Softwareentwicklung II (IB)

Blatt 2

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 28.01.2020 17:34

Abgabe der Aufgabe auf diesem Blatt: bis 02.05.18, 08:00 Uhr durch Pushen in das zur Aufgabe gehörende GitHub-Repository.

Aufgabe 1 — PaybackCards revisited

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/dXMeLACY.

Implementieren Sie die beiden Klassen `PaybackCard` und `PaybackCards`.

Sie können sich, analog zu den bisherigen Aufgaben, eine interaktive Klasse `PaybackCardApp` implementieren, mit der Sie `PaybackCards` erzeugen und verändern können. Für die Abgabe ist dies aber nicht notwendig.

`PaybackCard`

Die Klasse `PaybackCard` repräsentiert Payback-Karten. Eine Payback-Karte hat dabei folgende Eigenschaften:

- eine Kundennummer
- der Name des Kunden
- die Anzahl der Bonuspunkte, zu Beginn immer 0
- Standard- oder Premiumstatus (zu Beginn immer Standard)
- eine Markierung (das nennt man in der Programmierung gerne *Flag*) ob die Karte gültig ist

Wählen Sie sinnvolle Bezeichner und Typen. Alle Objektvariablen müssen `private` sein. Die Kundennummer und der Name müssen unveränderbar sein. Bei Namensänderung wird eine neue Karte erstellt und die Punkte übertragen.

Für jede Objektvariable muss es einen Getter geben. Die Bezeichner ergeben sich durch die Verwendung in den bereits im Repository vorhandenen Tests. An den Tests dürfen Sie, wie immer, nichts ändern.

Die Bonuspunkte können mit der Methode `addBonuspoints` erhöht werden. Der als Parameter übergebene Wert wird, sofern er positiv ist, auf den aktuellen Punktestand addiert, Ein negatives Argument wird ignoriert. Wenn die Karte nicht mehr gültig ist, wird auch nichts mehr dazu addiert. Ergebnis der Methode ist der neue Punktestand, wie er auch durch den Getter abgerufen werden könnte.

Um die Karte auf den Premiumstatus zu setzen, kann die Methode `setPremiumCustomer` verwendet werden. Die Rückgabe zeigt an, ob die Karte vorher schon Premiumstatus hatte.

Mit Hilfe der Methode `merge` kann eine andere Karte zu einer Karte hinzugefügt werden. Das funktioniert nur, wenn beide Karten gültig sind. Beim Zusammenführen werden alle Punkte auf die Karte (`this`) gebucht. Wenn eine der beiden bereits Premiumstatus hatte, soll diese Karte nach dem Zusammenfügen Premiumstatus haben. Die andere Karte wird auf ungültig gesetzt. Ergebnis der Methode `merge` ist ein Wahrheitswert, der anzeigt ob etwas gemerged wurde. Achtung: Sie müssen explizit auch darauf achten, dass die andere Karte wirklich eine **andere** Karte ist! `card.merge(card)` soll an der Karte `card` nichts verändern!

Die letzte Methode der Klasse `PaybackCard` ist die redefinierte[1] Methode `toString`. Diese Methode berechnet einen `String`, der sich folgendermaßen zusammensetzt:

- Kundennummer, so mit Leerzeichen vorne aufgefüllt, dass die letzte Ziffer an vierter Stelle steht.
- ein Punkt und ein Leerzeichen
- der Kundenname
- ein Komma und ein Leerzeichen
- wenn der Kunde 0 Punkte hat, gefolgt von `keine Punkte`, sonst Anzahl der Punkte, gefolgt von einem Leerzeichen und dem Wort `Punkte`
- nur wenn der Kunde Premiumstatus besitzt: ein Komma, ein Leerzeichen und das Wort `Premiumkunde`.

[1] Jede Klasse enthält schon automatisch eine *geerbte* (siehe später Kapitel Vererbung) Methode `public String toString()`. Um diese zu redefinieren, Schreiben Sie vor die bisherige Signatur ein `@Override`, also

```
@Override
public String toString() {
  ...
}
```

In den Tests wird die `String`-Repräsentation von vier Payback-Karten überprüft.

Analog zur Methode `System.out.printf`, gibt es auch die Methode `String.format`, die bei den selben Argumenten wie `printf` die entsprechende Zeichenkette berechnet. Verwenden Sie die Methode `String.format` in der Methode `toString`.

PaybackCards

Die Klasse `PaybackCards` dient der Verwaltung mehrere Payback-Karten in einer Liste. Eine Liste ist eine Datenstruktur in der keine oder mehrere gleichartige Objekte verwaltet werden können, ähnlich der Map auf dem letzten Blatt, aber ohne eine Schlüsselkomponente. Die Klasse `PaybackCards` hat also als einzige Objektvariable eine Liste von Payback-Karten. Definieren Sie die Objektvariable folgendermaßen:

```
private final List<PaybackCard> cards = new ArrayList<>();
```

Die Variable `cards` hat also den Typ `List<PaybackCard>`, also Liste von Payback-Karten. Eine konkrete Implementierung einer Liste ist beispielsweise eine `ArrayList`. Wenn wir dabei die spitzen Klammern leer lassen, weiß Java, dass dort genauso Payback-Karten gemeint sind. D.h. ausführlicher wäre:

```
private final List<PaybackCard> cards = new ArrayList<PaybackCard>();
```

