Objektorientierte Sprachen

Sommersemester 2021 - Vierte Prüfung

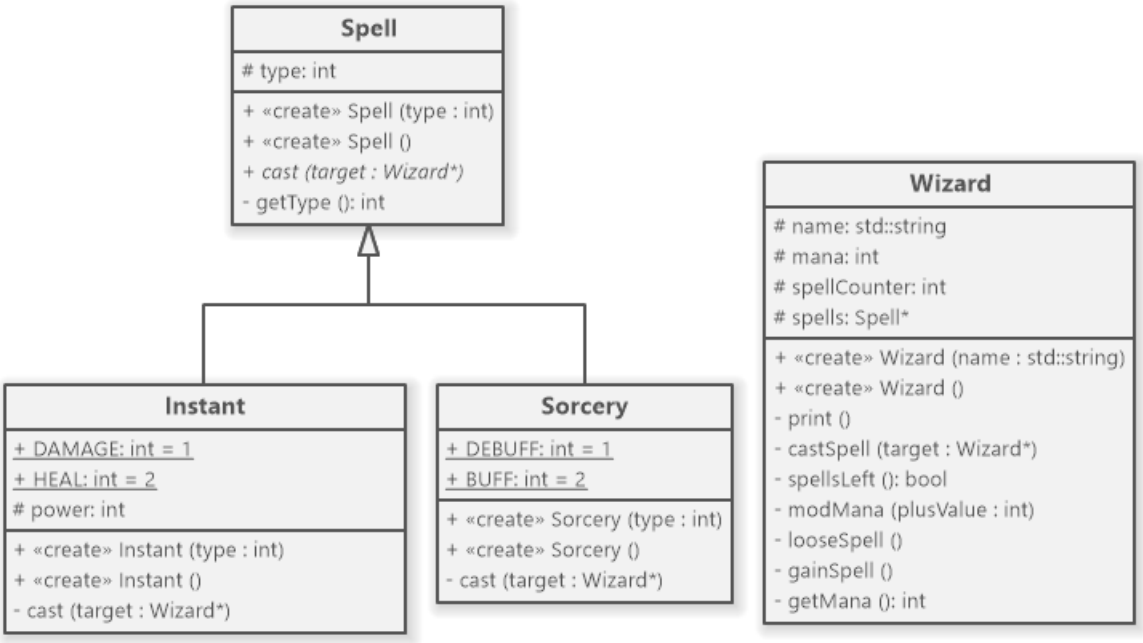
An- und Abgabe

Implementieren Sie ein Programm das zwei Zauberer nacheinander Sprüche wirken lässt, bis keiner der beiden mehr Sprüche übrig hat.
Sie haben dazu eine Grundstruktur vorgegeben, die bereits eine main-Funktion enthält, sowie die Header- und CPP-Dateien für die abstrakte Spell-Klasse bereitstellt. Erweitern Sie nun diese Grundstruktur, indem Sie die abstrakte Klasse ausbauen, zwei Klassen implementieren, die von ihr erben und die Wizard-Klasse komplett definieren und implementieren.

Laden Sie Ihre Abgabe in einem ZIP-Archiv verpackt auf Moodle hoch, fehlerhaft oder unvollständig hochgeladene Abgaben können nicht nachgereicht werden.
Die Abgabe ist bis 19:20 möglich.

Die Grundstruktur ist im Code durch Kommentare erklärt und auf der nächsten Seite finden Sie ein Klassendiagramm sowie eine Zusammenfassung des Programmablaufs.

Klassendiagramm



Übersicht

- Zu Beginn des Programms werden zwei neue Wizard-Objekte angelegt, dabei wird dem Konstruktor ein Name für das Objekt übergeben. **Implementieren Sie den Konstruktor der Klasse Wizard.**
- Im Konstruktor der Wizard-Klasse erhält der Zauberer sieben zufällige Sprüche, die entweder Instant- oder Sorcery-Objekte sind. Beide Klassen sollen von der abstrakten Klasse Spell erben. **Definieren Sie die rein virtuelle cast-Funktion in der Klasse Spell, führen Sie dann die Vererbung an die beiden Kindklassen durch und implementieren Sie deren Konstruktoren und cast-Funktionen.**
- Danach werden solange Sprüche der Zauberer gewirkt, bis kein Zauberer mehr Sprüche übrig hat. Wird ein Spruch gewirkt kann dieser nicht noch einmal verwendet werden. **Implementieren Sie die castSpell-Funktion der Zauberer und zur Prüfung, ob ein Zauberer noch Sprüche übrig hat die spellsLeft-Funktion.**
- **Implementieren Sie auch die print-Funktion**, zur Ausgabe des Zustands des Zauberers.
- Während der Ausführung der castSpell-Funktion kann es zu Exceptions kommen, **implementieren Sie dafür eine entsprechende Fehlerbehandlung in der main-Funktion**.
- Zum Schluss soll natürlich auch jeglicher Speicher korrekt freigegeben werden, **implementieren Sie dazu den Destruktor der Klasse Wizard.**

Aufgaben

main-Funktion

- Implementieren Sie eine Fehlerbehandlung durch Exceptions, für den Aufruf der castSpell-Funktion, fangen Sie dabei sowohl eine invalid_argument-Exception als auch allgemeine Objekte ab.

