Project 3: The Polymorphic Pub

Due Date: June 20th, 2025 (11:59 PM)

Project Guidelines:

If there are questions about how to format your project, please refer to the "Project Guidelines" PDF attached to the message concerning "Project 1". If there are any other questions, feel free to email either one of us at KAZI.MANSHA45@myhunter.cuny.edu or meheraan.khan66@myhunter.cuny.edu.

Project Task:

Rumors have begun to swirl through the floating isles — whispers of unrest among the creatures. Though you've organized them, they remain isolated, misunderstood, and a little bored. Clancy Pembroke has now called upon you again. His vision? A **Tavern** — a magical meeting ground where all creatures, from fiery Firecats to moaning Banshees, and streetwise Cat Thugs, can gather to unwind, share stories, argue, or even plot their next great magical move.

For a more detailed description of what is being done:

- You will be utilizing the code that you wrote for Project 2. If you did not do Project 2 completely I will be posting a link to a GitHub containing the correct code.
- You will create one new class that is derived from the MagicalBag class (MagicalBag class will be your base class).
  - Tavern
- Tavern will be a way that we will be storing all of our specialized creatures in a general area, so remember to review your polymorphism!

Tavern Class:

Data Members:

```
//The combined level of all creatures in the Tavern
int totalLevel_

//The total number of creatures in the Tavern
int creatureCount_
```

Member Functions:

```
/**
	Default constructor.
	Default-initializes all private members.
	Default total level count: 0
	Default creature count: 0
*/
Tavern()

/**
	@param: A creature entering the tavern
	@post: If the given creature is not already in the tavern, add the creature to
	the tavern and update the level sum and the creature count
	@return: Returns true if the creature was successfully added to the cavern,
	false otherwise
*/
bool enterTavern(Creature* newCreature)

/**
	@param: A creature exiting the tavern
```

```
        @post: Removes the creature from the cavern, if they exist, and updates the
        level sum and creature count accordingly.
        @return: Returns true if the creature was successfully removed from the
        cavern, false otherwise
*/
bool exitTavern(Creature* newCreature)


/**
        @return: Returns the combined level of all creatures in the tavern
*/
int getLevelSum() const

/**
        @return: Returns the amount of creatures in the tavern
*/
int getCreatureCount() const

/**
        @return: Returns the amount of creatures in the tavern that are hostile
*/
int getHostileCount() const

/**
        @post: Computes the percentage of hostile creatures in the tavern rounded
        up to 2 decimal places
        @return: Returns the percentage of all the hostile creatures in the tavern
*/
double calculateHostilePercentage() const

/**
        @param: A string representing a creature category
        @return: An integer tally of the number of creatures in the tavern of the
        given category
        Hint: If string parameter does not match a category, return 0
```

```
*/
int tallyCategory(const std::string& category) const

/**
        @param: A string representing a creature school
        @return: An integer tally of the number of creatures in the tavern of the
        given school
        Hint: If string parameter does not match a school, return 0
*/
int tallySchool(const std::string& category) const

/**
        @param: An integer representing the level threshold of the creatures to be
        removed from the tavern, with a default value of 0
        @post: Removes all creatures from the tavern whose level is less than the
        given level. If no level is given, remove no creatures. Ignore negative inputs.
        @return: The number of creatures removed from the tavern
*/
int releaseCreaturesBelowLevel(int level)

/**
        @param: A string representing a creature category to be removed from the
        tavern, with a default value of "ALL"
        @post: Removes all creatures from the tavern whose category matches the
        given category. If no category is given, remove all creatures. Ignore invalid
        inputs.
        @return: The number of creatures removed from the tavern
*/
int releaseCreaturesOfCategory(const std::string& category)

/**
        @param: A string representing a creature school to be removed from the
        tavern, with a default value of "ALL"
        @post: Removes all creatures from the tavern whose school matches the
        given school . If no school is given, remove all creatures. Ignore invalid
        inputs.
        @return: The number of creatures removed from the tavern
*/
int releaseCreaturesOfSchool(const std::string& school)

/**
        @post: Outputs a report of the creatures current in the tavern in the form:
        CATEGORY COUNT:
        MAGICAL: [INTEGER]
        UNDEAD: [INTEGER]
        ANIMAL: [INTEGER]
        HUMANOID: [INTEGER]
        CATEGORY_UNKNOWN: [INTEGER]

        SCHOOL COUNT:
        FIRE: [INTEGER]
        ICE: [INTEGER]
        STORM: [INTEGER]
        LIFE: [INTEGER]
        MYTH: [INTEGER]
        DEATH: [INTEGER]
        SCHOOL_UNKNOWN: [INTEGER]

        AVERAGE LEVEL: [INTEGER]
        HOSTILE: [DOUBLE]
*/
void tavernReport() const

/**
        @post: For every creature in the tavern, display each creature's information
*/
void displayCreatures() const
```