Die Klasse `PaybackCards` enthält die folgenden drei Methoden:

```
boolean add(final PaybackCard card)
PaybackCard findByCustomerId(final int customerId)
int removeInvalidCards()
```

Mit der Methode `add` kann eine Payback-Karte zur `cards`-Liste hinzugefügt werden. Dies soll aber nur dann erfolgreich sein, wenn noch keine Karte mit der selben Kundennummer in der Liste vorhanden ist. Dann ist das Ergebnis der Methode `add` `true`. Andernfalls soll die Karte nicht hinzugefügt werden und `false` zurück gegeben werden. Die bereits in der Liste vorhanden Karten können Sie dazu, analog zu den Werten in der Map vom letzten Blatt, mit einer foreach-Schleife durchgehen:

```
for (PaybackCard c: cards) {
  ...
}
```

Die Methode um in einer Liste etwas hinzuzufügen, heißt übrigens auch `add`.

Mit der Methode `findByCustomerId` kann die Karte aus der Liste gesucht werden, die die übergebene Kundennummer hat. Wird die Karte (mit einer foreach-Schleife) gefunden, soll diese als Ergebnis zurück gegeben werden. Wird keine Karte mit der Kundennummer gefunden, soll das Ergebnis `null` sein. Nachdem Karte nur mit `add` hinzugefügt werden können, ist in diesem Design klar, dass es keine zwei Karten mit der Kundennummer in der Liste geben kann.

Die Methode `removeInvalidCards` soll alle Karten, die ungültig geworden sind aus der Liste entfernen. Damit können anschließend die vergebenen Kundennummern wieder neu benutzt werden. Ergebnis der Methode `removeInvalidCards` ist die Anzahl der entfernten Karten. Sie können die Liste in diesem Fall nicht einfach durchgehen und die ungültigen gleich entfernen, da sich die Liste dann während der foreach-Schleife ändern würde und Java eine sog. Exception erzeugt.

Die Idee ist vielmehr innerhalb der Methode eine neue Liste zu erzeugen, in die alle ungültigen Karten (innerhalb der foreach-Schleife) eingefügt werden. Nach der Schleife können Sie dann mit der Listen-Methode `removeAll` alle diese Karten aus der `cards`-Liste entfernen. Um die Anzahl der entfernten Karten zurück geben zu können, müssen Sie nur die neue Liste nach ihrer Größe fragen (Methode `size`).

Aufgabe 2 — Punkte im dreidimensionalen Raum

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/z5r2gC4e.

Die Herausforderung bei dieser Aufgabe ist nicht der Java-Code (den sollten Sie sehr einfach hinbekommen) sondern das fehlende Gradle-Projekt. Im Repository finden Sie nur drei Dateien:

- `.gitignore` — gibt an, welche Dateien von Git ignoriert werden sollen
- `Jenkinsfile` — die Befehle für den Jenkins. Diese Datei sollen Sie nicht verändern.
- `src/test/java/PointTest.java` — die Unittests, die Sie auch nicht verändern sollen

Erstellen Sie ein Gradle-Projekt und implementieren Sie darin eine Klasse `Point` für Punkte im dreidimensionalen Raum.

- Ein Punkt besteht aus drei privaten Koordinaten `xcoord`, `ycoord` und `zcoord`.
- Implementieren Sie einen Default-Konstruktor der den Ursprung repräsentiert und einen Custom-Konstruktor der drei `int` für die drei Koordinaten als Parameter hat.
- Für jede Dimension gibt es eine Methode die den Punkt um den übergebenen Parameter in die jeweilige Dimension verschiebt. Die Methoden heissen `movex`, `movey` und `movez`.
- Für jeweils zwei Dimensionen gibt es die entsprechenden "Verschiebemethoden" `movexy`, `movexz` und `moveyz`. Diese Methoden dürfen keine Änderungen an den Objektvariablen unmittelbar vornehmen, sondern müssen die eindimensionalen Verschiebemethoden nutzen.
- Es gibt eine dreidimensionale Verschiebemethode `movexyz` die genau zwei Methodenaufrufe enthält.

- Die Methode `scale` vergrößert den Abstand des Punktes vom Ursprung in jeder Dimension um den übergebenen Faktor.
- Die Methode `toString` gibt den Punkt als Zahlentripel **ohne** Zeilenumbruch und **ohne** Leerzeichen zurück. Der Punkt mit den Koordinaten x=5, y=7 und z=12 wird also zurück gegeben als

  ```
  "(5,7,12)"
  ```

  Achtung: Wenn die `toString`-Methode nicht korrekt ist, schlagen alle JUnit-Tests fehl!

Probleme auf Jenkins?

Wenn es bei Ihnen lokal (unter Windows) läuft, aber Jenkins Ihnen den Fehler anzeigt

```
./gradlew : Permission denied
```

liegt es daran, dass die Datei `gradlew` unter Unix nicht als ausführbar markiert wurde. Sie können Sie mit Hilfe von Git folgendermaßen als ausführbar markieren:

```
git update-index --chmod=+x gradlew
```

und anschließend committen und pushen.