Projektblatt P6 (24 P)

Abgabe: Freitag 27. Oktober 2023, 12:00h
Entpacken Sie zunächst die Archiv-Datei `vorgaben-p06.zip`, in der sich die Rahmendateien für die zu lösenden Aufgaben befinden. Ergänzen Sie die Dateien durch Ihre Lösungen gemäß der Aufgabenstellung unten bzw. fügen Sie in einigen Fällen neue Klassen (ggf. in neuen Dateien) hinzu. 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.P06.zip`, welches Sie auf Ilias hochladen.
Führen Sie dazu in dem Verzeichnis, in dem Sie die Dateien bearbeitet haben, folgenden Befehl auf der Kommandozeile aus:

```
zip  IhrNachname.IhrVorname.P06.zip *.java
```

Aufgabe 1: Objektorientierter Entwurf 12 (= 2+2+3+5) Punkte

In dieser Aufgabe sollen Sie ein sehr einfaches Würfelspiel realisieren. Zwei Spieler würfeln abwechselnd mit einer vorgegebenen Anzahl von Würfeln. Die Würfelpunkte werden den Spielern dabei jeweils als Punkte gutgeschrieben. Nach einer festgelegten Anzahl von Runden gewinnt der Spieler, der mehr Punkte hat.
In der Datei `WuerfelSpiel` sind hierzu folgende Klassen (teilweise) vorgegeben:

- Die Klasse `Wuerfel` repräsentiert einen einzelnen Würfel, der durch zwei Merkmale beschrieben werden kann: einen Wert (die Augenzahl) und eine Nummer. Vorgegeben sind neben den beiden Instanzvariablen `wert` und `nr` eine Klassenvariable `count`, mit deren Hilfe Würfel automatisch nummeriert werden können.

- Die Klasse `Punkte` repräsentiert die Punkte, die ein Spieler hat. Hierzu ist in der Klasse die Instanzvariable `anzahl` vorgegeben.

- Die Klasse `Spieler` repräsentiert einen einzelnen Spieler mit einem (eindeutigen) Namen und einer Punktezahl. Um automatisch eindeutige Namen zu generieren, ist eine Klassenvariable `count` vorgegeben.

- Die Klasse `WuerfelSpiel` repräsentiert das eigentliche Spiel. In der Klasse sind zunächst zwei Konstanten vorgegeben, die die Anzahl der Runden und die der Würfel definieren. Darüberhinaus sind drei Instanzvariablen vorgegeben, die ein Feld für die Würfel sowie zwei Spieler definieren.

  Mit Hilfe der vorgegebenen Methode `ausgabeErgebnis` kann später das Ergebnis des Spiels ausgegeben werden. Der Code in der Methode ist noch auskommentiert, da Sie noch einige der darin aufgerufenen Methoden schreiben müssen.

  Die noch leere Methode `spielen` führt das Spiel aus und ist später noch von Ihnen zu implementieren.

Ergänzen Sie die Datei `WuerfelSpiel.java` nun wie im Folgenden beschrieben:

(a) Ergänzen Sie die Klasse `Wuerfel` wie folgt:

- Schreiben Sie einen Konstruktor, der die Instanzvariable `nr` mit einem eindeutigen Wert initialisiert. Die erste Würfelinstanz soll dabei die Nummer 1 erhalten, jede weiter dann einen um 1 größeren Wert. Nutzen Sie hierzu die Klassenvariable `count`.
- Schreiben Sie eine öffentliche Getter-Methode, die den Wert der Instanzvariablen `wert` zurückgibt.
- Schreiben Sie eine öffentliche Methode mit dem Namen `werfen`, die der Instanzvariable `wert` einen Zufallswert aus dem Intervall $[1, 6]$ zuweist. Dies soll den Würfelwurf simulieren.
- Überschreiben Sie die Methode `toString`, so dass ein `String` zurückgegeben wird, der eine String-Repräsentation eines Würfels darstellt und welcher wie im folgenden Beispiel (für einen Würfel mit der Nummer 1 und dem Wert 4) aussieht. Der String soll mit einem Leerzeichen beginnen.

```
W1 = 4
```

