Universität Leipzig
Institut für Informatik

WS 19/20

Modellierung und Programmierung 1 – Übungsserie 4

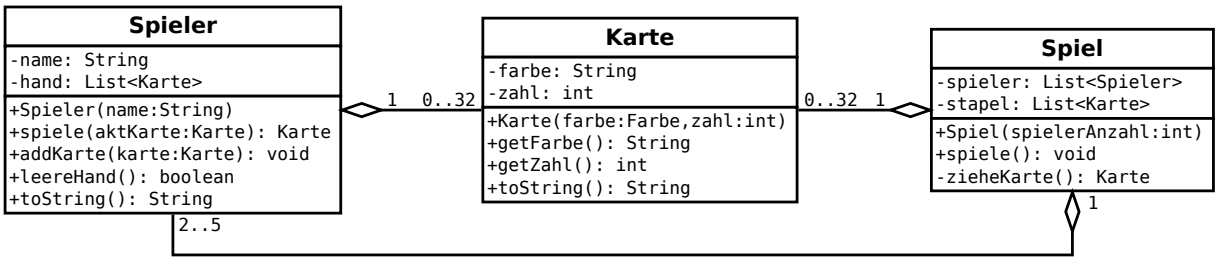
Abgabetermin: 18.12.2019, 23:59 Uhr
Abgabeformat: 1 ZIP-Datei
Max. Punkte: 42

Java Programmierung - Collections, Interface-Nutzung, Strings

1. Einfaches Kartenspiel simulieren *(17 Punkte)*

   Ziel dieser Aufgabe ist die Simulation eines einfachen Kartenspiels. Das Spiel hat folgende Regeln:

   - Das Spielblatt besteht aus vier Farben (Rot, Schwarz, Grün und Blau) mit je 8 Karten mit den Ziffern 1 bis 8. Insgesamt gibt es also 32 Karten.
   - Es können maximal 5 Spieler mitspielen.
   - Jeder Spieler erhält zu Beginnn 5 zufällige Karten.
   - Eine Karte wird zufällig als Startkarte gezogen.
   - Nacheinander versucht jeder Spieler eine seiner Karten, welche zur gerade liegenden Karte passt, auszuspielen. Eine Karte ist passend, wenn Farbe oder Zahl mit der liegenden Karte übereinstimmen. Kann keine Karte gelegt werden, muss der Spieler eine zufällige Karte aus dem Stapel der verbliebenen Karten ziehen.
   - Der Spieler welcher zuerst alle Karten ausgespielt hat, gewinnt das Spiel.
   - Beispiel für 3 Spieler: Karten wurden ausgeteilt, als Startkarte liegt Rot 3
     - Spieler 1 legt Rot 7
     - Spieler 2 legt Rot 4
     - Spieler 3 legt Grün 4
     - Spieler 1 legt Grün 2
     - Spieler 2 hat keine passende Karte und muss eine zufällige Karte aus dem Stapel ziehen
     - Spieler 3 legt Grün 8
     - usw.
   - Alle abgelegten Karten, außer die gerade Gelegte, kommen wieder in den Stapel der verbliebenen Karten. Diese können also von den Spielern wieder gezogen werden.



   Schreiben Sie die Java-Klassen `Spiel`, `Spieler` und `Karte` entsprechend dem UML-Klassendiagramm. Implementieren Sie die Methoden und Konstruktoren sinnvoll. Beachten Sie dabei aber folgendes:

a) Klasse `Karte` *(2 Punkte)*

- `toString` Liefert Farbe und Zahl als String zurück, also z.B. „Rot 7“.
- Optional kann für das Attribut `farbe` ein `enum` genutzt werden.

