1 Allgemeine Aufgabestellung

In dieser Übung sollen Sie den Umgang mit Schnittstellen (interfaces) und abstrakten Klassen (abstract classes) vertiefen. Die Aufgabe besteht darin, ein abstraktes Modell eines Kartenspiels zu modellieren und für die Implementierung eines konkreten Kartenspiels zu verwenden.

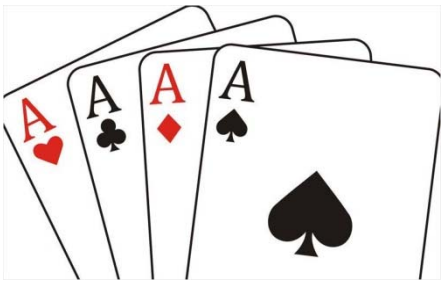
Kartenspiele allgemein

Für diese Übung nehmen wir an, dass ein Kartenspiel im allgemeinen aus einem Stapel Karten besteht, der gemischt, sortiert, ausgeteilt und wieder zusammengefügt werden kann. Jede Spielkarte im Stapel ist durch ihre Farbe und ihren Wert bestimmt.

Skat

Skat ist ein populäres deutsches Kartenspiel für drei Spieler das im thüringischen Altenburg entwickelt wurde. Skat wird sportlich organisiert betrieben, z.B. in einer Bundesliga und sogar mit Welt- und Europameisterschaften. Ähnlich Poker wird Skat auch online auf diversen Casino Portalen angeboten.

Ein Skatblatt besteht aus 32 Karten von denen jeder Spieler 10 Karten vor Spielbeginn erhält. Die 2 übrigen Karten werden als „Skat“ bezeichnet und zurückgehalten. Jede Skat-Karte ist durch ihre Farbe und Wert eindeutig bestimmt, d.h. es gibt keine Duplikate.



Das Skatblatt beinhaltet die folgenden vier Farben:

- Karo
- Herz
- Pik
- Kreuz.

Dabei ist Karo die niederwertigste und Kreuz die höchstwertigste Farbe.

Für jede Farbe gibt es acht Karten mit den folgenden Werten:

- 7
- 8
- 9
- 10
- Bube
- Dame
- Koenig
- Ass.

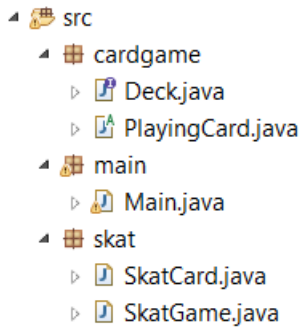
Für dieses Übungsblatt nehmen wir an, dass die Wertigkeit der Karten der oben genannten Reihenfolge entspricht. D.h. 7 ist der niedrigste Wert und das Ass besitzt den höchsten Wert innerhalb einer Farbe.

Bitte achten Sie darauf, dass in ihrem Programm Umlaute weder für Bezeichner noch in Zeichenketten verwendet werden.

2 Erstellen der Projektstruktur

Erstellen und Implementieren Sie die folgenden Packages, Klassen und Interfaces.

- Packages main, cardgame und skat
- Klasse main.Main implementiert die main()-Methode
- Interface cardgame.Deck und Klasse cardgame.PlayingCard
- Klassen skat.SkatCard und skat.SkatGame



3 Abstrakte Klasse PlayingCard (3 Punkte + 2 Zusatzpunkte)

Erstellen Sie die abstrakte Klasse ***PlayingCard*** (Spielkarte) anhand der folgenden Beschreibung.