(b) Ergänzen Sie die Klasse `Punkte` wie folgt:

- Schreiben Sie eine öffentliche Getter-Methode, die den Wert der Instanzvariablen `anzahl` zurückgibt.
- Schreiben Sie eine öffentliche Methode mit dem Namen `update`, die einen gannzahligen Parameter hat, dessen Wert zu der aktuellen Punktezahl hinzuaddiert wird. Falls die Anzahl der Punkte anschließend kleiner als 0 ist, setzen Sie den Wert auf 0 (es sollten keine negativen Punktzahlen möglich sein).
- Überschreiben Sie die Methode `toString`, so dass ein `String` zurückgegeben wird, der eine String-Repräsentation des Punktestands darstellt, welche wie im folgenden Beispiel aussieht.

```
42 Punkte
```

(c) Ergänzen Sie die Klasse `Spieler` wie folgt:

- Schreiben Sie einen Konstruktor, der den Namen des Spielers mit Hilfe der Klassenvariablen `count` fortlaufend wie folgt benennt: Der erste Spieler soll den Namen "Spieler-1" erhalten, der zweite Spieler den Namen "Spieler-2", usw. Initialisieren Sie auch die Instanzvariable `punkte` mit einem neuen Objekt der Klasse `Punkte`.
- Schreiben Sie eine Getter-Methode für das Attribut `name`.
- Schreiben Sie eine parameterlose Methode mit dem Namen `getErgebnis`, die einen String zurückgibt, der den aktuellen Punktestand des Spielers beschreibt und so aussieht wie im folgenden Beispiel:

  ```
  Spieler-1 hat 47 Punkte
  ```

- Schreiben Sie eine Methode `compareTo`, die einen Parameter vom Typ `Spieler` hat und einen `int`-Wert zurückgibt. Dieser Wert soll negativ (bzw. positiv) sein, wenn der Spieler, für den die Methode aufgerufen wird, weniger (bzw. mehr) Punkte hat als der Spieler, der als Parameter übergeben wird.
- Schreiben Sie eine Methode mit dem Namen `wuerfeln`, der ein Feld von Würfeln übergeben wird. In der Methode sollen alle Würfel einmal geworfen werden. Die Würfelpunkte jedes Würfels sollen dann anschließend mit Hilfe der Methode `update` auf die Punkte des Spielers angerechnet werden.

(d) Ergänzen Sie die Klasse `WuerfelSpiel` wie folgt:

- Schreiben Sie einen Konstruktor, der die beiden Spieler `sp1` und `sp2` mit zwei Instanzen der Klasse `Spieler` initialisiert, sowie das Feld `wuerfel` anlegt und mit der entsprechenden Anzahl von Würfel-Instanzen füllt.
- Schreiben Sie eine Methode mit dem Namen `ausgabeWurf`, die als Parameter ein Feld von Würfeln besitzt und einen String generiert und zurückgibt, der die Werte der einzelnen Würfel enthält. Der String sollte dabei wie in folgendem Beispiel aussehen:

  ```
  W1 = 4 W2 = 2 W3 = 6
  ```

  Verwenden Sie hierbei die `toString`-Methode aus der Klasse `Wuerfel`.