b) Klasse `Spieler` *(6 Punkte)*

- `spiele` Findet für die übergebene Karte eine Passende in der Spielhand und gibt diese zurück. Falls keine passende Karte gefunden wurde, soll `null` zurückgegeben werden.
- `addKarte` Fügt die übergebene Karte der Hand des Spielers hinzu.
- `leereHand` Liefert `true` zurück, wenn der Spieler keine Karten mehr hat.
- `toString` Gibt den Namen des Spielers zurück.

c) Klasse `Spiel` *(8 Punkte)*

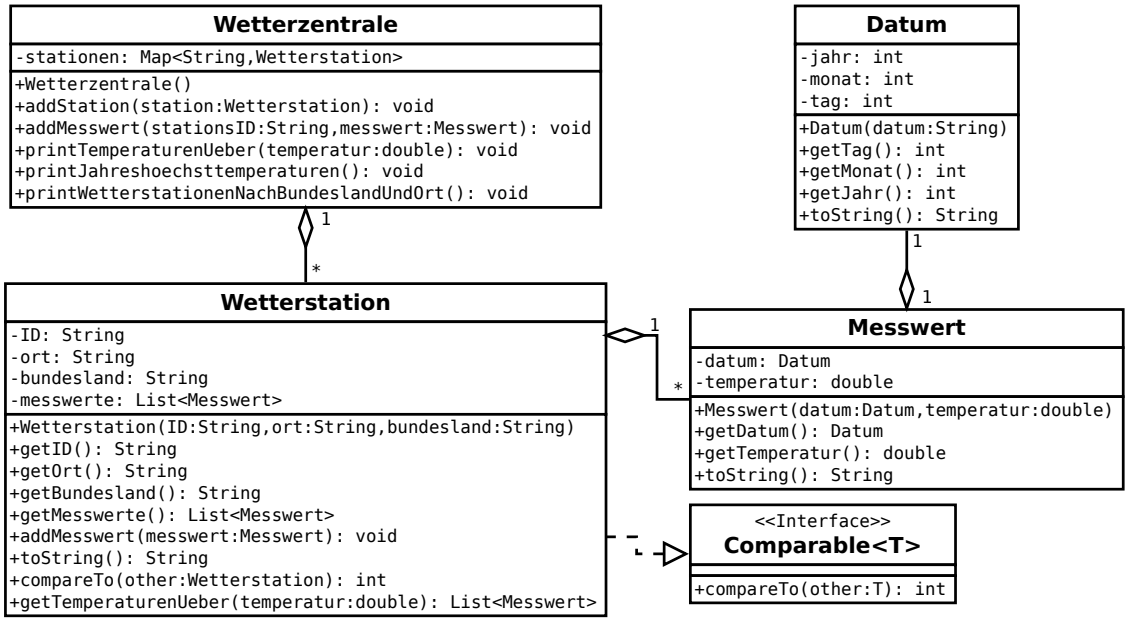
- Konstruktor erstellt die Karten, die Spieler und verteilt die Karten an die Spieler.
- `spiele` Führt das oben beschriebene Spiel durch, bis ein Spieler gewinnt. Dabei sollen Ausgaben der Form „Spieler 1 spielt Schwarz 1“, „Spieler 2 muss ziehen“ und „Spieler 1 gewinnt“ erfolgen.
- `zieheKarte` Entnimmt dem Stapel eine zufällige Karte und gibt diese Karte zurück.

d) Klasse `Main` *(1 Punkt)*

In der Klasse `Main` wird die Methode `public static void main( String[] args)` erstellt. Diese erstellt eine Instanz der Klasse `Spiel` mit einer beliebigen zulässigen Spielerzahl und startet die Spielsimulation.

2. Wetterdaten *(25 Punkte)*

In dieser Aufgabe werden Daten zu Wetterstationen eingelesen und verarbeitet.



