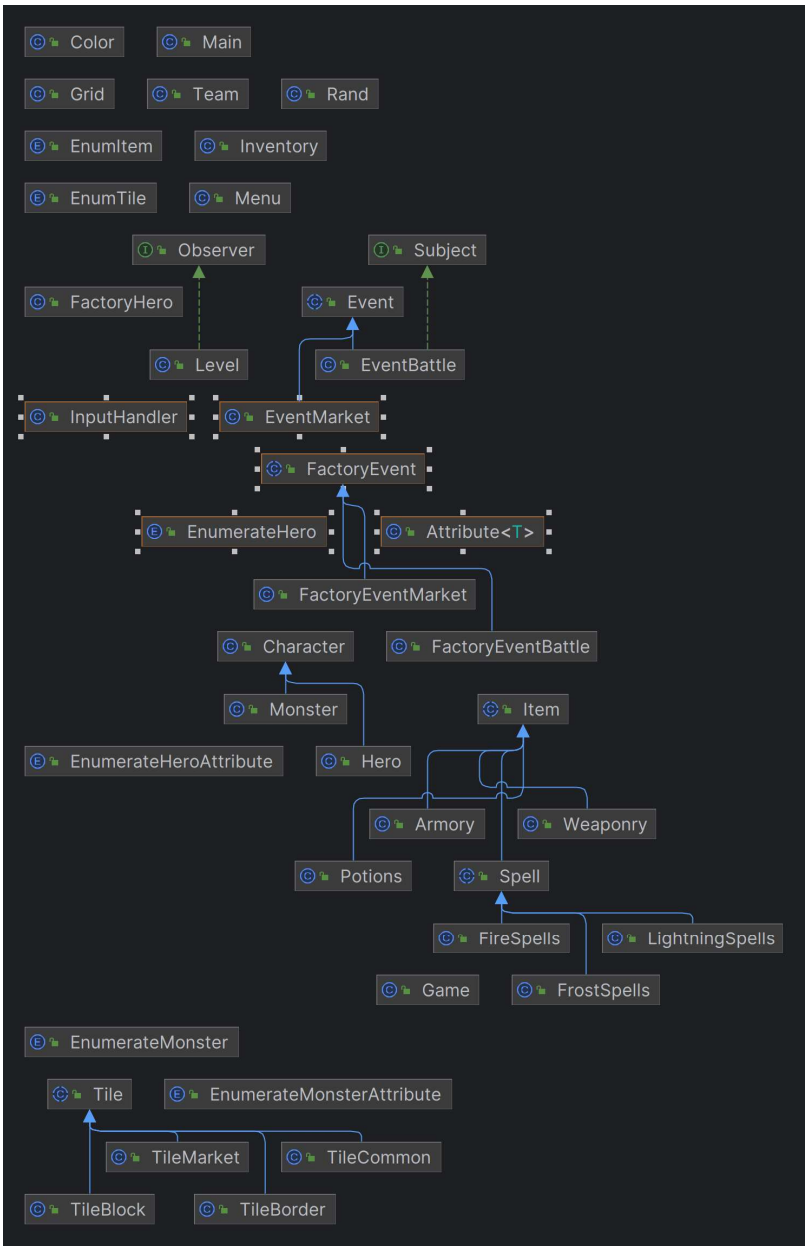
# Design documentation

## Overall

This program utilizes design patterns like Observer, Singleton, Factory to implements a monster and hero game.

## Architecture




## Class

/character/Attribute.java: Class managing hero or monster attributes

/character/FactoryHero.java: Class creating hero

/character/Hero.java: Class of a hero

/character/Monster.java: Class of a monster

/character/Inventory.java: Class of a hero's inventory

/character/Level.java: Class managing a hero's level. Observer Pattern.

/character/Team.java: Class of a team that contains multiple Heroes.

/data/*.txt: Raw data of characters / Items

/event/Event.java: Class of an event. An event is bind to a tile in grid

/event/EventBattle.java: Class of a battle

/event/EventMarket.java: Class of a market

/event/FactoryEventBattle.java: Class to create an EventBattle. Factory Pattern

/event/EventBattle.java: Class to create an EventBattle. Factory Pattern

/grid/Grid.java: Class storing the game board information

/grid/Tile.java: Class storing a tile. Base class.

/grid/TileBlock.java: Class storing a block tile. Subclass.

/grid/TileBorder.java: Class storing a border tile. Subclass.

/grid/TileCommon.java: Class storing a common tile. Subclass.

/grid/TileMarket.java: Class storing a market tile. Subclass.

/item/Item: Class of an item. An item is stored in either FactoryEventMarket or a hero's inventory.

/item/Item*: The specific item.

/utility/Color: Class that handles printing with color

/utility/InputHandler: Class that collect user input. Singleton Pattern/

/utility/Observer: Interface that a observer should implement

/utility/Subject: Interface that the class being observed should inplement

/utility/Rand: Class that handle random numbers.

/Enum*: enum class

/Game: Class that handles the game cycle

/Menu: Class that maintains a menu layer in the game cycle

/Main: The main entrance