Praktikum 2 zur OOS SS16 G1: 06.05. & G2: 29.04.

Ziel:

Es soll in diesem Versuch das grundlegende Konzept einer persistenten Nutzerverwaltung implementiert werden.

Hierzu sind die oo-Konzepte des Interfaces und implementierender Klassen, sowie die Java-spezifischen Möglichkeiten zur Serialisierung und Deserialisierung zu verwenden.

Außerdem wird das Konzept der Exceptionverarbeitung geübt.

Hinweise:

Gehen Sie so vor, dass Sie zuerst überlegen, welche Tests Sie anwenden und implementieren Sie diese zuerst im main-Programm. Lassen Sie sich die entsprechenden Methoden aus den Tests heraus so weit wie möglich erzeugen und implementieren Sie diese.
Sie können Klassen der Java-Bibliothek verwenden. Das erleichtert die Arbeit ungemein!

Aufgabe 1: (Nutzerverwaltung ohne Persistenz)

Es müssen zwei Klassen `Benutzer` und `BenutzerVerwaltungAdmin` implementiert werden, wobei letztere das Interface `BenutzerVerwaltung` implementiert.

a) Implementieren Sie die Klasse `Benutzer`, die alle Informationen über Benutzer des Systems enthält.
Die Klasse enthält die *Attribute* `userId` und `passWort`, die als `String` bei `userId` und als `char-Array` bei `passWort` deklariert werden.
Als *Konstruktoren* soll neben dem Default-Konstruktor noch ein Konstruktor implementiert werden, der beide Attribute initialisiert.
Als *Methoden* sollen lediglich die beiden Standardmethoden `equals` und `toString` implementiert werden. Hierbei ist `equals` als *Überlagerung* der gleichen Methode der Klasse `Object` zu implementieren.
Berücksichtigen Sie hierbei, dass für `char`-Arrays weder `toString` noch `equals` überlagert wurden. Sie können dies umgehen, durch Verwendung der statischen Methode `copyValueOf` der Klasse `String`.

b) Definieren Sie das Interface `BenutzerVerwaltung`, das die beiden Methoden
- `void benutzerEintragen(Benutzer benutzer)` und
- `boolean benutzerOk(Benutzer benutzer)`
die allen Nutzern des Systems, insbesondere auch den späteren Clients, zur Verfügung stehen, deklariert.
Die Semantik der beiden Methoden ist wie folgt definiert:
- `void benutzerEintragen(Benutzer benutzer)`
Das Parameterobjekt wird in eine Datenhaltung eingetragen.
Überlegen Sie sich sinnvolle Ausnahmemöglichkeiten für diese Methode und leiten Sie die dabei auftretenden Exceptions an den Aufrufer weiter.
- `boolean benutzerOk(Benutzer benutzer)`
liefert `true`, falls das Parameterobjekt in der Datenhaltung vorhanden ist, sonst `false`.

c) Implementieren Sie das Interface `BenutzerVerwaltung` durch die Klasse `BenutzerVerwaltungAdmin`, die neben den Datenstrukturen zur Implementierung der Datenhaltung und den beiden Methoden des Interfaces auch noch die folgende Methode implementiert:
- `void benutzerLöschen(Benutzer benutzer)`
Das Parameterobjekt wird aus der Datenhaltung entfernt.
Werfen Sie für diese Methode und für die Methode `benutzerEintragen`

selbstdefinierte sinnvolle Exceptions, die Sie direkt von der `Exception` Klasse ableiten und an den Aufrufer weiter leiten.
Verbieten Sie den Zugriff auf die Datenstrukturen zur Implementierung der Datenhaltung von anderen Klassen aus.

d) Überlegen Sie sich sinnvolle Testfälle, die alle möglichen Fehlerquellen abdecken und führen Sie diese in einem `main`-Programm aus.

Lösen Sie Aufgabe 2 erst, wenn Sie von der Fehlerfreiheit Ihrer Implementierung überzeugt sind!

Aufgabe 2: (Persistenz der Nutzerverwaltung)

Im Folgenden sollen die Datenstrukturen der Nutzerverwaltung persistent als Datei auf dem Dateisystem des Rechners abgelegt werden.

a) Erweitern Sie die Klasse `BenutzerVerwaltungAdmin` um eine Methode `dbInitialisieren()`, die ein neues leeres Objekt Ihrer Datenstruktur anlegt, serialisiert und dieses unter einem von Ihnen zu wählenden Namen im Dateisystem ablegt.
Diese Methode wird wie die Methode `benutzerLöschen` nur lokal aufgerufen und deshalb ebenfalls nicht im Interface `BenutzerVerwaltung` zur Verfügung gestellt.

b) Erweitern Sie die Implementierungen der drei anderen Methoden der Klasse `BenutzerVerwaltungAdmin`, so dass Änderungen und Überprüfungen in der Datenstruktur immer auf der von der Methode `dbInitialisieren()` angelegten Datei durchgeführt werden und somit persistent gemacht werden.
Lesen Sie hierzu jeweils die aktuelle Datenstruktur aus dem File, deserialisieren Sie diese und verweisen Sie mit dem `private`-Attribut auf das erhaltene Objekt.
Führen Sie dann die Änderungen oder Überprüfungen darauf aus.
Im Falle einer durchgeführten Änderung serialisieren Sie danach die veränderte Datenstruktur wieder und überschreiben damit die Datei in Ihrem Dateisystem.

c) Verwenden Sie die Testfälle aus Aufgabe 1 und führen Sie diese jeweils in einem `main`-Programm aus.
Testen Sie insbesondere, ob sich Ihr Programm sowohl mit einer initialisierten Datenstruktur, als auch mit einer nicht initialisierten, sondern schon bearbeiteten korrekt verhält.