Programmieren Sie die Java-Klassen `Datum`, `Messwert`, `Wetterstation`, `Wetterzentrale` und `Main`. Halten Sie sich dabei an die Vorgaben durch das UML-Klassendiagramm. Beachten Sie aber folgende Hinweise zur Implementierung bestimmter Methoden:

a) Klasse `Datum` *(3 Punkte)*

- Konstruktor erhält ein Datum als `String` in der Form „JJJJ-MM-TT“, z.B. „2019-05-12“, und setzt die Attribute entsprechend.
- `toString` gibt das gespeicherte Datum in der obigen Form zurück. Beachten Sie dabei die führenden Nullen.

b) Klasse `Messwert` *(1 Punkt)* Implementieren Sie die Klasse `Messwert` wie vorgegeben.

- `toString` gibt die in der Klasse gespeicherten Attribute als String zurück, z.B. in folgender Form: „2002-07-02 Temperatur: 40.5°C“

c) Klasse `Wetterstation` *(7 Punkte)*

- `toString` gibt die in der Klasse gespeicherten Attribute bis auf die Meßwerte als String zurück, z.B. in folgender Form: „ID: 2 Erfurt-Weimar (Thüringen)“
- `getTemperaturenUeber` Gibt eine Liste mit allen Messwerten über den als Parameter gegebenen Temperaturwert `temperatur` zurück.
- `compareTo` liefert den Rückgabewert entsprechend der Vorgabe durch das Interface `Comparable<T>` so, dass die Wetterstation zuerst anhand des Attribut `bundesland` und in zweiter Reihe nach dem Attribut `ort` sortiert wird.

d) Klasse `Wetterzentrale` *(12 Punkte)*

- `addStation` fügt die gegebene Wetterstation mit ihrer ID als Schlüssel in die Map `stationen` ein.
- `addMesswert` fügt den übergebenen Messwert der Station mit der gegebenen ID `stationsID` hinzu.
- `printTemperaturenUeber` Gibt die Messwerte aller Wetterstationen mit Temperaturen über der als Parameter gegebenen Temperatur aus. Zum Beispiel:

```
Messwerte über 28.0°C für ID: 0 Leipzig/Halle (Sachsen)
2006-07-01 Temperatur: 29.91°C
2018-07-01 Temperatur: 28.22°C
Messwerte über 28.0°C für ID: 1 München-Flughafen (Bayern)
2003-08-01 Temperatur: 29.45°C
2006-07-01 Temperatur: 28.3°C
2015-08-01 Temperatur: 28.21°C
```

- `printJahreshoechsttemperaturen` Bestimmt über alle Messwerte aller Wetterstationen die Jahreshöchsttemperaturen und gibt diese aus. Zum Beispiel:

```
Jahreshöchsttemperaturen nach Jahren
2001: 26.280000686645508°C
2002: 26.030000686645508°C
2003: 29.450000762939453°C
```

- `printWetterstationenNachBundeslandUndOrt` Gibt die Wetterstationen alphabetisch sortiert nach Bundesland und Ort aus, d.h. die erste Sortierordnung ergibt sich durch die Bundesländer und die Zweite durch die Orte. Zum Beispiel:

```
Wetterstationen sortiert nach Bundesland und Ort
ID: 1 München-Flughafen (Bayern)
ID: 0 Leipzig/Halle (Sachsen)
ID: 2 Erfurt-Weimar (Thüringen)
ID: 3 Jena (Sternwarte) (Thüringen)
```

e) Klasse `Main` *(2 Punkte)* In der Klasse `Main` wird die Methode `public static void main( String[] args)` erstellt. Diese erzeugt eine Instanz der Klasse `Wetterzentrale` und liest die Daten aus den Methoden `getStationen` und `getTemperaturen` der gegebenen Klasse `Wetterdaten` in die erzeugte Instanz der Klasse `Wetterzentrale` ein.

Danach werden folgende Methoden der Instanz der Klasse `Wetterzentrale` aufgerufen:

- `printTemperaturenUeber` mit 28°C als Parameter
- `printJahreshoechsttemperaturen`
- `printWetterstationenNachBundeslandUndOrt`