Objektorientierte Sprachen

Sommersemester 2021 - Dritte Prüfung

An- und Abgabe

Implementieren Sie ein Programm, in dem ein heldenhafter Kämpfer in einer Arena gegen eine Reihe von Monstern antritt, bis entweder alle Monster oder der Held besiegt wurden.

Sie haben dazu eine Grundstruktur vorgegeben, die bereits eine main-Funktion und einige Ausgaben bereitstellt. Erweitern Sie nun diese Grundstruktur, anhand der Angabe, indem Sie die abstrakte Klasse Creature, sowie die von ihr erbenden Monster und den Helden implementieren. Die Arena-Klasse soll dabei die Monster verwalten und die Kämpfe durchführen.

Laden Sie Ihre Abgabe in einem ZIP-Archiv verpackt auf Moodle bis zum angegebenen Zeitpunkt hoch, fehlerhaft oder unvollständig hochgeladene Abgaben können nicht nachgereicht werden.

Übersicht

- Zu Beginn des Programms soll ein Held erstellt werden sowie die Arena mit 5-15 zufällig ausgewählten Monster befüllt werden. Die Anzahl soll ebenfalls zufällig sein.
- Dann sollen, solange noch Monster in der Arena sind und der Held noch am Leben ist, Kämpfe durchgeführt werden, indem die attackMonster-Methode der MonsterHandler-Klasse aufgerufen wird.
- Implementieren Sie 2 Monster: Zombie und Gobli, die von dem Interface Creature erben.
- Die attackHero-methode nimmt eine zufällige Nummer und den Helden als Creature-Referenz. Sie soll dann den Angriffsschaden des Helden dem Monster zufügen woraufhin sich das Monster verteidigen kann, wenn es noch am Leben ist.
- Das Programm endet, wenn alle Monster oder der Held besiegt wurden. Achten Sie auf die korrekte Freigabe von Speicher.

Klassen & Methoden:

main-Funktion

- Ergänzen Sie hier die Erstellung der Monster & die Zuweisung zur Arena mit der addMonster-Methode

Abstrakte Klasse Creature

Creature bildet die Basis für alle Monster und Helden. Sie soll folgende Methoden beinhalten:

- std:string getName(): Gibt den Namen der Kreatur als string zurück.
- bool isAlive(): Gibt true zurück wenn die Kreatur noch am Leben ist (üblicherweise >0 Lebenspunkte).
- int getDamage(): Gibt den Schaden für einen Angriff der Kreatur zurück.
- dealDamage(int damage): Hat keinen Rückgabewert aber fügt die übergebenen Schadenspunkte der Kreatur zu.

Entscheiden Sie selbst welche dieser Methoden rein virtuell sein sollen und wo eine generische Implementierung Sinn macht. Sie können diese Klasse auch um Attribute oder andere Methoden erweitern.

Klasse Held:

Der Held erbt von der Creature-Klasse und hat folgende Eigenschaften:

- Name: frei wählbar
- Lebenspunkte: 20
- Der Schaden soll bei jedem Angriff zufällig 1,2 oder 3 sein.

Klasse Goblin:

Der Goblin erbt von der Creature Klasse und hat folgende Eigenschaften:

- Name: Goblin
- Lebenspunke: 3
- Schaden: 2

Klasse Zombie:

Der Zombie erbt von der Creature Klasse und hat folgende Eigenschaften:

- Name: Zombie
- Schaden: 1
- Der Zombie kann nur besiegt werden, wenn ein in einem Schlag 3 Schadenspunkte auf einmal bekommt

Klasse Arena:

Die Arena Klasse verwaltet die Monster in einer geeigneten Datenstruktur und führt die Kämpfe über die attackMonster()-Methode durch. Implementieren Sie:

- attackMonster(int monsterNumber, Creature& attacker): Diese Methode hat eine Nummer und eine Kreaturen-Klasse als Parameter (der Held). Bei einer gültigen Nummer fügt der Held zuerst seine Schadenspunkte der Kreatur zu und danach kann sich die Kreatur verteidigen, wenn sie noch am Leben ist. Die Ausgaben hierzu sind bereits vorhanden dürfen aber modifiziert werden, falls notwendig.
- Achten Sie auf die korrekte Fehlerbehandlung bei ungültigen Monsternummern. Verwenden Sie hierzu Exceptions und achten Sie auf deren korrekte und sinnvolle Behandlung!
- Int getMonsterAmount(): Gibt die Anzahl an Monstern aus, die noch am Leben sind.
- addMonster(Creature* newCreature): fügt der Arena ein neues Monster der Creature-Klasse hinzu.
- Achten Sie auf die korrekte Freigabe von Speicher.

Hinweise:

- Trennen Sie Ihr Programm in Header & cpp-Dateien!
- Die vorgegebenen Ausgaben sollen Ihnen als Hilfe dienen, Sie können diese aber verändern, wie Sie wollen.
- Implementieren Sie zuerst den Goblin, bis ihr Programm funktioniert.

Benotung

Aspekt	Bewertung
Creature	15%
Hero	7.5%
Goblin	5%
Zombie	7.5%
Arena	15%
Fehlerbehandlung mit Exceptions	15%
Vererbung, Verwendung von virtual und pure virtual	25%
Speicherverwaltung	10%
Gesamt	100%

≥88%	Sehr gut
≥75% und <88%	Gut
≥63% und <75%	Befriedigend
≥50% und <63%	Genügend
<50%	Nicht genügend