

Vier Gewinnt

Dieses Dokument enthält alle Teilaufgaben zur Java-Pflichtaufgabe für das Sommersemester 2008.

Aufgabe 1 (Vier Gewinnt – 1. Teil)

Implementieren Sie eine Java-Klasse `VierGewinnt1`, mit einer `main`-Methode, die folgende Spielsituation mit Hilfe von `System.out.print`- und `println`-Anweisungen auf der Konsole ausgibt:

Rot / O am Zug.

```
| |X| | | | |
|O|X| |X| | |
|O|O|X|X| |X|
|X|X|O|O| |O|
O|O|O|X|X| |X|
O|O|X|O|X| |O|
```

Mögliche Züge sind: 0 1 3 4 5 6

Welche Spalte wählen Sie? >

Verwenden Sie die Klasse `Eingabe` von der Website der Rechnerübungen und die dortige Methode `readInt()`, um eine Zahl von der Tastatur einzulesen und in einer lokalen Variablen `spalte` zu speichern. Geben Sie dann noch folgenden Text aus (wobei die Zahl die eingegebene Zahl sein soll).

Gewählte Spalte: 2

Aufgabe 2 (Vier Gewinnt – 2. Teil)

Implementieren Sie eine Java-Klasse `VierGewinnt2`, mit einer `main`-Methode, die folgende lokalen Variablen enthält:

```
int spieler = 1; // 1 = Rot, 2 = Gelb
boolean vierInEinerZeile = false;
boolean vierInEinerSpalte = true;
boolean vierInHauptdiagonale = false;
boolean vierInNebendiagonale = false;
```

- Überlegen Sie sich einen Booleschen Ausdruck, der genau dann `true` ist, wenn das Spiel gewonnen ist, das heißt eine der vier Booleschen Variablen ist `true`.
- Führen Sie eine Boolesche Variable `gewonnen` ein, die mit diesem Booleschen Ausdruck initialisiert wird.

- Erweitern Sie das Programm um Anweisungen, die im Falle des Spielendes, die Farbe des Spielers auf der Konsole ausgibt, der gewonnen hat.

`Rot hat gewonnen`

- Erweitern Sie das Programm um Anweisungen, so dass in der Variablen `spieler` die Nummer des nächsten Spielers, der nun am Zug ist, steht und geben Sie einen entsprechenden Text auf der Konsole aus:

`Gelb ist am Zug`

Testen Sie Ihr Programm mit unterschiedlichen Werten für die oben gegebenen lokalen Variablen.

Aufgabe 3 (Vier Gewinnt – 3. Teil)

Implementieren Sie eine Java-Klasse `VierGewinnt3` mit einer Klassenvariabel, die den Wert des aktuellen Spielers enthält. Wie bisher soll Rot als 1 und Gelb als 2 kodiert werden. Verwenden Sie statt 1 und 2 für die Spielerfarbe je eine Konstante `ROT` und `GELB`. Die initiale Farbe des Spielers soll Rot sein.

Implementieren Sie folgende drei Klassenmethoden:

- `spielerWechseln()`: Diese Methode soll bei einem Aufruf den Wert der Variablen für die Spielerfarbe auf den nächsten Spieler ändern: also von Rot auf Gelb und von Gelb auf Rot. Diese Methode gibt keinen Wert zurück.
- `spielerAnzeigen()`: Diese Methode soll einen Text `Rot ist am Zug` (bzw. `Gelb ist am Zug`), wenn der aktuelle Spieler die Farbe Rot (bzw. Gelb) hat. Die Methode gibt keinen Wert zurück.
- `gelbIstAmZug()`: Diese Funktion soll genau dann `true` zurückgeben, wenn der Spieler mit der Farbe Gelb gerade am Zug ist.

Aufgabe 4 (Vier Gewinnt – 4. Teil)

Laden Sie sich die Klasse `VierGewinnt4.java` von der Webseite der Rechnerübungen herunter. Diese enthält einige Methode, deren Implementierung fehlt, weil wir noch nicht genügend Java-Sprachelemente dazu kennen. Lesen Sie sich den Javadoc-Kommentar der Methoden durch, um zu verstehen was die Methoden machen sollen. Für diese Aufgabe nehmen wir an, dass die Methoden implementiert wären.

Erweitern Sie die Klasse um eine Klassenvariable für den aktuellen Spieler, um zwei Konstanten für die Farben Rot und Gelb, sowie um eine Variable, die die Spielerfarbe des Gewinners enthalten soll. Diese Variable soll mit 0 initialisiert werden (0 bedeutet: noch kein Gewinner vorhanden).

Implementieren Sie folgende Klassenmethoden. Wählen Sie Klassenennamen, die klar aussagen, was die Methoden machen sollen:

