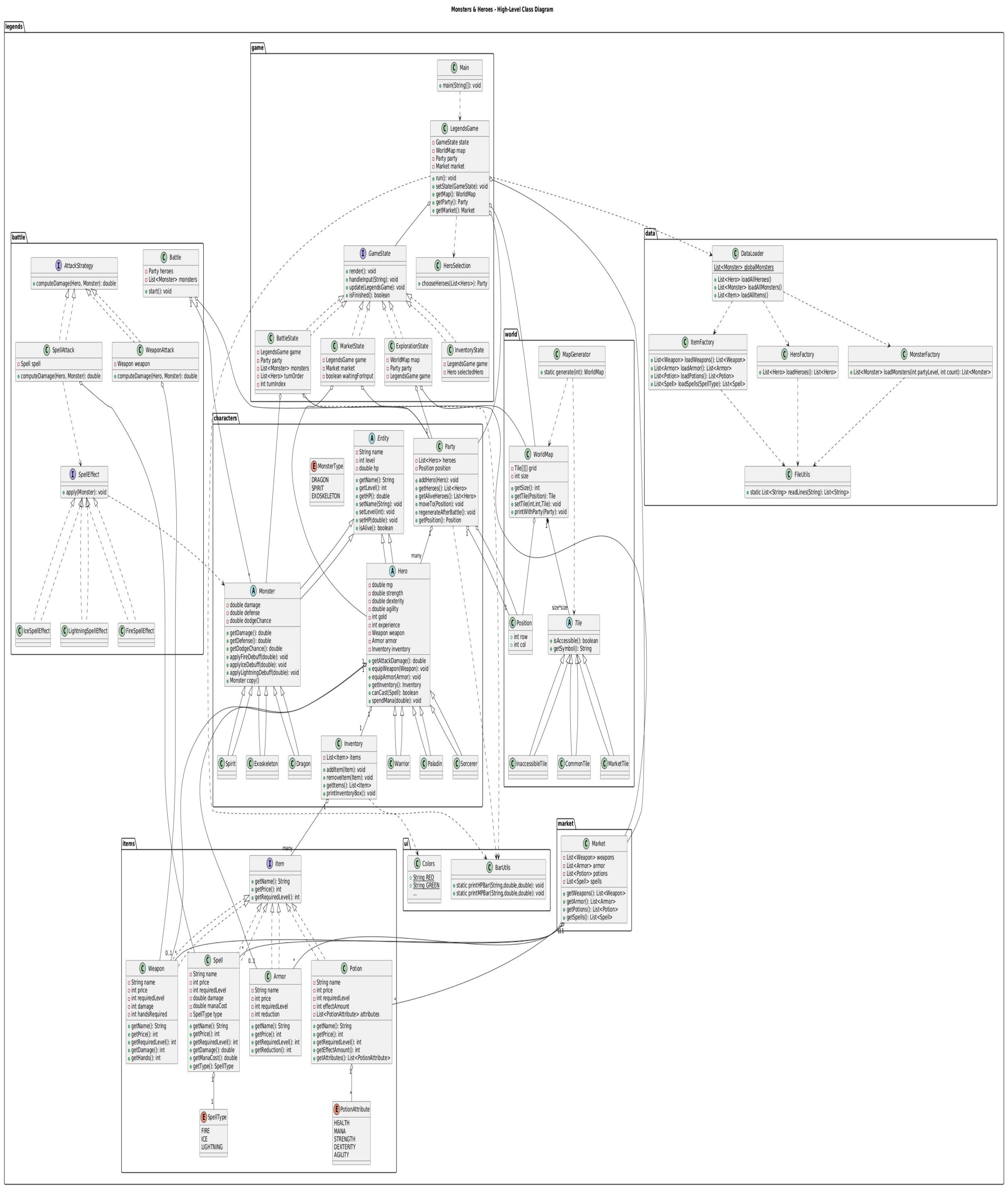
DesignDocument



Monsters & Heroes – Design Document

1. Introduction

Monsters & Heroes is a turn-based, text-based RPG implemented in Java. The player controls a party of heroes that travel across a world map, fight monsters, access markets, and manage inventory. The design emphasizes modularity, extensibility, and clear separation of responsibilities using object-oriented principles and patterns.

2. High-Level Architecture

The system is organized into several packages:

- legends.game – Game engine, states, main loop

- legends.characters – Heroes, monsters, inventories

- legends.items – Weapons, armor, spells, potions

- legends.world – Tiles, map generation, positions

- legends.market – Market structure

- legends.battle – Combat strategies and battle flow

- legends.data – Factories and data loaders

- legends.ui – Console color and UI utilities

3. Package Descriptions

Each package contributes a specific piece of functionality, ensuring isolation and easier maintenance. The game uses a state machine to transition between exploration, battle, and market modes.

3.1 legends.game

Contains the core game loop and GameState design pattern. LegendsGame drives the application, while ExplorationState, BattleState, MarketState, and InventoryState handle different gameplay modes.

3.2 legends.characters

Defines Entity, Hero, Monster, and subclasses. Party manages team logic and positioning. Inventory provides item storage and operations.

3.3 legends.items

Includes Item interface and concrete classes: Weapon, Armor, Spell, Potion. Supports item equipping, spell casting, and potion stat boosts.

3.4 legends.world

Implements Tile hierarchy, WorldMap, and MapGenerator. Manages map creation, tile access, and movement restrictions.

3.5 legends.market

Defines Market, containing purchasable weapons, armor, potions, and spells. Interacted with through MarketState.

3.6 legends.battle

Implements AttackStrategy, WeaponAttack, SpellAttack, and SpellEffect classes. Handles damage calculation and special spell effects.

3.7 legends.data

Factory and loader classes parse hero/monster/item files and construct objects. Provides a clean separation between data and logic.

3.8 legends.ui

Provides UI utilities such as ANSI color codes and health/mana bar generators.

4. Key Design Patterns

State Pattern for handling gameplay modes.

Strategy Pattern for attacks and spell effects.

Factory Pattern for object creation from external data.

Inheritance and polymorphism for defining character types and items.

5. Game Flow Overview

Startup loads data and map, then enters ExplorationState.

Player moves, explores, and may enter battles or markets.

BattleState handles turn-based combat.

MarketState and InventoryState handle trading and equipment.

Flow continues until heroes die or player exits.

6. Extensibility

New heroes, monsters, tiles, spells, or states can be added easily. The architecture supports long-term growth and additional mechanics.