Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. R. Hennicker, Dr. A. Klarl

WS 2016/17
Übungsblatt 13

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung: Lösungsvorschlag

Aufgabe 13-1 **Ballonwettbewerb** Präsenz

In einer Stadt werden mehrere tausend Ballons mit Absenderpostkarten losgelassen. Bis zu einem gewissen Stichtag kommt ein Teil der Postkarten mit Angabe des Fundortes zurück. Sieger ist, wessen Ballon am weitesten geflogen ist (in km), wobei bei gleicher Entfernung mehrere Sieger möglich sind.

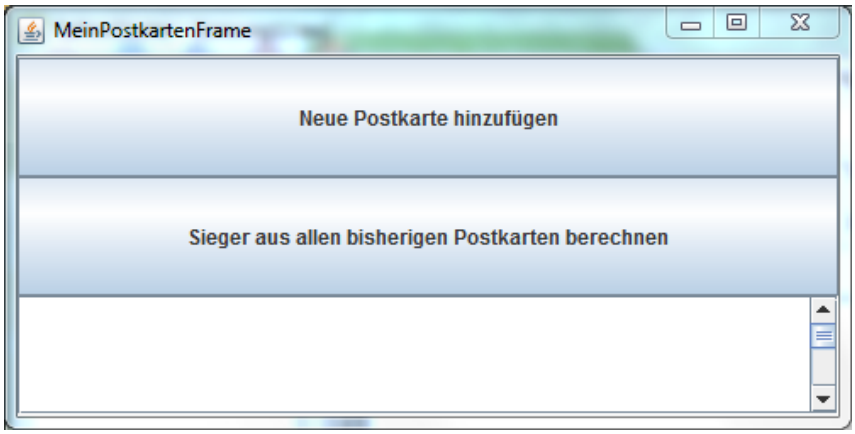
a) Schreiben Sie eine Klasse `Postkarte`, die eine Absenderpostkarte wie oben beschrieben repräsentiert. Die Klasse soll Attribute für den Namen des Teilnehmers und die zurückgelegte Entfernung (als Ganzzahl vom Typ `int`) haben. Fügen Sie einen Konstruktor zur Initialisierung der Attribute sowie "Getter" für beide Attribute hinzu.

Lösung:
siehe Klasse `Postkarte` auf der Webseite

b) Implementieren Sie nun eine verkettete Liste zur Speicherung von Postkarten nach dem Beispiel in der Vorlesung. Nennen Sie die Klasse, die Ihre verkettete Liste darstellt, `MeinePostkartenListe` und die Klasse für Elemente in dieser Liste `MeinPostkartenListenElement`.

Lösung:
siehe Klasse `MeinePostkartenListe` und `MeinPostkartenListenElement` auf der Webseite

c) Die grafische Benutzeroberfläche für die Verwaltung des Ballonwettbewerbs soll wie folgt aussehen:



Es soll zwei Buttons mit der oben genannten Aufschrift geben. Darunter soll ein Ausgabebereich platziert werden.

Schreiben Sie eine Klasse `MeinPostkartenFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `MeinPostkartenFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `MeinPostkartenFrame` abspeichern.

Erweitern Sie Ihre Klasse `MeinPostkartenFrame` um eine Ereignisbehandlung für die beiden Buttons. Wird der Button "Neue Postkarte hinzufügen" gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach dem Namen des Teilnehmers und der zurückgelegten Entfernung gefragt werden. Anschließend soll aus diesen Angaben ein Objekt vom Typ `Postkarte` erstellt werden und in einer Liste vom Typ `MeinePostkartenListe` gespeichert werden. Darüber soll der Benutzer im Ausgabebereich informiert werden.

Wird der Button "Sieger aus allen bisherigen Postkarten berechnen" gedrückt, sollen in der Liste aller bisher eingegebenen Postkarten alle Postkarten gesucht werden, die die weiteste Entfernung erreicht haben. Der Benutzer soll anschließend darüber informiert werden, welche Postkarten die weiteste Entfernung erreicht haben. Beachten Sie, dass dies auch mehrere Postkarten sein können.

Hinweis: In Ihrer Klasse `MeinPostkartenFrame` werden Sie ein Attribut brauchen, um die Liste von Postkarten speichern zu können (Modell der GUI).

