ECE2800J Exercise 4

Topics

Exception, ADT, Inheritance.

Background

In the mystical land of Aetheryn, the **Elemental Mage Academy** holds its annual "Grand Mage Trial". Each apprentice must master five elemental forces: **Fire**, **Ice**, **Lightning**, **Earth**, and **Wind**.

To control their magic, every mage carries a **Spellbook**—a magical artifact that records spells and channels arcane energy. But magic is dangerous! Attempting to cast an unlearned spell or exceeding one's mana limit can trigger a **Backfire**, risking explosion, frostbite, or spontaneous levitation!

Kurumi is now a novice elemental mage. She realizes that manually managing spells is error-prone and decides to build a **C++-based Spell Management System** to avoid being blown up by her own Fireball.

She designs a base class `Spellbook` and plans to extend it into a more powerful `MasterSpellbook` for advanced mages. But she's still debugging the mana validation and elemental restrictions...

She needs your help to complete the system!

Predefined Types

- `MAX_SPELLS` - Maximum number of spells that can be stored in a spellbook, set to 5.

```
const int MAX_SPELLS = 5;
```

- `ElementType` - enumeration of five elemental forces.

```
enum ElementType {
    Fire,
    Ice,
    Lightning,
    Earth,
    Wind
};
```

- `elementTypeNames` - corresponding names of `ElementType`.

```
const std::string elementTypeNames[] = {
    "Fire",
    "Ice",
    "Lightning",
```

```
    "Earth",
    "Wind"
};
```

- `Spell` - struct with its constructors. You do not need to consider invalid spells.

```
struct Spell {
    std::string name;    // Name of the spell with at least 1 character.
    ElementType element; // Element type.
    int manaCost;        // Mana cost when casting the spell. Non-negative.

    Spell() : name(""), element(ElementType::Fire), manaCost(0) {}
    Spell(const std::string& n, ElementType e, int c) : name(n), element(e),
manaCost(c) {}
};
```

- `Exception` - class for error messages.

```
class Exception {
    std::string message; // Error message.
public:
    // Constructor with error message.
    Exception(const std::string& msg);

    // Return the error message.
    std::string what() const;
};
```

Your Tasks

You will implement a spell management system using **classes, inheritance, and exception handling**. The goal is to model realistic spell behavior with safety checks and dynamic interactions.

Part 1: `Exception`

- Implement the constructor.
- Implement `what()`.

Part 2: Implement the Base Class `Spellbook`

- Default constructor: Set `maxMana` to 100, set `currentMana` to 50, and set `spellCount` to 0.
- `learnSpell(const Spell& spell)`: Learns a new spell and adds it to the spellbook. Adds from index 0 to 4.
    - If there exists a spell in the spellbook whose name is the same with the new `spell`, overwrite it with the new one. It has the priority over the following `Exception`.

    - If the spellbook reaches its limit (`MAX_SPELLS`), the function throws an `Exception` with the message `"The spellbook is full!"`.

- `castSpell(const std::string& spellName)`: Casts a spell by name. Print success message "Casted [name].\n" to stdout and deducts mana.

    - if the spell with `spellName` is not found, the function throws an `Exception` with the message `"Spell [name] not learned!"`. For example,

```
Spell EX-calibur not learned!
```

    - If mana is not enough, the function throws an `Exception` with the message `"Not enough mana to cast [name]!"`. For example,

```
Not enough mana to cast Magic-Trick!
```

- `printSpells() const`: Prints all spells in index order (from 0 to 4) to stdout. Ignore empty slots.

    - For each spell, the format is `"[Name] ([Element]) - [Cost] mana.\n"`. Printing ends with `"Total spells: [spellCount].\n"`. For example,

```
Magic-Trick (Wind) - 10 mana.
Fire-Ball (Fire) - 5 mana.
Total spells: 2.
```

    - If the spellbook is empty, the function throws an `Exception` with the message `"Spellbook is empty!"`. No other printing needed.

- `restoreMana(int amount)`: Restores mana by `amount` (up to `maxMana`).

    - If `amount <= 0`, the function throws an `Exception` with the message `"Restore amount must be positive!"`. You **must** check it as long as the function is called.

Part 3: Implement the Derived Class `MasterSpellbook`

This class inherits from `Spellbook` and represents an advanced spellbook used by elite mages. Due to its immense power, it enforces stricter rules and may hold less spells.

- `MasterSpellbook(ElementType forbidden, int maxSpells)`: It Sets maxMana to 150, sets currentMana to 100, sets spellCount to 0, sets the forbidden element and the maximum number of spells that can be learned. If `maxSpells > MAX_SPELLS`, set to `MAX_SPELLS`.

    - If `maxSpells < 1`, throws an `Exception` with the message `"The master spellbook can hold at least 1 spell!"`.

- `learnSpell(const Spell& spell)`: Overrides `learnSpell` - forbids learning spells of the banned element.
    - If the element of `spell` is banned, the function throws an `Exception` with the message `"[Element] is forbidden in the master spellbook!"`. For example,

      ```
      Fire is forbidden in the master spellbook!
      ```

    - If there exists a spell in the master spellbook whose name is the same with the new `spell`, overwrite it with the new one. It has the priority over the following `Exception`.
    - If the master spellbook is full, the function throws an `Exception` with the message `"The master spellbook is full!"`.
    - If both exceptions occur, throw an `Exception` with the message `"[Element] is forbidden in the master spellbook!"`. (i.e. The element exception has the priority.)

Implementation Details

- The declarations of the classes and methods are provided for you in `ex4.h`. You should implement the methods in `ex4.cpp`.
- You should **not** modify the provided `ex4.h` file. You can check details in the file.
- You should throw an `Exception` object when an error occurs along with the corresponding error message. Only the mentioned errors above will occur.
- Integer inputs will be bounded in $[-2^{31}, 2^{31} - 1]$.
- **Pay attention to any typo or formatting errors.** They will be treated as failed test cases.
- Do **not** add a newline to any `Exception` messages.

Submission

Submit `ex4.cpp` on gitea under `ex4` directory. Release it when you are ready. Due time is 11.23 23:59.