CECS 277 – Project 2

Dungeon Master Part 2

Modify the code you wrote for Project 1 (fix all errors reported from gradesheet). Add in the following new features to your game:

1. Gold – the hero now has money. The hero can now buy and sell items at the store.
    a. Additional Hero instance variable: int gold
    b. Additional Hero methods: int getGold(), void collectGold(int g), and void spendGold(int g).
2. Store – the user has the option of visiting the store when the hero visits the start location on the map. The store sells Health Potions and Keys, and will buy any items from the hero's inventory and give the hero that item's value in gold.
    a. Add a method: void store(Hero h) to the main class.
    b. Modify a method: void dropItem(int i) should now return an Item.
3. Items – there are new features for items:
    a. Additional Item instance variable: int value – the amount of gold the item is worth at the store.
    b. Additional Item instance variable: char type – the type of item it is.
        i. p – potion – drink to restore hero's hp (consumed).
        ii. k – key – used to unlock door and move to next level (consumed).
        iii. m – money – can only be sold for gold
        iv. a – armor – can be used to deflect an attack (consumed).
    c. Additional Item methods: int getValue(), char getType().
    d. Additional ItemGenerator methods: getPotion() and getKey().
4. Armor – heroes can now wear armor to help defend against enemies.
    a. Items with type 'a' are armor items, and can be used to block an enemy's attack if one is in the hero's inventory (ex. enemy attacks for 4 damage, the entire 4 damage is blocked), after which the item is consumed and is removed from inventory.
    b. Additional Hero method: int hasArmorItem() – returns index of first armor item in inventory, or -1 if not found.
5. Key – The finish is now locked and requires a key to move to the next level. If the hero has a key in inventory, then they may progress to the next map.
    a. Additional Hero method: boolean hasKey() and void useKey()

Incorporate the following design patterns to better the design of your program:

1. Singletons:
    a. Item Generator, Enemy Generator, and Map
2. Prototype
    a. Item – call the clone() function in ItemGenerators generateItem() function.
3. Decorator
    a. Enemy - The EnemyGenerator will no longer read from the file, instead create the following base classes: Troll, Froglok, Orc, Goblin. Then create the decorators Warrior and Warlock. Attach the decorator titles to the name of the enemy (ie. Troll Warrior or Troll Warlock). Warlock should implement the Magical interface and its attack method will randomly call one of these. Warrior should increase its maxHp by 2 when decorated, and Warlock should increase its maxHp by 1 for when decorated.
4. Factory
    a. EnemyGenerator – no longer reads from the file. Instead randomly generate one of the base enemies at the level of the hero. If its level is greater than 1 randomly choose Warlock or Warrior. Then for each level greater than one, repeatedly decorate it with that type (ie. level 3 Troll might be decorated with Warrior twice), this will cause it to gain additional maxHp and do an additional attack for each level.