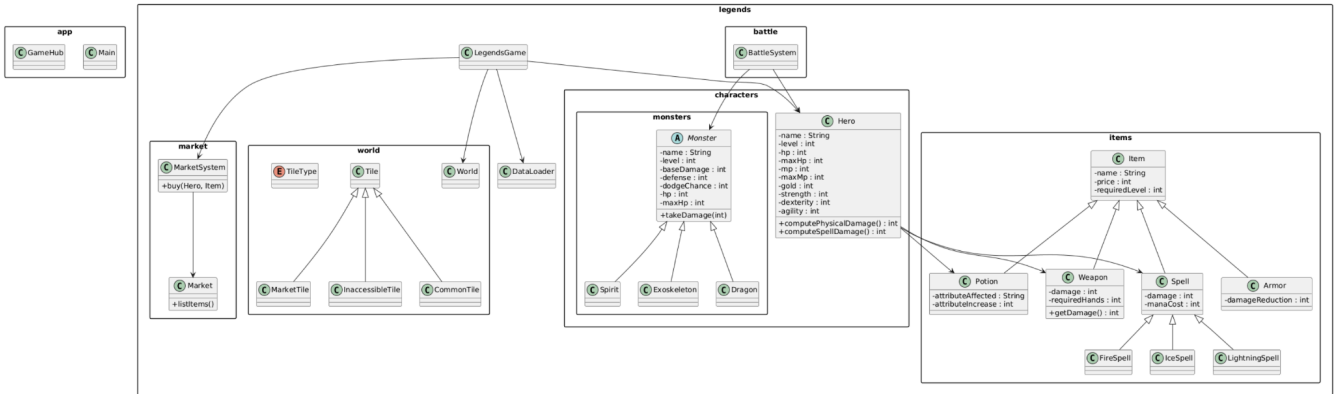
ASSIGNMENT 4
DESIGN DOCUMENTATION
NAME : JIGAR KANAKHARA
ID : U21762912

UML diagram



The UML diagram (see page above) shows all major modules of the **Legends: Monsters and Heroes** project, organized into packages:

- app
- legends.battle
- legends.characters
- legends.characters.monsters
- legends.items
- legends.market
- legends.world

- legends.txt_data

The diagram illustrates class relationships, inheritance, dependencies, and composition across all game components.

Class Descriptions

App Module

Main

Responsible for launching the entire system.

- main(String[]) — delegates execution to GameHub.

GameHub

Displays the main menu and launches sub-games.

- run() — handles user selection and executes chosen game mode.

Legends Module

Battle Submodule

BattleSystem

Handles turn-based combat between heroes and monsters.
Attributes:

- List<Hero> party
- Monster monster
- Scanner scanner

Key Methods:

- runBattle() — complete combat loop
- heroTurn(Hero) — hero actions: attack, spell, potion, guard
- monsterTurn() — balanced monster attack logic
- computeMonsterDamage() — fixed ~40 damage with jitter
- chooseWeapon(Hero) — per-turn weapon switching
- printStatus() — displays HP/MP states

Characters Submodule

Hero

Represents a playable hero.
Attributes:

- name, level, hp, mp, gold
- strength, agility, dexterity
- weapons, potions, spells

Operations:

- computePhysicalDamage()
- computeSpellDamage(Spell)
- takeDamage(int)
- heal(int)
- equipWeapon(Weapon)
- usePotion(int)
- isAlive()

Monsters Submodule

Monster (Abstract)

Base class for all monster types.
Attributes:

- name
- level
- baseDamage
- defense
- dodgeChance
- hp = 300, maxHp = 300

Methods:

- takeDamage(int)
- getDodgeChance()
- scaleToLevel(int)

Dragon / Spirit / Exoskeleton

Concrete subclasses loading monster attributes from text files.

Items Submodule

Item (Abstract)

Base class for all inventory items.
Attributes:

- name

- price
- requiredLevel

Weapon

Attributes:

- damage, requiredHands
  Methods:
- getDamage()

Armor

Attributes:

- damageReduction

Potion

Attributes:

- attributeAffected
- attributeIncrease

Spell

Attributes:

- damage, manaCost
  Subclasses: FireSpell, IceSpell, LightningSpell.

Market Submodule

Market

Stores items available for purchase.

- listItems() — display catalog

MarketSystem

Purchasing and inventory management.

- buy(Hero, Item) — validates gold and level requirements

World Submodule

World

8×8 tile-based grid representing the map.
Attributes:

- Tile[][] map
- heroRow, heroCol

Methods:

- move(char) — WASD movement
- render() — draw ASCII map
- getCurrentTile() — determine if tile triggers battle or market

Tile (Abstract)

Represents a map square.
Subtypes:

- CommonTile — walkable, may start battle
- MarketTile — access to shop
- InaccessibleTile — blocked square

TileType (enum)

Labels tile categories for world generation.

txt_data Submodule

DataLoader

Loads hero, monster, spell, weapon, armor, and potion definitions from text files.
Provides parsed objects to LegendsGame.

Game Controller

LegendsGame

Central orchestrator of the Legends RPG.

Attributes:

- World world
- List<Hero> party
- int battleWins
- Scanner scanner

Methods:

- start() — main game loop
- createParty() — hero selection and initialization
- handleMove(char) — movement and event logic
- triggerBattle() — constructs BattleSystem
- checkVictoryCondition() — 3 wins required to finish the game

Class Relationships

Inheritance

- Monster → Dragon, Spirit, Exoskeleton
- Item → Weapon, Armor, Potion, Spell
- Spell → FireSpell, IceSpell, LightningSpell
- Tile → CommonTile, MarketTile, InaccessibleTile

Associations

- BattleSystem uses Hero and Monster
- LegendsGame uses World, DataLoader, MarketSystem, BattleSystem
- MarketSystem uses Hero inventories

Composition

- World contains the entire grid of Tiles
- Hero contains lists of weapons, spells, and potions

Dependency

- DataLoader depends on text-based input files
- World depends on valid tile types

Encapsulation

- All fields in gameplay classes are private or protected
- Public getters/setters restrict direct modification
- Internal combat rules remain encapsulated inside BattleSystem
- Market logic is independent and cleanly separated
- World navigation is isolated in World class

Scalability

- Adding heroes, monsters, or items requires only text file updates
- New tile types can be added via subclassing Tile
- Map size can be changed by modifying one constant
- Multiple heroes are supported without structural changes
- Supports future expansions: spells, equipment, or new systems

Extendibility

- New spells can override damage formulas
- New item types inherit from Item
- New battle mechanics can be introduced inside BattleSystem
- Additional monster archetypes can be created easily
- GUI can be built on top of existing backend logic

Testing Summary

- Verified map navigation across all tile types
- Verified market purchases and level/gold requirements
- Verified correct loading of items and monsters from text files
- Tested complete combat loop including:
    - attack damage
    - dodge mechanics
    - spell damage
    - potion effects

    - weapon switching

- Confirmed monster damage remains 35–45 consistently
- Verified win condition triggers after 3 battles