Die Klasse soll als öffentlich zugängliche sowie als abstrakte Klasse implementiert werden, da sie sehr allgemein ist und die Implementierung einiger Methoden evtl. nicht auf sinnvollen Werten operieren können. Die Klasse beinhaltet zwei String Attribute, ***suit*** (Farbe) und ***rank*** (Wert). Beide Attribute sollen in der Klasse selbst, sowie in Unterklassen sichtbar sein. Stellen Sie durch weitere Elemente der Klasse sicher, dass auf die Attribute auch außerhalb des definierten Sichtbarkeitsbereichs lesend zugegriffen werden kann. Die Werte der beiden Attribute sollen nur einmal beschrieben werden, nämlich in einem öffentlich zugreifbaren ***Konstruktor***. Weiterhin soll die Klasse eine öffentliche Methode ***compareTo*** anbieten, die es erlaubt ein Objekt der Klasse PlayingCard mit einer weiteren Spielkarte zu vergleichen. D.h. die Methode hat einen Eingabeparameter der Klasse PlayingCard und gibt folgenden Rückgabewert zurück:

- 0, falls Farbe und Wert beider Objekte identisch sind
- 1, falls Farbe und/oder Wert des Übergabeparameters niederwertiger sind
- -1 in allen anderen Fällen.

Überlegen Sie sich, ob der Rumpf dieser Methode bereits in der Klasse PlayingCard implementiert werden sollte und wählen Sie dementsprechend (k)ein Schlüsselwort in der Methodendeklaration.

Überschreiben Sie die ***toString*** Methode in der Klasse PlayingCard derart, dass sie die Farbe der Karte gefolgt von einem Leerzeichen und dem Wert der Karte zurückgibt.

Die Fehlerbehandlung in der Klasse PlayingCard wird nicht bewertet.

4 Interface Deck (1 Punkt)

Erstellen Sie das Interface ***Deck*** (Kartenstapel) mit den folgenden Methoden:

- Methode ***getCards*** gibt eine Kopie aller Spielkarten des Kartenstapels als Array zurück
- Methode ***getCount*** gibt die aktuelle Anzahl von Spielkarten im Kartenstapel zurück
- Methode ***addCard*** legt eine Spielkarte auf dem Kartenstapel ab. Es wird festgelegt, das die unterste Karte am kleinsten Index einer entsprechenden Datenstruktur zu finden ist
- Methode ***insertCard*** fügt eine Spielkarte an einer beliebigen Position des Kartenstapels ein. Es ist definiert, dass der Index der angegebenen Position beginnend von 1 gezählt wird.
- Methode ***drawCard*** hebt die oberste Spielkarte vom Kartenstapel ab und gibt sie zurück
- Methode ***shuffleDeck*** mischt den Kartenstapel
- Methode ***sortDeck*** sortiert den Kartenstapel
- Methode ***dealCards*** teilt Spielkarten an eine gegebene Anzahl von Spielern aus und gibt die Spielkarten für alle Spieler in Form einer 2-dimensionalen Matrix zurück. D.h. die Zeilen der Matrix entsprechenden den Spielern eines Kartenspiels und die Spalten entsprechen den Karten die jedem Spieler zugeteilt werden
- Methode ***printDeck*** gibt alle Spielkarten im Kartenstapel, beginnend mit der obersten Karte, in ihrer aktuellen Reihenfolge auf der Konsole aus.

5 Klasse SkatCard (2 Punkte + 1 Zusatzpunkt)

Implementieren Sie die abstrakte Klasse ***PlayingCard*** mit der Klasse ***SkatCard*** (Skatkarte). Überlegen Sie sich wie Sie die verschiedenen Werte, welche die Attribute ***suit*** und ***rank*** einer Skatkarte annehmen können, in der Klasse SkatCard abbilden. Wägen Sie Instanz- und Klassenattribute gegeneinander ab.

Es wird festgelegt, dass die Methode ***compareTo*** Spielkarten erst anhand der Farbe (suit) und dann anhand des Werts (rank) vergleicht. Hinweis, für den Vergleich können Sie sich z.B. auch eine Hilfsmethode überlegen, welche die Werte des rank Attributes in ganze Zahlen abbildet, um das Vergleichen einfacher zu machen.

Die Fehlerbehandlung in der Klasse SkatCard wird nicht bewertet.