- Implementieren Sie die Methode `spielen` wie folgt: Es soll `anzahlRunden` Mal der erste Spieler und anschließend der zweite Spieler (mit allen Würfeln) würfeln und dann der jeweilige Wurf ausgegeben werden. Die Ausgabe sollte dann wie in folgendem Beispiel aussehen:

  ```
  Wurf Spieler 1: W1 = 4 W2 = 2 W3 = 6
  Wurf Spieler 2: W1 = 1 W2 = 6 W3 = 2
  ```

Um Ihre Implementierung zu testen, entfernen Sie die Blockkommentarzeichen in der Methode `ausgabeErgebnis` und führen Sie dann die Klasse `WuerfelSpiel` aus.
Die Ausgabe bei einem Programmlauf sollte dann wie in folgendem Beispiel aussehen:

```
Wurf Spieler 1: W1 = 4 W2 = 2 W3 = 6
Wurf Spieler 2: W1 = 1 W2 = 6 W3 = 2
--------------------------------------
Wurf Spieler 1: W1 = 6 W2 = 2 W3 = 5
Wurf Spieler 2: W1 = 2 W2 = 2 W3 = 2
--------------------------------------
Wurf Spieler 1: W1 = 1 W2 = 2 W3 = 1
Wurf Spieler 2: W1 = 5 W2 = 1 W3 = 4
--------------------------------------
Wurf Spieler 1: W1 = 1 W2 = 4 W3 = 1
Wurf Spieler 2: W1 = 6 W2 = 2 W3 = 2
--------------------------------------
Wurf Spieler 1: W1 = 6 W2 = 2 W3 = 4
Wurf Spieler 2: W1 = 1 W2 = 2 W3 = 2
--------------------------------------
Spieler-1 hat 47 Punkte
Spieler-2 hat 40 Punkte
Spieler 1 hat gewonnen
```

Aufgabe 2: Listen, Vererbung 12 (= 5+4+3) Punkte

In dieser Aufgabe sollen Sie verschiedene Listenklassen implementieren, die Ihre Elemente jeweils unterschiedlich speichern. Hierzu müssen sie jeweils eine neue Klasse anlegen, die direkt oder indirekt von der vorgegebenen Klasse `Liste` abgeleitet ist.
Folgende Klassen sind in den drei Dateien `TestData.java`, `Teilnehmer.java` und `Liste.java` vorgegeben:

- Die Klasse `TestData` beinhaltet ein zweidimensionales Feld von Strings. Jede "Zeile " in diesem Feld definiert einen Teilnehmer in einem 10km Lauf (mit Ausnahme der Startnummer). Die Klassenmethode `getTeilnehmerFeld` generiert daraus ein Feld mit Instanzen der Klasse `Teilnehmer`.

- Die Klasse `Teilnehmer` definiert einen Teilnehmer eines 10km Laufs. Die Teilnehmer besitzen eine eindeutige Startnummer sowie einen (evtl. nicht eindeutigen) Namen und ein Attribut für die gelaufene Zeit (in Sekunden). Die Klasse besitzt neben einem Konstruktor eine Methode, die eine String-Repräsentation des Teilnehmers generiert und zurückgibt. Diese besteht aus dem Namen, der Startnummer, dem Jahrgang und der Laufzeit (formatiert auf Minuten und Sekunden). Der Konstruktor generiert dabei mit Hilfe der Klassenvariable `count` fortlaufende Startnummern für die Teilnehmer. Die Klasse besitzt eine (parameterlose) Methode mit dem Namen `getKey`, die als Schlüsselwert das Geburtsjahr des Teilnehmers (Attribut `jahrgang`) zurückgibt.

- Die Klasse `Listenelement` definiert ein Element einer Liste, dessen Attribute einen Teilnehmer und das jeweils nächste Listenelement referenzieren.

- Die Klasse `Liste` definiert eine Datenstruktur für eine verkettete Liste von Teilnehmern. Neben einem Konstruktor und einer Methode für die Ausgabe der Liste ist eine Methode `insert` vorgegeben, die einen neuen Teilnehmer am Kopf der Liste einfügt.

