Objektorientierte Sprachen

Sommersemester 2022 - Dritte Prüfung

An- und Abgabe

Implementieren Sie ein Programm, das Single und Multiple Choice Fragen generieren und benoten kann.

Sie haben dazu bereits ein grundlegendes Prüfungsprogramm gegeben, dass aus einem Exam Object besteht, dass mit Question-Objekten befüllt werden kann. Erweitern Sie das Programm um die zwei Fragentypen, die von Question erben sollen.

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

- Es existiert bereits ein Grundkonstrukt, wo allerdings noch nicht alles implementiert ist, implementieren Sie falls notwendig Konstruktoren, Destruktoren und die nextQuestion-Methode.
- Es soll 2 neue Fragetypen geben: Single Choice und Multiple Choice. Bei Single Choice Fragen darf nur eine Antwort als richtig ausgewählt werden, bei Multiple Choice Fragen können es mehrere sein.
- In der gegebenen Vorlage befindet sich bereits Code, um eine Prüfung zu generieren und zu testen. Dieser Code ist derzeit noch auskommentiert.
- Es steht Ihnen frei, zusätzlich zu den Gegebenen eigene Methoden, Attribute oder Klassen zu implementieren.

Klassen & Methoden:

main-Funktion

Implementieren Sie wo notwendig richtige Fehlerbehandlung mit Exceptions.

Klasse Question

Erweitern Sie die Klasse um alle Eigenschaften, die Single und Multiple Choice Fragen gemeinsam haben. Entscheiden Sie welche Methoden in der Elternklasse implementiert werden und welche nur rein virtuelle Funktionen sein sollen.

- void addAnswer (string name, bool isCorrect): Diese Methode fügt der Frage eine neue mögliche Antwort hinzu, die entweder richtig oder falsch sein kann.

Klasse SingleChoiceQuestion:

- void markAnswer(int answerNumber): Speichert die Antwort als ausgewählt für die Auswertung. Falls die übergebene Nummer nicht eine der möglichen Optionen ist, soll eine sinnvolle Exception (invalid_argument) geworfen werden. Falls bereits eine Antwort gewählt wurde, soll eine sinnvolle andere Exception (logic_error) geworfen werden.
- int getPoints(): Berechnet die Punkte der Frage. Falls die richtige Antwort gewählt wurde, ist das 1x pointValue, ansonsten 0.

Klasse MultipleChoiceQuestion:

- void markAnswer(int answerNumber): Speichert die Antwort als ausgewählt für die Auswertung. Falls die übergebene Nummer nicht eine der möglichen Optionen ist, soll eine sinnvolle Exception geworfen werden.
- int getPoints(): Berechnet die Punkte der Frage.
    - Für jede richtige Antwort, die gewählt wurde, ist das 1x pointValue
    - Für jede falsche Antwort, die gewählt wurde und jede richtige Antwort, die nicht gewählt wurde, wird jeweils 1x pointValue abgezogen
    - Die Gesamtpunkte der Frage dürfen aber nicht unter 0 sein.

Klasse Exam:

- Ergänzen Sie, falls nötig, den Konstruktor und Destruktor
- void getCurrentQuestion(): Erweitern Sie diese Methode, sodass sie nicht immer nur die erste Frage zurückgibt, sondern nach Aufruf von nextQuestion() jeweils die nächste.
- void nextQuestion(): nach Aufruf diese Methode soll der nächste Aufruf von getCurrentQuestion(), die nächste Frage zurückgeben. Falls es keine weitere Frage mehr gibt, soll der Aufruf von getCurrentQuestion() einen nullptr zurückgeben.
- Speichern Sie die aktuelle Frage auf sinnvolle Weise im Exam Objekt
- int grade(): Ruft die getPoints()-Methode von allen Fragen auf und gibt deren Summe zurück.

Klasse/Struct Answers:

- string name
- bool isCorrect

Benotung

Aspekt	Bewertung
Klasse Question	15%
Klasse SingleChoiceQuestion	15%
Klasse MultipleChiceQuestion	15%
Klasse Exam	15%
Fehlerbehandlung mit Exceptions	15%
Sinnvolle Vererbung & Verwendung von virtual und pure virtual	15%
Speicherverwaltung	10%
Gesamt	100%