Projektblatt P8 (24P)

Abgabe: Freitag 10. November 2023, 12:00h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p08.zip`, in der sich die Rahmendateien für die zu lösenden Aufgaben befinden. Die Dateien sind hierbei verteilt in einer hierarchischen Verzeichnisstruktur (siehe auch Aufgabenstellung). Ergänzen Sie die Dateien durch Ihre Lösungen gemäß der Aufgabenstellung unten. Der hinzuzufügende Java-Sourcecode sollte **syntaktisch richtig** und **vollständig formatiert** sein. Alle Dateien sollten am Ende fehlerfrei übersetzt werden können.
Verpacken Sie die `.java` Dateien für Ihre Abgabe in einem ZIP-Archiv mit dem Namen `IhrNachname.IhrVorname.P08.zip`, welches Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie das Unterverzeichnis `p08` liegt, folgenden Befehl auf der Kommandozeile aus:

```
zip -r IhrNachname.IhrVorname.P08.zip p08
```

Aufgabe 1: 24 (= 7+9+3+5)Punkte

Ein Stack kann als eine Liste von Elementen aufgefasst werden, bei der das Einfügen und Entnehmen von Elementen nur an einer Seite (z.B. am Ende) erlaubt ist. Sie sollen in dieser Aufgabe zwei verschiedene Listenklassen so erweitern, dass Sie als Stack verwendet werden können. Hierzu sollen Sie zwei neue Klassen schreiben, die von zwei vorgegebenen Listenklassen abgeleitet sind und ein ebenfalls vorgegebenes Interface implementieren. Dazu sollen Sie eine Datenklasse (für die Listenelemente) vervollständigen und Testcode schreiben.
Hierzu ist Folgendes vorgegeben:

Das Interface `p08.types.Stack` (im Unterverzeichnis `p08/types`) spezifiziert Folgendes:

- Die Konstante `MAX_SIZE` gibt die maximal erlaubte Anzahl von Elementen an, die auf dem Stack gespeichert werden können.
- Die Methode `push` soll ein Objekt, welches als Parameter übergeben wird, auf dem Stack ablegen. Dieses Objekt soll der abstrakten Klasse `Element` im Package `p08.data` angehören, welche Sie dementsprechend in dem Unterverzeichnis `p08/data` finden. Ist noch genug Platz auf dem Stack, so dass ein Element erfolgreich eingefügt werden kann, dann soll nach dem Einfügen der Wert `true` zurückgegeben werden. Ist der Stack voll, d.h. enthält er bereits die maximal erlaubte Anzahl von Elementen, soll nur der Wert `false` zurückgegeben werden.
- Die Methode `pop` soll ein Element vom Stack (das Element, welches als letztes hinzugefügt wurde) entfernen und als Resultat zurückgeben. Ist der Stack leer, soll eine Nullreferenz zurückgegeben werden.
- Die Methode `peek` soll das oberste Element vom Stack zurückgeben, ohne es vom Stack zu entfernen. Ist der Stack leer, soll eine Nullreferenz zurückgeben werden.
- Die Methode `size` soll die Anzahl der aktuell auf dem Stack gespeicherten Elemente zurückgeben.
- Die Methode `toArray` soll ein Feld anlegen und zurückgeben, welches (Referenzen auf) die aktuell auf dem Stack liegenden Elemente enthält.

Die vorgegebene Klasse `p08.data.Element` stellt eine abstrakte Oberklasse einer Elementklasse dar, deren Objekte in einer Liste / auf einem Stack gespeichert werden sollen. Sie besitzt nur eine Instanzvariable, welche mit Hilfe des Konstruktors und einer Klassenvariablen mit einem eindeutigen ganzzahligen Wert initialisiert wird. Darüberhinaus ist die `toString`-Methode überschrieben.
Die von Ihnen später noch zu ergänzende Klasse `p08.aufgabe_1.StudentElement` repräsentiert ein*n Studierende*n mit einem Namen und einer Matrikelnummer. Neben den Instanzvariablen ist ein Konstruktor vorgegeben und die aus der Klasse Object geerbten Methoden `toString` und `equals` sind überschrieben.
Die Klasse `p08.types.List` im Unterverzeichnis `p08/types` definiert neben einer inneren Klasse für Listenelemente einen Konstruktor und vier Methoden:

