ulm university universität ulm

1. Zulassungsklausur (C) zur Veranstaltung

Objektorientierte Programmierung

im Sommersemester 2023

Prüfer: Prof. Dr. Matthias Tichy Fakultät Ingenieurwissenschaften, Informatik, Psychologie

24.06.2023, 12 Uhr **Bearbeitungszeit: 45 min**

Nachname:	Vorname:	Matrikelnummer:
Studiengang und Abschluss:		Account:

Hiermit erkläre ich, dass ich prüfungsfähig bin.
Sollte ich nicht auf der Liste der angemeldeten Studierenden aufgeführt sein, dann nehme ich hiermit zur Kenntnis, dass diese Prüfung nicht gewertet werden wird.

_____

Datum, Unterschrift des Prüfungsteilnehmers

Zur allgemeinen Beachtung:

- Füllen Sie das Deckblatt vollständig und korrekt aus.
- Lesen Sie sich zunächst die Klausur sorgfältig durch (sie besteht aus 4 Seiten).
- Zum Erreichen der Vorleistung benötigen Sie **15** der insgesamt *31* Punkte.
- Im Ordner ∼/`materials` finden Sie das zu bearbeitende Projekt sowohl als ZIP als auch entpackt.
- Sie dürfen im Projekt beliebig Hilfsmethoden und Klassen hinzufügen. Die Signaturen der vorgegebenen Methoden und Interfaces dürfen nicht verändert werden. Die schon implementierten Methoden dürfen nicht geändert werden.
- **Vor Beginn der Bearbeitungszeit** haben Sie 5 Minuten Zeit um einen Editor/eine IDE Ihrer Wahl zu starten und sich in den Materialien zurecht zu finden.
- **Nach Ablauf der Bearbeitungszeit** haben Sie 5 Minuten Zeit um Ihre Lösung im Ordner ∼/`export` abzulegen. **Bleiben Sie sitzen!**
- Abgaben sind als Projektexport in Zip-Form abzugeben. Achten Sie darauf Ihr **gesamtes** Projekt, mit allen zugehörigen Ordnern, zu exportieren.
- Abgaben die Kompilefehler produzieren werden nicht bewertet.

Punkteverteilung				
1 von 07	2 von 09	3 von 15	$\sum$ von 31	**Note**
				Korrektur

o	o					
x	x	x	x			
x	o	o	x			
x	o	x	o	o	x	o

Tabelle 1: Beispiel-Spielfeld. Der Spieler der 'x' benutzt hat gewonnen.

Das Projekt

Das Ihnen zur Verfügung gestellte Projekt soll das Spiel 4-Gewinnt implementieren. Bei diesem Spiel treten zwei Spieler gegen einander an. Die Spieler werfen abwechselnd Spielsteine in eine von 7 Spalten, die jeweils 6 Plätze tief sind. Spielsteine belegen immer die unterste freie Stelle in einer Spalte. Der Spieler, der zuerst eine horizontale, vertikale oder diagonale Linie mit seinen Symbolen erstellt, hat eine Runde gewonnen. Es werden maximal fünf Runden gespielt. Hat ein Spieler bereits dreimal gewonnen, gilt das Spiel als beendet.

Das Spielfeld ist ein 6x7 Char-Array. Das die 7 Spalten und 6 Positionen pro Spalte beschreibt. `[5][0]` ist die Position ganz unten links, während `[0][6]` die Position ganz oben rechts ist. Während des Spiels geben die Spieler die Spalte, in die sie einen Spielstein werfen möchten über die Nummer der Spalte von links nach rechts gezählt (1-7) an.

Das Projekt besteht aus den folgenden **4** Klassen:

`oop.sose2023.admission_exam.Main`: Diese Klasse initialisiert ein neues Spiel zwischen den Spielern Bob und Alice, und enthält die Logik, dass ein Spiel maximal 5 Runden dauert, abgesehen davon, wenn ein Spieler bereits 3-mal gewonnen hat.

`oop.sose2023.admission_exam.group03.Game`: Diese Klasse implementiert den Spielzustand und die Spiellogik.

`oop.sose2023.admission_exam.group03.Player`: Diese Klasse implementiert den Zustand der einzelnen Spieler.

`oop.sose2023.admission_exam.group03.util.InputManagement`: Diese Klasse implementiert die Logik, um Eingaben über die Konsole als Koordinaten zu interpretieren.