Lösung:
Bei der Implementierung der GUI geht man wie gewohnt vor. Da die Ausgabe der siegreichen Postkarten allerdings relativ lang sein kann, können Sie mit Hilfe der Klassen `ScrollPane` und `ScrollPaneConstants` der Swing-Bibliothek dem Ausgabebereich auf folgende Weise eine vertikale Scrollbar hinzufügen.

```
1 JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
2 scrollPane.setHorizontalScrollBarPolicy(
3    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

Vergessen Sie außerdem nicht Ihrer Klasse `MeinPostkartenFrame` ein Attribut vom Typ `MeinePostkartenListe` hinzuzufügen, das alle eingetragenen Postkarten speichern soll und bei der Benutzung des Buttons "Neue Postkarte hinzufügen" um ein Element erweitert wird.

Um alle Postkarten mit der größten Entfernung zu finden, geht man wie folgt vor:

1. Lege eine neue Liste zur Speicherung aller Sieger-Postkarten an.
2. Deklariere eine lokale Variable `groessteEntfernung`, die die maximale Entfernung einer Postkarte speichern soll, und initialisiere sie mit -1.
3. Durchlaufe die Liste von gespeicherten Postkarten und wiederhole für jedes Element:
   - Falls die Postkarte weiter geflogen ist als die aktuell größte Entfernung, lösche die bisherige Siegerliste. Füge die aktuelle Postkarte in die nun leere Siegerliste ein und speichere die Entfernung dieser Postkarte als nun aktuell größte Entfernung in der lokalen Variablen `groessteEntfernung`.
   - Falls die Postkarte genauso weit geflogen ist wie die aktuell größte Entfernung, füge die aktuelle Postkarte in die Siegerliste ein.
4. Gebe die Siegerliste aus.

Die Implementierung der Klassen `MeinPostkartenFrame` und `MeinPostkartenFrameMain` finden Sie auf der Webseite.

d) Zur Lösung der Aufgabe sollen nun die generischen Klassen `LinkedList<E>` und `Iterator<E>` der Java-Bibliothek verwendet werden. Nehmen Sie die Benutzeroberfläche aus Aufgabe c) zur Grundlage und schreiben Sie eine neue Klasse `PostkartenFrame`. Das Aussehen der Benutzeroberfläche sowie die Funktionsweise der Buttons soll nicht verändert werden, allerdings soll anstelle eine Liste vom Typ `MeinePostkartenListe` nun eine Liste vom Typ `LinkedList<Postkarte>` verwendet werden, um alle Postkarten zu speichern. Ebenso soll zum Durchlaufen einer Liste stets ein Objekt der Klasse `Iterator<Postkarte>` benutzt werden. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `PostkartenFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse

`PostkartenFrame` abspeichern.

Lösung:
Die Implementierung der Klassen `PostkartenFrame` und `PostkartenFrameMain` finden Sie auf der Webseite.

Aufgabe 13-2 **Rechnungsverwaltung** Hausaufgabe

Eine Firma möchte beliebig viele Rechnungen erfassen, die sie an Kunden ausgestellt hat. Für jede einzelne Rechnung sollen die Rechnungsnummer und der Rechnungsbetrag angegeben werden. Die Firma möchte außerdem ihre besten Kunden herausfinden, indem sie den/die Kunden mit dem höchsten Rechnungsbetrag ermittelt.

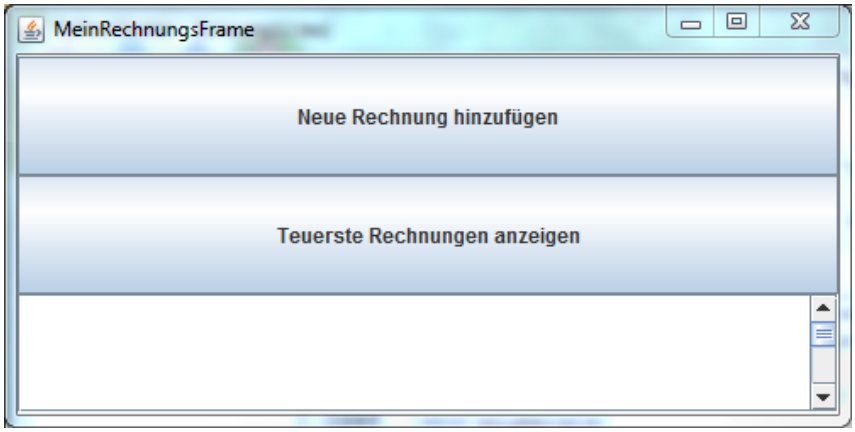
a) Schreiben Sie eine Klasse `Rechnung`, die eine Rechnung wie oben beschrieben repräsentiert. Die Klasse soll Attribute für die Rechnungsnummer und den Rechnungsbetrag (als Ganzzahl vom Typ `int`) haben. Fügen Sie einen Konstruktor zur Initialisierung der Attribute sowie "Getter" für beide Attribute hinzu.

Lösung:
siehe Klasse `Rechnung` auf der Webseite

b) Implementieren Sie nun eine verkettete Liste zur Speicherung von Rechnungen nach dem Beispiel in der Vorlesung. Nennen Sie die Klasse, die Ihre verkettete Liste darstellt, `MeineRechnungsListe` und die Klasse für Elemente in dieser Liste `MeinRechnungsListenElement`.

Lösung:
siehe Klasse `MeineRechnungsListe` und `MeinRechnungsListenElement` auf der Webseite

c) Die grafische Benutzeroberfläche für die Rechnungsverwaltung soll wie folgt aussehen:



Es soll zwei Buttons mit der oben genannten Aufschrift geben. Darunter soll ein Ausgabebereich platziert werden.

Schreiben Sie eine Klasse `MeinRechnungsFrame`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `MeinRechnungsFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `MeinRechnungsFrame` abspeichern.

Erweitern Sie Ihre Klasse `MeinRechnungsFrame` um eine Ereignisbehandlung für die beiden Buttons. Wird der Button "Neue Rechnung hinzufügen" gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach der Rechnungsnummer und dem Rechnungsbetrag

gefragt werden. Anschließend soll aus diesen Angaben ein Objekt vom Typ `Rechnung` erstellt werden und in einer Liste vom Typ `MeineRechnungsListe` gespeichert werden. Darüber soll der Benutzer im Ausgabebereich informiert werden.

Wird der Button "Teuerste Rechnungen anzeigen" gedrückt, sollen in der Liste aller bisher eingegebenen Rechnungen alle Rechnungen gesucht werden, die den höchsten Rechnungsbetrag aufweisen. Der Benutzer soll anschließend darüber informiert werden, welche Rechnungen den größten Betrag aufweisen. Beachten Sie, dass dies auch mehrere Rechnungen sein können.

Hinweis: In Ihrer Klasse `MeinRechnungsFrame` werden Sie ein Attribut brauchen, um die Liste von Rechnungen speichern zu können (Modell der GUI).

Lösung:
Bei der Implementierung der GUI geht man wie gewohnt vor. Da die Ausgabe der teuersten Rechnungen allerdings relativ lang sein kann, können Sie mit Hilfe der Klassen `ScrollPane` und `ScrollPaneConstants` der Swing-Bibliothek dem Ausgabebereich auf folgende Weise eine vertikale Scrollbar hinzufügen.