6 Klasse SkatGame (8,5 Punkte + 3 Zusatzpunkte)

Implementieren Sie das Interface ***Deck*** mit der Klasse ***SkatGame*** (Skat Spiel). Deklarieren Sie notwendige Attribute, um die Spielkarten des Kartenstapels in einem Array zu verwalten. Überlegen Sie sich eine geeignete Sichtbarkeit der Attribute.

Bestimmen und Verwenden Sie geeignete **Konstanten** die grundlegende Eigenschaften eines Skat Spiels kodieren, mindestens die Anzahl der Karten eines Skatspiels, Anzahl der Karten pro Spieler und die Anzahl der aktiven Spieler.

Implementieren Sie einen **Konstruktor** der die Attribute der Klasse initialisiert, inklusive der Erzeugung und Speicherung aller 32 Karten eines Skatspiels. Hinweis, für das Hinzufügen von neuen Karten zum Kartenstapel kann man Methoden aus dem Interface Deck verwenden.

Stellen Sie sicher, dass die Methode ***getCards*** eine Kopie des Stapels anstelle einer Referenz zurückgibt. Bedenken Sie die Datentypen (primitive vs. Referenz) der Elemente des Arrays aller Spielkarten.

Beachten Sie, dass die Methoden ***addCard*** und ***insertCard*** so implementiert werden, dass

- Duplikate auf dem Kartenstapel vermieden werden und
- Nur zulässige Skatkarten dem Kartenstapel hinzugefügt werden können.

Stellen Sie sicher, dass die Karte, die mit der Methode ***drawCard*** gezogen wird, auch vom Kartenstapel entfernt wird.

Für die Methode ***shuffleDeck*** sind Sie frei in der Wahl der Implementierung. Hinweis: man kann das Mischen der Karten mit einer Schleife implementieren, die Karten vom Kartenstapel nimmt und sie wieder an einer beliebigen Position einfügen. Hierzu können Sie auf die Methode ***Random.nextInt(int bound)*** aus der Java API zurückgreifen. Für die Anzahl der Iteration haben Sie freie Hand. Allerdings sollte das Mischen nicht länger als 1 Sekunde dauern.

Beachten Sie, dass die Methode **sortDeck** den Kartenstapel in absteigender Reihenfolge sortieren soll.

Die Methode ***printDeck*** soll die Karten im Kartenstapel beginnend mit der obersten Karte ausgeben und für die Ausgabe die ***toString*** Methode der Klasse ***SkatCard*** benutzen.

Implementieren Sie für die Methoden der Klasse SkatGame eine rudimentäre Validierung der Eingabegrößen und wenn nötig, auch der globalen Variablen (Attribute der Klasse). Führen Sie kein Exception Handlung unter der Verwendung von Exception-Objekten durch, sondern verwenden Sie die Rückgabewerte der Methoden und das reservierte Wort ***return***.

7 Bewertung

Aufgabe	Voraus-setzung	Punkte
Programm compilierbar	X	-
Klassen, Pakete, Methoden, Signaturen richtig benannt und umgesetzt	X	-
Klasse PlayingCard		3
Interface Deck		1
Klasse SkatCard		2
Klasse SkatGame		8,5
Zusatzpunkte Klasse PlayingCard		2
Zusatzpunkte Klasse SkatCard		1
ZusatzPunkte Klasse SkatGame		3
Gesamtpunkte 100%		**14,5**
maximale Gesamtpunkte 141%		**20,5**

8 Anwendungsbeispiel

```
6  public class Main {
7
8      public static void main(String[] args) {
9
10         SkatGame game = new SkatGame();
11         System.out.println(game.getCount());
12
13         PlayingCard card = game.drawCard();
14         System.out.println(card);
15
16         game.insertCard(card, 17);
17         System.out.println(game.getCount());
18
19         System.out.println("Shuffle...");
20         game.shuffleDeck();
21         game.printDeck();
22
23         System.out.println("Sort...");
24         game.sortDeck();
25         game.printDeck();
26
27         PlayingCard[][] hands = game.dealCards(3);
28         System.out.println("letzte Karte des ersten Spielers: " + hands[0][9]);
29     }
30 }
```