.
    - Implements methods like startBattle, heroTurn, useSpell, useWeapon, and rewardHeroes.
2. MarketEvent
    - Handles buying and selling of items.
    - Provides the action method to initiate market interactions.
3. LegendsOfValor
    - Core game logic controller.
    - Handles board setup, hero/monster interactions, and the game loop.
    - Methods include initializeHeroPositions, moveAndCheckForEncounter, and respawnDeadHeroes.
4. MusicPlayer
    - Adds background music to enhance the user experience.
    - Provides methods to play and stop music.
5. MainEvent
    - Manages high-level interactions between heroes and monsters.
    - Coordinates battle and market events.
6. Game Interface
    - Abstract interface for defining game operations like starting, moving, and displaying instructions.

Entity Package

1. Heroes and Monsters
    - Hero Classes: Warrior, Paladin, Sorcerer.
    - Each class specializes in attributes like strength, agility, or dexterity.
    - Monster Classes: Dragon, Exoskeleton, Spirit.
    - Each class provides unique abilities like doubling damage or dodge chances.
2. Board
    - LegendBoard: Implements a fixed 8x8 grid with diverse cell states (Nexus, Bush, Cave, Koulou).
    - Boardcells: Represents individual cells on the board.
    - State Enum: Encapsulates possible cell states, influencing hero and monster actions.

3. Items
    - Weapons, Armor, Potions, Spells: Each item type affects specific attributes (e.g., strength, HP, mana).
    - Items are managed via the Gamedatabase and are accessible in the market.
4. Team
    - Groups heroes or monsters into a cohesive team for easier management.

Repository Package

1. CharacterFactory
    - Interface for creating heroes and monsters.
    - Encapsulates object creation logic to promote scalability.
2. ItemFactory
    - Interface for creating game items.
    - Ensures consistency in item instantiation across different types.
3. Event Interface
    - Defines actions like attack, castSpell, and usePotion.
    - Implemented by HeroEventImp and MonsterEventImp for specific behaviors.

3. Design Patterns

1. Factory Pattern
    - Used in CharacterFactory and ItemFactory to create heroes, monsters, and items dynamically.
    - Promotes extensibility by isolating instantiation logic.
2. Strategy Pattern
    - Achieved through the Event interface and its implementations (HeroEventImp, MonsterEventImp).
    - Enables interchangeable behavior for heroes and monsters during encounters.
3. State Pattern
    - Implemented in LegendBoard through the State enum.
    - Alters gameplay based on cell states (e.g., Bush increases dexterity, Cave increases agility).

4. Board Design

The 8x8 board is divided into:

- Nexus: Hero and monster base rows; heroes can access markets in Nexus cells.
- Lanes: Pathways for hero and monster movement, separated by inaccessible walls.
- Special Zones: Cells like Bush, Cave, and Koulou grant attribute bonuses.

5. Gameplay Flow

1. Initialization
    - Heroes and monsters are created using the factory pattern.
    - The board is initialized with Nexus, lanes, and special zones.
2. Turn-Based Actions
    - Heroes and monsters take turns performing actions (e.g., attacking, moving).
    - Market transactions are available when heroes are in Nexus cells.
3. Encounters
    - Heroes and monsters engage in battles when adjacent.
    - Victory conditions are checked after each round.

4. Round Management
    - Dead heroes respawn in Nexus.
    - New monsters are spawned periodically.

6. **Extensibility**

1. Adding New Character Classes
    - Extend the Hero or Monster base class.
    - Update CharacterFactoryImp to support the new class.
2. Expanding Items
    - Add new item types to the Gamedatabase.
    - Create corresponding classes (e.g., AdvancedWeapon, HealingPotion).
3. Enhancing Board Mechanics
    - Introduce new cell states (e.g., traps, teleporters) in the State enum.
    - Update LegendBoard to include these states in the initialization logic.

Compiling and Running

1. Navigate to the home directory.
2. Run:
   javac *.java
   java Main

Conclusion

Legends of Valor is a robust, extensible project showcasing advanced object-oriented design principles. By leveraging design patterns like Factory, Strategy, and State, the system is both flexible and maintainable, capable of accommodating future gameplay enhancements.