```
1 JScrollPane scrollPane = new JScrollPane(this.ausgabeBereich);
2 scrollPane.setHorizontalScrollBarPolicy(
3    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
```

Vergessen Sie außerdem nicht Ihrer Klasse `MeinRechnungsFrame` ein Attribut vom Typ `MeineRechnungsListe` hinzuzufügen, das alle eingetragenen Rechnungen speichern soll und bei der Benutzung des Buttons "Neue Rechnung hinzufügen" um ein Element erweitert wird.

Um alle Rechnungen mit dem höchsten Rechnungsbetrag zu finden, geht man wie folgt vor:

1. Lege eine neue Liste zur Speicherung aller teuerster Rechnungen an.
2. Deklariere eine lokale Variable `hoechsterBetrag`, die den Rechnungsbetrag der teuersten Rechnung speichern soll, und initialisiere sie mit -1.
3. Durchlaufe die Liste von gespeicherten Rechnungen und wiederhole für jedes Element:
   - Falls die Rechnung teurer ist als die teuerste Rechnung, lösche die bisherige Liste teuerster Rechnungen. Füge die aktuelle Rechnung in die nun leere Liste teuerster Rechnungen ein und speichere den Rechnungsbetrag dieser Rechnung als nun aktuell teuerste Rechnung in der lokalen Variablen `hoechsterBetrag`.
   - Falls die Rechnung genauso teuer ist wie die aktuell teuerste Rechnung, füge die aktuelle Rechnung in die Liste teuerster Rechnungen ein.
4. Gebe die Liste teuerster Rechnungen aus.

Die Implementierung der Klassen `MeinRechnungsFrame` und `MeinRechnungsFrameMain` finden Sie auf der Webseite.

d) Zur Lösung der Aufgabe sollen nun die generischen Klassen `LinkedList<E>` und `Iterator<E>` der Java-Bibliothek verwendet werden. Nehmen Sie die Benutzeroberfläche aus Aufgabe c) zur Grundlage und schreiben Sie eine neue Klasse `RechnungsFrame`. Das Aussehen der Benutzeroberfläche sowie die Funktionsweise der Buttons soll nicht verändert werden, allerdings soll anstelle eine Liste vom Typ `MeineRechnungsListe` nun eine Liste vom Typ `LinkedList<Rechnung>` verwendet werden, um alle Rechnungen zu speichern. Ebenso soll zum Durchlaufen einer Liste stets ein Objekt vom Typ `Iterator<Rechnung>` benutzt werden. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `RechnungsFrameMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `RechnungsFrame` abspeichern.

Lösung:
Die Implementierung der Klassen `RechnungsFrame` und `RechnungsFrameMain` finden Sie auf der Webseite.

Erinnerung an Aufgabe 12-3: Klausurvorbereitung 6 und 9 ECTS
Überlegen und formulieren Sie Fragen zu Themen aus der Vorlesung, bei denen Sie noch Probleme haben, und geben Sie diese bis Mittwoch, 01.02.2017, 14:00 Uhr über UniworX ab. Die am meisten gestellten Fragen werden in den Tutorien 01.02.2017 bis 08.02.2017 behandelt.

Besprechung der Präsenzaufgaben in den Übungen vom 25.01.2017 bis zum 01.02.2017. Abgabe der Hausaufgaben bis ***Montag****, 06.02.2017, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung). Erstellen Sie zu jeder Aufgabe Klassen, die die Namen tragen, die in der Aufgabe gefordert sind. Geben Sie nur die entsprechenden* `.java`*-Dateien ab. Wir benötigen* ***nicht*** *Ihre* `.class`*-Dateien.*