Praktikum 2: Objektorientierte Programmierung

1. Lernziele

Die folgenden, in der Vorlesung behandelten Themen sollen vertieft und angewendet werden:

- Objektorientierte Programmierung
- Klassen- und Objektmodell
- Sichtbarkeit von Attributen und Methoden durch die Zugriffsmodifikatoren `public` und `private`
- Exceptions und Errorhandling
- Generische Datentypen

2. Aufgabe

In dieser Praktikumsaufgabe soll die aus Praktikum 1 bekannte Vorrangwarteschlange durch eine Klassenstruktur aus C++ ersetzt werden.

Teil 1: Erstellen Sie die Klasse `PriorityQueue` analog zur Praktikumsaufgabe 1. In der Klasse soll durch Methoden die gleiche Funktionalität enthalten sein, wie sie in Aufgabe 1 durch die C-Funktionen gegeben war. Implementieren Sie hierzu die folgenden Methoden der Klasse `PriorityQueue`:

- `void insert(std::string value, float priority)`
  fügt den Wert `value` mit der Priorität `priority` in die Vorrangwarteschlange ein.
- `std::string extractMin(void)`
  liefert den Wert aus der Vorrangwarteschlange mit höchster Priorität (→ kleinster nummerischer Wert des Attributs `priority`).
- `void decreaseKey(std::string value, float priority)`
  ändert die Priorität des Wertes `value` auf den neuen Wert `priority`.
- `void remove(std::string value)`
  löscht den Wert `value` aus der Vorrangwarteschlange.
- `bool isEmpty()`
  gibt `true` zurück, wenn die Vorrangwarteschlange leer ist.

Ersetzen Sie die C-Funktionen `pqueue_create` und `pqueue_destroyPriorityQueue` durch einen Konstruktor und Destruktor. Zur Fehlerbehandlung verwenden Sie Exceptions. Erstellen Sie hierzu eine Fehlerklasse `QueueException`, die einen Fehlertext speichern kann. Implementieren Sie ein entsprechendes Exceptionhandling für die Methoden `extractMin`, wenn keine Elemente in der Queue sind und `decreaseKey`, wenn das Element nicht gefunden wurde.

Nutzen Sie zur Datenkapselung die in C++ zur Verfügung stehende Möglichkeit den Zugriff auf Attribute und Methoden mit `public` und `private` zu regeln. Die Implementierung soll einen Zugriff auf die interne Repräsentation verhindern.

Realisieren Sie die Speicherverwaltung mit `new` und `delete`.

Erstellen Sie anschließend eine weitere Version ihrer Warteschlange, in der die Werte durch einen generischen Datentyp ersetzt werden (Templates).

Teil 2: Um die Korrektheit Ihrer Implementierung nachzuweisen, schreiben Sie einen Testtreiber (Programm), der Ihre Datenstruktur testet.
Tests müssen vollständig automatisiert ablaufen.

3. Testat

Voraussetzung ist jeweils ein fehlerfreies, korrekt formatiertes Programm. Der korrekte Programmlauf muss nachgewiesen werden. Sie müssen in der Lage sein, Ihr Programm im Detail zu erklären und ggf. auf Anweisung hin zu modifizieren.

Das Praktikum muss spätestens zu Beginn des nächsten Praktikumtermins vollständig bearbeitet und abtestiert sein.