- Die Klasse `TestListen` besitzt eine `main`-Methode, in der ein Feld von 50 Teilnehmern vorgegeben ist (generiert aus den Daten in der Klasse `TestData`)

(a) Entwickeln Sie nun zunächst eine Listenklasse, die neue Teilnehmer immer sortiert (nach den Laufzeiten) einfügt:

Ergänzen Sie hierzu die Klasse `Teilnehmer` durch eine Methode mit dem Namen `compareTo`, die die Laufzeiten der aktuellen Instanz (also der durch `this` refererenzierten Instanz) mit der eines als Parameter übergebenen Teilnehmers vergleicht. Ist der Teilnehmer, der die aktuelle Instanz repräsentiert, schneller gewesen als der andere Teilnehmer (= kürzere Zeit), soll ein negativer Wert zurückgegeben werden. Ist der Teilnehmer langsamer gewesen, soll ein positiver Wert zurückgegeben werden. Sind die Laufzeiten gleich, soll der Wert 0 zurückgegeben werden.

Entwickeln Sie nun eine neue Klasse `SortierteListe` (in einer eigenen Datei), die Sie von der Klasse `Liste` ableiten. Überschreiben Sie dort die Methode `insert`, so dass neue Teilnehmer aufsteigend nach der Laufzeit sortiert eingefügt werden.

(b) Entwickeln Sie nun eine Klasse, die die Teilnehmer in verschiedenen Listen speichert, die wiederum in einem Feld gesammelt werden. Die Teilnehmer werden dann mit Hilfe eines Schlüsselwerts einer einzelnen Liste zugewiesen.

Legen Sie eine neue Klasse mit dem Namen `ListenFeld` in einer neuen Datei an, die von der Klasse `SortierteListe` abgeleitet ist und statten Sie sie wie folgt aus:

- Deklarieren Sie eine ganzzahlige Konstante `N=5` in der Klasse (mit der Anweisung: `private static int N = 5;`) sowie ein Feld der Länge N mit dem Komponententyp `Liste`.
- Schreiben sie einen Konstruktor, der dieses Feld mit N leeren Listen (Instanzen der Klasse `SortierteListe`) initialisiert.
- Überschreiben Sie die Methode `print` so, dass die einzelnen, in dem Feld gespeicherten Teilnehmerlisten nacheinander ausgegeben werden (zuerst die Teilnehmer in der Liste, die in dem ersten Feldelement gespeichert ist, dann die Teilnehmer aus der Liste, die im zweiten Feldelement gespeichert ist, usw.). Nutzen Sie dabei die in der Oberklasse definierte Methode, um jede Teilliste auszugeben. Trennen Sie die einzelnen Teillisten in der Ausgabe dabei jeweils durch eine Leerzeile.

- Überschreiben Sie die Methode `insert` so, dass die Teilnehmer auf die in dem Feld referenzierten Teillisten verteilt werden. Holen Sie sich dazu zunächst den Schlüsselwert aus dem einzufügenden `Teilnehmer`-Objekt. Bilden Sie den Wert so auf einen Indexwert ab, dass je zwei aufeinanderfolgende Jahrgänge den gleichen Indexwert erhalten (und damit später in einer Teilliste landen) und dann gleichmäßig auf die N Teillisten verteilt werden. Sie dürfen davon ausgehen jeder Jahrgang j im Intervall $[2010, 2019]$ liegt (daher werden auch $N = 5$ Teillisten benötigt).

(c) Ergänzen Sie nun die `main`-Methode der Klasse `TestListen` wie folgt:

- Definieren sie ein Feld mit drei Komponenten vom Typ `Liste`. Initialisieren Sie den Inhalt des Felds dann mit je einer neuen Instanz der drei Klassen `Liste`, `SortierteListe` und `ListenFeld`.
- Füllen Sie alle Listen jeweils mit den vorgegebenen Teilnehmern.
- Geben Sie die drei Listen am Ende mit Hilfe der `print`-Methode aus.

Die Ausgabe für den vorgegebenen Datensatz finden Sie in der beigefügten Textdatei `Ausgabe.txt`.