CSE 102L Computer Programming Laboratory – Homework05

Due Date: 23:59 Sunday April 14th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *HW05_yourStudentNumber.java*. Submissions made after the deadline will not be accepted, be sure to submit your work before the due date and make sure to click turn in button. Your code will be automatically controlled, so be sure to have the same class, method, variable names as described here. Failure to do so may result in you receiving 0 from this exercise. **All classes should be written to a single Java file. In a single java file there can only be single public class. Do NOT use Inner Classes. Be careful naming your file. If your editor inserts the file into a package, remove that line from the file but do NOT delete the import statements.**

Write set of classes according to the following specifications. Declare all attributes as **private** if not requested otherwise and use camelCase formatting for attributes.

Interfaces

1. Damageable
- Methods:
    - takeDamage(damage: int): None – decreases *health* by damage
    - takeHealing(healing: int): None – increases *health* by healing
    - isAlive(): Boolean – returns true if health > 0
2. Caster
- Methods:
    - castSpell(**target**: Damageable): None – uses takeHealing method of **target**. Sends *intelligence* + use method of *spell* to takeHealing method of **target**
    - learnSpell(spell: Spell): None – Basically a setter method
3. Combat – child of Damageable
- Methods:
    - attack(**target**: Damageable): None – uses takeDamage method of **target**. If the class that implements this is:
        - NonPlayableCharacter: sends sum of attributes to takeDamage method of **target**
        - PlayableCharacter: sends *strength* + use method of *weapon* to takeDamage method of **target**
    - lootWeapon(weapon: Weapon): None – Basically a setter method
4. Useable
- Methods:
    - use(): int

Classes

1. Spell – implements Useable

- Attributes:
    - minHeal: int
    - maxHeal: int
    - name: String
- Methods:
    - Constructor that takes *name, minHeal,* and *maxHeal*
    - Acessor and Mutator for *name* attribute
    - heal(): int – **private** method that returns a random number between *minHeal* and *maxHeal*
    - use(): int – just returns the result of **heal()** method.

2. Weapon – implements Useable

- Attributes:
    - minDamage: int
    - maxDamage: int
    - name: String
- Methods
    - Constructor that takes *name, minDamage,* and *maxDamage*
    - Accessor and Mutator for *name* attribute
    - attack(): int – **private** method that returns a random number between *minDamage* and *maxDamage*
    - use(): int – just returns the result of **attack()** method.

3. Attributes

- Attributes:
    - strength: int
    - intelligence: int
- Methods
    - No-Arg Constructor that sets *strength* and *intelligence* to 3
    - Constructor that takes *strength* and *intelligence*
    - increaseStrength(amount: int): None – increases the *strength* by amount
    - increaseIntelligence(amount: int): None – increases the *intelligence* by amount
    - Getters for attributes
    - toString(): String – returns a string in format:
        - “Attributes [Strength= ” + *strength* + “, intelligence= “ + intelligence + “]”

4. Character – an Abstract class that implements Comparable and Damageable, compares characters using *level* attribute.

- Attributes:
    - name: String
    - level: int – **protected**
    - attributes: Attributes – **protected**
    - health: int - **protected**
- Methods:
    - Constructor that takes *name,* and *attributes*
    - Getters for *name* and *level*
    - levelUp(): None – an **abstract** method
    - toString(): String – returns a String in the format:
        - ClassName + " LvL: " + *level* + " – " + attributes

5. PlayableCharacter – an Abstract child class of **Character**

- Attributes:
    - inParty: boolean
    - party: Party
- Methods:
    - Constructor that takes *name* and *attributes*
    - isInParty(): Boolean
    - levelUp: None – Increases level by 1
    - joinParty(party: Party): None – if in a party throws **AlreadyInPartyException**. Otherwise, **tries** to add **this** to *party.* If successful sets the *inParty* to true and sets *party* to party. Otherwise, catch **PartyLimitReachedException** and displays the error message
    - quitParty(): None – if is in a party, **tries** to remove **this** from party. If successful sets *inParty* to false and *party* to null. Catches **CharacterIsNotInParty** and displays the error message. If not in a party throws **CharacterIsNotInPartyException**.

6. NonPlayableCharacter – an Abstract child class of **Character**

- Methods:
    - Constructor that takes *name* and *attributes*

7. Merchant – child of **NonPlayableCharacter**

- Attributes:
    - Inventory: Collection of Useables
- Methods:
    - Constructor that takes *name* and creates an Attributes object with 0 strength and intelligence

- levelUp(): None – Empty
- display(): None – Displays all the items, each in a newline
- buy(itemNumber: int) Useable – returns the item at the given index, catches **IndexOutOfBoundsException** and throws **ItemNotFoundException**
- sell(item: Useable): void – adds the given item to inventory

8. Skeleton – child of **NonPlayableCharacter** that implements Combat

- Methods:
  - lootWeapon(weapon: Weapon): None – Empty
  - levelUp(): None – overrides super's method to increase *level, strength and intelligence by 1*
  - takeHealing(healing: int): instead of increasing health, healing received decreases health (just use takeDamage of super)
  - Other methods are as described above

9. Warrior – child of **PlayableCharacter** that implements Combat

- Attributes:
  - weapon: Useable
- Methods:
  - Constructor that only take *name* and **sets *health*** to 35. Creates an attribute instance with 4 *strength* and 2 *intelligence* and sends it to super's constructor
  - levelUp(): None – overrides and reuses super's method to increase *strength* by 2 and *intelligence* by 1.
  - Other methods are as described above

10. Cleric – child of **PlayableCharacter** that implements Caster

- Attributes:
  - spell: Useable
- Methods:
  - Constructor that only take *name* and **sets *health*** to 25. Creates an attribute instance with 2 *strength* and 4 *intelligence* and sends it to super's constructor.
  - levelUp(): None – overrides and reuses super's method to increase *strength* by 1 and *intelligence* by 2.
  - Other methods are as described above

11. Paladin – child of **PlayableCharacter** that implements Combat **and** Caster

- Attributes:
  - weapon: Useable

    - spell: Useable
- Methods:
    - Constructor that only take *name* and **sets *health*** to 30. Creates an attribute instance with No-arg Attributes constructor and sends it to super's constructor.
    - levelUp(): None – overrides and reuses super's method to increase *strength* by 2 and *intelligence* by 1 when level is even. Vice versa when odd.
    - Other methods are as described above

12. Party

- Attributes
    - partyLimit = 8: int – constant
    - fighters: a collection of instances that implement Combat
    - healers: a collection of instances that implements Caster
    - mixedCount: int – number of paladin in party
- Methods:
- addCharacter(character: PlayableCharacter): None – can throw **PartyLimitReachedException**. Adds the character to suitable collection
- removeCharacter(character: PlayableCharacter): None – can throw **CharacterIsNotInPartyException**. Removes the character from suitable collection
- partyLevelUp(): None – level ups the entire party once. (be careful to not level up paladins twice)
- toString(): String – returns a string in the format:
    - Displays each character in party (once!) in ascending order in a newline according to their *levels*

Exceptions

1. PartyLimitReachedException – child of Exception
2. AlreadyInPartyException – child of Exception
3. CharacterIsNotInPartyException – child of Exception
4. ItemNotFoundException – child of Exception

Exceptions constructor takes a single String parameter and sends it to super's constructor.