- Die private Methode `insert` fügt ein als Parameter übergebenes, neues Element nach einem Listenelement ein, welches ebenfalls als Parameter übergeben wird.
- Die öffentliche Methode `insertAt` fügt ein neues Element an k. Position ein ($k = 1, 2, \dots$), sofern die Liste mindestens $k - 1$ Elemente hat.
- Die Methode `removeElementAt` löscht das k. Element ($k = 1, 2, \dots$) in der Liste, sofern mindestens `k` Elemente in der Liste sind. Das aus der Liste gelöschte Element wird von der Methode zurückgegeben.
- Die Methode `getElementAt` liefert das k. Element ($k = 1, 2, \dots$) in der Liste zurück, sofern mindestens `k` Elemente in der Liste sind.
- Die Methode `size` gibt die (aktuelle) Anzahl der Elemente in der Liste zurück.

Mit Hilfe der Klasse `p08.aufgabe_1.TestStack` können Sie Ihre Implementierung später testen. Hierzu müssen Sie Klasse später noch ergänzen.
Es ist Ihnen im Folgenden nicht getattet, die vorgegebenen Klassen und das Interface in den Packages `p08.types` und `p08.data` zu verändern.

Implementieren Sie nun zunächst zwei Klassen `ArrayStack` und `ListStack` im Package `p08.aufgabe_1`, die beide das Interface `Stack` implementieren sollen, aber die Elemente der Liste jeweils unterschiedlich verwalten: in einem

Feld bzw. in einer Liste. Anschließend sollen Sie die Klasse `StudentElement` im Package `p08.aufgabe_1` ergänzen, so dass diese in den beiden Stack-Klassen verwendet werden kann und schließlich sollen Sie die Test-Klasse vervollständigen.

(a) Legen Sie zunächst im Package (also im zugehörigen Unterverzeichnis) `p08.aufgabe_1` eine neue Datei für die Klasse `ListStack` an. Implementieren Sie die Klasse wie folgt:

- Fügen Sie die Package-Definition hinzu und importieren Sie die Klassen `Element` und `List` sowie das Interface `Stack` aus den Packages `p08.data` und `p08.types`.
- Definieren Sie die Klasse als Unterklasse der Klasse `List` und geben Sie an, dass die Klasse das Interface `Stack` implementiert.
- Implementieren Sie dann die Methoden des Interface so, dass die Methoden wie oben beschrieben arbeiten. Sorgen Sie dabei dafür, dass nicht mehr als die maximale Anzahl von Elementen in der Liste gespeichert werden.

(b) Legen Sie im Package `p08.aufgabe_1` nun eine neue Datei für die Klasse `ArrayStack` an. Implementieren Sie die Klasse wie folgt:

- Fügen Sie die Package-Definition hinzu und importieren Sie die Klassen `Element` sowie das Interface `Stack` aus den Packages `p08.data` und `p08.types`.
- Definieren Sie die Klasse so, dass die Klasse das Interface `Stack` implementiert.
- Die Klasse `ArrayStack` soll zur Speicherung der Elemente ein Feld mit einer Länge von `MAX_SIZE` verwenden. Hinweis: Verwenden Sie eine private Instanzvariable, um den Index des zuletzt eingefügten Werts zu speichern und nutzen Sie diese bei der Implementierung der Methoden des Interface `Stack`.
- Implementieren Sie dann die Methoden des Interface so, dass die Methoden wie oben beschrieben arbeiten. Sorgen Sie auch hier dafür, dass nicht mehr als die maximale Anzahl von Elementen in dem Feld gespeichert werden.