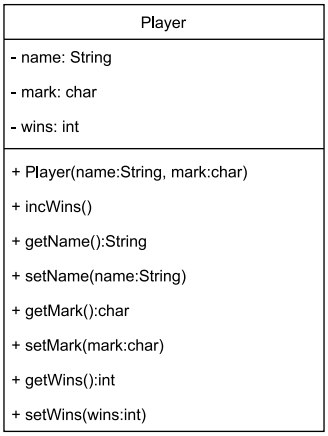

Abb. 1: Klassendiagramm zur Verwaltung von Spielerinformationen

Aufgabe 1 - Player **07 Punkte**

Implementieren Sie die Klasse `Player`, die den Zustand eines Spielers verwaltet. Abbildung **1** zeigt das Klassendiagramm für die zu implementierende Klasse `Player`. Ihre Implementierung soll nur die dargestellten Attribute und Methoden implementieren. Achten Sie auf die Schreibweise der Namen, die angegebenen Sichtbarkeiten und die festgelegten Typen. Übernehmen Sie auch die im Klassendiagramm vorgegebenen Sichtbarkeiten: **Private** Attribute oder Methoden werden mit einem vorangestellten – gekennzeichnet. **Öffentliche** Attribute oder Methoden werden mit einem vorangestellten + gekennzeichnet. Die Klasse soll nur einen Konstruktor zur Verfügung stellen. Dieser soll den Namen (`name`) eines Spielers und dessen Symbol (`mark`) als Parameter übergeben bekommen und die gleichnamigen Attribute in der Klasse setzen. Zudem soll der Konstruktor die Anzahl der gewonnen Runden (`wins`) auf 0 setzen. Die Methode `incWins()` soll den Wert im Attribut `wins` um 1 erhöhen. Methoden, die mit dem Präfix `get` beginnen, sollen jeweils den Wert des gleichnamigen Attributes zurückgeben. Methoden, die mit dem Präfix `set` beginnen, sollen jeweils das gleichnamige Attribut auf den übergebenen Parameterwert setzen.

Aufgabe 2 - runGame() **09 Punkte**

Implementieren Sie in der Klasse `Main` die Methode `runGame(Game game)`. Diese Methode realisiert den Prozess, der hinter den einzelnen Spielzügen steckt. Dies bedeutet, dass solange kein Gewinner feststeht weiterhin die Spieler neue Zielspalten für ihre Züge eingeben, das entsprechende Symbol auf dem Spielfeld gesetzt wird und anschließend der Spielzustand evaluiert wird. Verwenden Sie die Methoden der Klasse `Game` und beachten Sie die JavaDoc Kommentare in der Klasse `Game`. Beachten Sie, dass solange das Attribut `winner`, eines `Game`-Objekts, den Wert -1 besitzt, gibt es noch freie Felder auf dem Spielfeld und noch keinen Gewinner.

Aufgabe 3 - Gewinn Erkennung **15 Punkte**

Ein Spieler bei 4-Gewinnt hat gewonnen, wenn die Steine seiner Farbe (bzw. seines Symbols) eine horizontale, vertikale oder diagonale 4er Reihe auf dem Spielfeld bildet. Die Methode `horizontalWin()` gibt zurück, ob ein Spieler eine horizontale Reihe erzielt hat (vgl. Abb. **2**) und die Methode `diagonalBLtoURWin()` gibt zurück, ob ein Spieler eine diagonale Reihe, die von links unten (Bottom Left (BL)) nach rechts oben (Upper Right (UR)) verläuft geschafft hat (vgl. Abb. **3**).

Implementieren Sie die entsprechende Methoden `verticalWin()` für den Sieg durch das erreichen einer vertikalen Reihe (vgl. Abb. **4**) und die Methode `diagonalBRtoULWin()` für eine diagonale Reihe von rechts unten nach links oben (vgl. Abb. **5**).

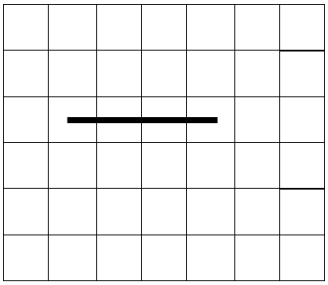

Abb. 2: Beispiel für eine horizontale Reihe.

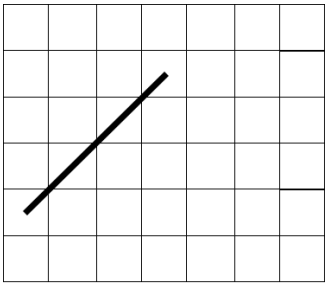

Abb. 3: Beispiel für eine diagonale Reihe von unten links nach oben rechts.

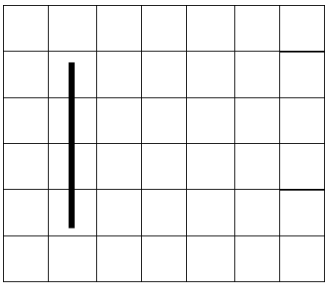

Abb. 4: Beispiel für eine vertikale Reihe.

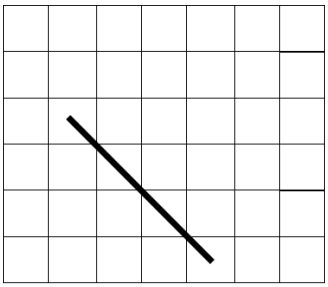

Abb. 5: Beispiel für eine diagonale Reihe von unten rechts nach oben links.

Implementieren Sie anschließend in der Klasse `Game` die Methode `checkWinner()`. Diese Methode prüft, ob der Spieler des aktuellen Spielzugs gewonnen hat oder nicht. Ist dies der Fall, soll die Methode `true` zurückliefern, sonst `false`. Beachten Sie, dass der Index des Spielers des aktuellen Spielzugs im Attribut `currentPlayer` verwaltet wird. Verwenden Sie die Methoden `horizontalWin()`, `verticalWin()`, `diagonalBLtoURWin()` und `diagonalBRtoULWin()`.