- Eine Methode, die den Wert des aktuellen Spielers auf die nächste Farbe setzt (von Rot auf Gelb und von Gelb auf Rot).
- Eine Methode, die einen Text auf der Konsole ausgibt, um den Gewinner anzuzeigen: etwa `Rot hat gewonnen`. Bei Aufruf der Methode, darf dieser Text nur ausgegeben werden, wenn ein Gewinner existiert.
- Eine Methode, die das Spielfeld ausgibt. Da wir das Spielfeld noch nicht implementiert haben, geben Sie mit Ausgabeanweisungen irgend eine Spielsituation aus.
- Eine Methode, die auf der Konsole die möglichen Spielzüge ausgibt und einen Text, um den Benutzer aufzufordern eine Eingabe zu machen. Auch hier gebene Sie einfach für eine Spielsituation einen fixen Text aus.
- Eine Methode, mit der auf Basis aller anderen Methoden, der Spielablauf implementiert wird:
 1. Solange es noch keinen Gewinner gibt, soll das Spielfeld ausgegeben werden und der aktuelle Spieler aufgefordert werden einen Stein zu setzen.
 2. Der Spieler muß solange aufgefordert werden, bis er eine Spaltennummer angegeben hat, in der ein Stein gesetzt werden kann.
 3. Wenn dann das Spiel gewonnen ist, muß die Farbe des Siegers in die zugehörige Variable geschrieben werden.
 4. Die Farbe des aktuellen Spielers muß wechseln (Rot auf Gelb, Gelb auf Rot).
 5. Am Ende des Spiels soll der Gewinner angezeigt werden.
- Implementieren Sie eine `main`-Methode, die die Simulation startet. Testen Sie die Methoden mit unterschiedlichen Werten für die Klassenvariablen.

Aufgabe 5 (Vier Gewinnt – 5. Teil)

Implementieren Sie die 4. Teilaufgabe in einer Klasse `VierGewinnt` objekt-orientiert. Ausser den Konstanten für die Farben Rot und Gelb, dürfen nur Objektvariablen und -methoden enthalten sein. Alle Methoden, bis auf diejenige, welche die Simulation startet, müssen als `private` deklariert sein. Verwenden Sie die objekt-orientierte Version der Klasse `Eingabe` von der Website der Rechnerübungen.

Implementieren Sie eine weitere Klasse mit einer `main`-Methode, in der Sie eine Instanz von `VierGewinnt` erzeugen, in einer lokalen Variablen speichern und dann die Simulation starten.

Aufgabe 6 (Vier Gewinnt – 6. Teil)

Programme werden in der Praxis in unterschiedliche Module (in Java sind dies Klassen) aufgeteilt. Einzelne Module lassen sich besser testen, ändern und wiederverwenden. Ein sehr gängiges Schema ist die Aufteilung der Software in die folgenden drei Einheiten:

- Dem Modell (model): Diese enthält keine Ein- oder Ausgaben, sondern nur Klassen mit reiner Verarbeitungslogik.
- Der Sicht (view): Diese enthält nur Programmanweisungen zur Ausgabe auf dem Bildschirm. Vom View wird über Methoden des Modells auf dessen Daten zugegriffen, um sie auszugeben.
- Der Steuerung (control): Diese wertet Eingaben – zum Beispiel von der Tastatur – aus und entscheidet in Abhängigkeit der Eingabe, welche Programmteile daraufhin ausgeführt werden sollen. Der Controller ruft Methodenvom View und dem Modell auf.

Das Modell darf keine Methoden vom View oder Controller aufrufen. Das View darf keine Methoden vom Controller aufrufen. Um auf die Methoden des Modells zuzugreifen, hat das View eine Beziehung zu den Modellinstanzen: diese werden üblicherweise im Konstruktor übergeben. Ebenso hat der Controller eine Beziehung zu den Modell- und den Viewinstanzen.

Bei der Vier-Gewinnt-Simulation ist zum Beispiel die Ausgabe von Text, die den aktuellen Spielstand ausgibt, im View zu implementieren. Der Simulationsablauf mit der Auswertung der Benutzereingaben ist im Controller zu implementieren. Die Klasse Eingabe gehört zum Controller. Das Modell enthält alle Daten und Methoden, die diese Daten auswerten und verarbeiten. Der Controller nutzt die Methoden vom Modell. Der Controller (und der View) darf keine Berechnungen oder Vergleiche von Daten des Modells selbst durchführen: eine Abfrage, ob der Gewinner den Wert 1 hat, ist immer im Model als Funktion zu implementieren (z.B. `rotIstGewinner()`).

Implementieren Sie drei Klassen: `VierGewinnt` (Modell), `VierGewinntSicht` (view) und `VierGewinntSteuerung` (control) mit den bisherigen Methoden. Implementieren Sie die `main`-Methode in der Steuerung.

Aufgabe 7 (Vier Gewinn – Letzter Teil)

Das Spielfeld soll als zwei-dimensionales Feld bei der Modellklasse implementiert werden (7 Spalten mit 6 Zeilen). Jetzt können auch die noch fehlenden oder leeren Methoden implementiert werden:

- Im View kann das Spielfeld ausgegeben werden. Dabei soll X der Spielstein für Gelb und O für Rot sein. Beachten Sie das der View nicht selbst direkten Zugriff auf das Feld haben darf.
- Im View sollen bei den möglichen Spielezügen nur die noch nicht vollen Spalten angezeigt werden.
- Im Modell ist zu überprüfen, ob eine Stein in eine bestimmte Spalte geworfen werden kann

- Im Modell soll die Methode zum Hineinwerfen eines Stein in eine Spalte implementiert werde.
- Im Modell soll eine Funktion implementiert werden, die genau dann true zurückgibt, wenn beim geworfenen Stein Vier Steine in einer Reihe entstehen. Dazu müssen Sie ausgehend von der Spielsteinposition für die Vertikale, Horizontale und den beiden Diagonalen zählen, wie viele Steine gleicher Farbe dort enthalten sind. Implementieren Sie für jeden dieser vier Fälle eine eigene Funktion.

Das Spiel soll komplett spielbar sein.

Testen Sie Ihre Modellklasse mit JUnit. Für jede öffentliche Methode, muss mindestens eine Testmethode vorhanden sein.