(c) Erweitern Sie nun die Klasse `p08.aufgabe_1.StudentElement` wie folgt:

- Erweitern Sie die Klassendefinition so, dass diese Klasse von der Klasse `Element` aus dem Package `p08.data` abgeleitet wird. Importieren Sie diese Klasse mit einem entsprechenden `import` Statement.
- Entfernen Sie den Blockkommentar um die Methode `compareTo` und ergänzen Sie die Methode wie folgt:
  - Wenn das als Parameter übergebene Objekt vom Typ `StudentElement` ist (nutzen Sie hierzu den instanceof Operator), vergleichen Sie zunächst die Namen der beiden Objekte (nutzen Sie dazu die `compareTo`-Methode in der Klasse `String`). Falls die Namen gleich sind, vergleichen Sie die Matrikelnummern. Hinweis: Sie müssen das als Parameter übergebene Objekt jeweils auf die Klasse `StudentElement` casten, bevor Sie auf die Attributwerte zugreifen können.
  - Wenn das als Parameter übergebene Objekt nicht vom Typ `StudentElement` verwenden Sie die bereits vorgegebene Codezeile, um eine Exception zu generieren.

(d) Ergänzen Sie schließlich die `main`-Methode der im Package `p08.aufgabe_1` vorgegebene Klasse `TestStack` wie folgt:

- Entfernen Sie den Blockkommentar um die Definition des Felds `elements`.
- Erzeugen Sie je eine Instanz der Klasse `ListStack` und `ArrayStack` und legen Sie diese in einem Feld vom `Stack` Elememten ab.
- Führen Sie dann für beide `Stack`-Instanzen in dem Feld jeweils folgende Aktionen durch:
  - Befüllen Sie die Instanz mit den Elementen aus dem Feld `elements` mit Hilfe der Methode `push`.
  - Holen Sie danach mit Hilfe der Methode `pop` wieder alle Elemente vom Stack und geben Sie diese auf der Kommandozeile aus. Nutzen Sie hierbei die Methode `size` aus dem Interface Stack, um genau alle Elemente vom Stack zu holen (und nicht den Wert `elemente.length`).

– Befüllen Sie den Stack erneut mit allen Elementen aus dem Feld `elemente` und lassen Sie sich den Inhalt des Stacks anschließend mit Hilfe der Methode `toArray` in Form eines Felds liefern. Sortieren Sie dieses mit Hilfe der statischen Methode `sort` aus der Klasse `java.util.Arrays` und geben Sie den Inhalt des Felds schließlich elementweise aus.

Um die Klassen erfolgreich übersetzen bzw. ausführen zu können, sollten Sie den CLASSPATH dynamisch um das Verzeichnis erweitern, in dem das Verzeichnis `p08` liegt. Unter Linux (in einer Bash) also wie folgt, wenn das Verzeichnis `p08` beispielsweise im Unterverzeichnis `pp1` liegt, was wiederum direkt in Ihrem Home-Verzeichnis liegt.

```
export CLASSPATH=$HOME/pp1:$CLASSPATH
```

Für MAC-User ist hierzu das folgende YT Video zu empfehlen:
`https://www.youtube.com/watch?v=y60prRFl9VM`
Wechseln Sie zum Übersetzen der Klassen im Unterverzeichnis `aufgabe_1` dann in das Verzeichnis, in dem das Verzeichnis `p08` liegt und übersetzen Sie eine Klasse mit dem Namen `X.java`, die in diesem Verzeichnis liegt, dann mit dem folgenden Befehl:

```
javac p08/aufgabe_1/X.java
```

Das Programm sollte dann wie folgt aufgerufen werden können:
`java p08.aufgabe_1.TestStack`

Die Ausgabe des Programms sollte wie folgt aussehen:

```
Stack 1:
Suse (10000001)
Tobi (10203040)
Tim (11336699)
Anna (12312300)
Tom (11111111)
Tom (12131415)
---------------
Anna (12312300)
Suse (10000001)
Tim (11336699)
Tobi (10203040)
Tom (11111111)
Tom (12131415)
---------------
Stack 2:
Suse (10000001)
Tobi (10203040)
Tim (11336699)
Anna (12312300)
Tom (11111111)
Tom (12131415)
---------------
Anna (12312300)
Suse (10000001)
Tim (11336699)
Tobi (10203040)
Tom (11111111)
Tom (12131415)
---------------
```