# CS 611 Legend\_Of\_Valor

## UML



## Structure

### Folder src/Characters

The Folder Characters contains classes to represent monsters and humans.

**Classes:**

1. Character:

The parent class of Monster and human, which contains some common attributes between human and monster.

2. Human:

The parent class of all three types of heroes. This class deals with items and levels and.

3. Paladin or Sorcerer or Warriors:

The classes for different types of heroes. Each of them has their own levelUp() method.

4.Monster:

The class of Monster. The Monsters’ HP depends on their levels.

5.**CharacterMaps:**

The class of all characters loaded from .txt file. We put all heroes and monsters in their own Map. As a result, we have three maps for different monsters and three for heroes. All data from .txt files will be checked and distinguished automatically.

### Folder src/Colors

This folder contains an interface who handles colors.

**Interface:**

1. ColorsCodes:

This interface contains a lot of color codes.

### Folder src/data:

This folder contains all .txt data from homework instruction.

### Folder src/Items:

This folder contains all Items, like weapon, spell, armory and potion. Every item has their own type.

Classes:

1. Items:

The parent class of all items. All items have name, price, level and type.

2. Armory:

The class of Armory.

3. Spell

The parent class of all three types of spells.

4. fire/ice/lightSpell:

The class of different spells and they differ in attribute “spellType”.

5. Weapon:

The class of weapon

### Folder src/places:

This folder contains all places and each place is a piece.

classes:

1. common:

The class of a base place, who has a rate to give heroes group a fight.

2. herosGroup:

The class of herosGroup piece, who contains a list of heroes in a game.

3. market:

The class of market piece.

4. wall:

The class of wall piece.

5. aMonster:

The class of monster piece.

6. heroPossiblePosition

Used in teleport to temporarily display the spaces a hero can teleport to,

each marked with a unique index as its sign.

### Folder src/tool:

This class contains all the tools that can be used in anywhere in the project.

Interface:

1. formats:

The interface contains many string formats. We use them when we need to print some information with format.

**2. toolClass:**

The class of all tools and every tool is static so we don’t need to new an instance to use these tools.

The tools can be divided into two groups: random tools and loading tools.

The random tools deal with random logic, like getting a random number or throwing a dice. Also, **we can get a random value from any map or list.**

The loading tools deal with loading .txt files, like getting data line by line from a .txt file.

### Folder src/workLogics:

This folder contains some important game logic like shopping or battle.

Classes:

1. Battle:

The class of battle, who handles the whole work flow of a battle. In each round, heroes can attack, use spell, pick a weapon or armory, print information, use potion or keep waiting.

2. Formulas:

The class of most formulas. We calculate the damage, dodge chance, Exp and Gold after battle …… in this class. Most of formulas are able to accept different multiply powering to fit in different logics.

3. Market:

The class of shopping, not the class of Market piece. This class handles the whole workflow of shopping with one hero. The goods in Market are random and the shopping results will be kept during the game flow.

**Class market is a market factory** because every market on the map needs to be individual. In this way, we ensure that they are different market instances.

### Folder src:

The root folder contains all game files.

Classes:

1. board:

The class of board.

2. Cell:

The class of sell.

3. Main:

The class that contains the main function.

4. workFlow:

The class of game flow.

## Patterns

1. We use **Factory pattern** to handle the heroes loading process. If we want to add a new type of hero, we only need to add the .txt file and a new class for the hero.  
2. We use **Facade pattern** in CharacterMaps and itemLists. Both of them provide a unified interface to get the data from .txt files.  
3. We use **Singleton pattern** in AudioPlayer. We only need one instance of AudioPlayer to play music.

## design evaluation

1. We created some tools to pick random items from a map or a list. Because we load and put all items/heroes/monsters into maps or lists. In this way, we are able to get a random item at any time.

2. We create CharaterMaps and itemLists to handle all characters and items, which is a façade pattern. In this way, we can manage multiple interfaces more efficiently without disorganized.

3. All cells will not contain any sign of heroes or monsters. Pieces of heroes and monsters does “move” from one cell to another cell. We created stacks for every cell to contain pieces. In this way, we can put multiple pieces in one cell.

4. We picked parts of battle logics from Heroes and Monsters. To be specific, we split the original turn-based combat logic into single attacks. So, both monsters and heroes can only attack once in their turn.