Abstrakte Klasse Spell

Definieren Sie folgende **rein virtuelle** Funktion für die abstrakte Klasse:

- **cast(Wizard* target):** Die Funktion soll keinen Wert zurückgeben.

Klasse Instant

Definieren Sie die Klasse Instant, die von Spell erben soll, aufgeteilt in separate Header- und CPP-Dateien. Es gibt zwei Typen von Instant-Objekten, schädigende und kräftigende. Diese Klasse soll ein zusätzliches Integer-Attribut power besitzen. Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor(int type):** Der power-Wert soll zufällig zwischen 1 und 5 liegen. Der Typ soll dem übergebenen Parameter entsprechen und markiert ob es sich um einen schädigenden oder kräftigenden Spruch handelt.
- **cast(Wizard* target):** Ist der Spruch schädigend soll das als Parameter übergebene Ziel Mana in Höhe des power-Werts verlieren. Ist der Spruch kräftigend erhält das Ziel Mana in Höhe des power-Werts.

Klasse Sorcery

Definieren Sie die Klasse Sorcery, die von Spell erben soll, aufgeteilt in separate Header- und CPP-Dateien. Es gibt zwei Typen von Sorcery-Objekten, debuffende und buffende.Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor(int type):** Der Typ soll dem übergebenen Parameter entsprechen und markiert ob es sich um einen debuffenden oder buffenden Spruch handelt.
- **cast(Wizard* target):** Ist der Spruch debuffend soll das als Parameter übergebene Ziel seinen nächsten Spruch verlieren. Ist der Spruch buffend erhält das Ziel, wenn es weniger als 7 Sprüche hat einen zusätzlichen Spruch. Achten Sie dabei auf die Speicherverwaltung.

Klasse Wizard

Definieren Sie die Klasse Wizard aufgeteilt in separate Header- und CPP-Dateien. Die Klasse soll ein siebenstelliges Array enthalten das aus Spell-Pointern besteht, damit dann über

Polymorphismus auf Instant- und Sorcery Objekte zugegriffen werden kann. Weiters hat jeder Zauberer einen Namen (String) und einen Mana-Wert (Integer). Implementieren Sie folgende Funktionen für die Klasse:

- **Konstruktor(string name):** Der Konstruktor erhält den Namen als Parameter und soll ihn entsprechend setzen. Weiters wird der Mana-Wert auf 20 gesetzt und sieben zufällige Sprüche für das Spell-Pointer-Array erstellt werden (50:50 Wahrscheinlichkeit sowohl ob es sich um Instant oder Sorcery sowie ob es sich um schädigend/debuffend bzw. kräftigend/buffend handelt).
- **Destruktor:** Der Destruktor gibt den Speicher, der im Konstruktor für die Sprüche beansprucht wurde wieder frei.
- **print():** Gibt den Namen und Mana-Wert des Zauberers aus.
- **castSpell(Wizard* target):** Ist der Parameter null wird eine invalid_argument-Exception geworfen.
  Der Zauberer wirkt den nächsten Spruch der am Ende seines Spell-Pointer-Arrays liegt, diesen Spruch kann er danach nicht nochmal wirken. Als Ziel wählt er bei schädigenden/debuffenden Sprüchen den übergebenen Parameter und bei kräftigenden/buffenden Sprüchen sich selbst.
- **spellsLeft():** Die Funktion soll prüfen ob der Zauberer noch Sprüche übrig hat, die er wirken kann oder nicht.
- Sie können beliebige weitere Funktionen implementieren, die Sie für die Abhandlung von Sprüchen brauchen.

Benotungsaspekte

Aspekt	Bewertung
Spell	15%
Instant	7.5%
Sorcery	7.5%
Wizard	20%
Fehlerbehandlung mit Exceptions	15%
Vererbung, Verwendung von virtual und pure virtual	25%
Speicherverwaltung	10%
Gesamt	100%

Compiler-Warnungen, Speicherfehler und schlechter Stil können zu Punkteabzügen führen.

Benotung

$\geq 88\%$	Sehr gut
$\geq 75\%$ und $< 88\%$	Gut
$\geq 63\%$ und $< 75\%$	Befriedigend
$\geq 50\%$ und $< 63\%$	Genügend
$< 50\%$	Nicht genügend