

Anmerkungen

- **Abgabe:** Jede Aufgabe in einem separaten Ordner. Bei 5-1 und 5-2 Quellcode als .java-Dateien, bei 5-3 Lösung als .pdf-Datei.
- Jede Quellcode-Datei enthält **Name(n)** und **Matrikelnummer(n)**.

Aufgabe 5-1

Sie sollen ein “Vier gewinnt“ Spiel programmieren, bei dem man wahlweise gegen einen menschlichen Gegner oder den Computer spielen kann.

Laden Sie von ILIAS die Dateien `VierGewinnt.java`, `HumanPlayer.java`, `ComputerPlayer.java`, `Token.java` und `IPlayer.java` herunter. Die Klasse `VierGewinnt` enthält bereits Methoden `play()` (definiert den Spielablauf), `main` (startet das Spiel) und `displayField()` (graphische Darstellung des Spielfelds):

```

Player X choose a column between 1 and 7: 2
+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+
| X |   |   |   |   |   |   |
+---+---+---+---+---+---+
| 0 | X | 0 |   |   |   |   |
+---+---+---+---+---+---+
| X | 0 | X |   |   |   | X |
+---+---+---+---+---+---+
| 0 | 0 | 0 | X | X |   | 0 |
+---+---+---+---+---+---+
      1       2       3       4       5       6       7
Player X wins!

```

Um das Spiel zum Laufen zu bekommen, müssen Sie in der Klasse **VierGewinnt** die folgenden Methoden implementieren (die anderen gegebenen Methoden dürfen Sie *nicht* verändern):

1. **insertToken**: Der übergebene Stein (Token-Objekt) soll in die gewählte Spalte (**column**) des Spielfelds (Array **board**) gefüllt werden. Falls eine nicht existierende oder bereits bis oben gefüllte Spalte gewählt wurde, soll das Programm mit einer Fehlermeldung abbrechen. Verwenden Sie dazu **System.exit(1)**.
2. **isBoardFull**: gibt genau dann **true** zurück, wenn alle Felder durch einen Stein besetzt sind.
3. **checkVierGewinnt**: überprüft – ausgehend vom durch **col** und **row** gegebenen Feld – ob es in einer der vier Richtungen (d.h. **-**, **|**, **/**, ****) mindestens vier gleiche Steine gibt. In diesem Fall wird **true** zurückgegeben, andernfalls **false**. Tipp: Schreiben Sie für jede der vier Richtungen eine Hilfsmethode.

Verbessern Sie anschliessend die Klasse `ComputerPlayer` derart, dass der Computer zumindest ein wenig intelligenter spielt. Stellen Sie sicher, dass "er" keine ungünstigen Züge macht, also insbesondere keine Spalte auswählt, die bereits voll ist.

Aufgabe 5-2

Laden Sie von ILIAS die Dateien `Book.java` und `Store.java` herunter (**verwenden Sie nicht die Datei aus der Serie 3**). Das Programm `Store` verfügt über ein Menü, anhand dessen man neue Bestellung erfassen kann. Bestellungen bestehen aus (beliebig vielen) Büchern, DVDs und CDs:

```

=====
|  1. Create a new order      2. Show all registered articles  |
|  3. Show all orders        9. Exit                          |
|=====
What do you want to do? 1

1 (Book) Die Blechtrommel, by Guenter Grass, 1959, 29 CHF
3 (Book) L'Etranger, by Albert Camus, 1942, 25 CHF
4 (DVD) Casablanca, 1942, 29 CHF
6 (CD) Nirvana, Nevermind, 1991, 19 CHF
8 (CD) Britney Spears, ...Baby One More Time, 1999, 50 CHF

Enter id of ordered article (press x when done): 6
Successfully added: 6 (CD) Nirvana, Nevermind, 1991, 19 CHF
Enter id of ordered article (press x when done): 4
Successfully added: 4 (DVD) Casablanca, 1942, 29 CHF
Enter id of ordered article (press x when done): x
Enter the customer's name: Susi Meier
Enter the customer's address: Mittelstrasse 10, 3011 Bern
    
```

Ihre Aufgabe ist es, dafür zu sorgen, dass das Programm `Store` einwandfrei funktioniert. `Store` selbst darf *nicht* verändert werden.

Sie müssen also folgende Klassen und Interfaces programmieren:

1. Ein Interface `IArticle`, das die folgenden Methoden definiert: `int getId()`, `int getPrice()` und `String getDescription()`. Passen Sie die Klasse `Book` derart an, dass sie dieses Interface implementiert.
2. Schreiben Sie Klassen `DVD` und `CD`, die beide das Interface `IArticle` implementieren. `CD` soll einen Interpreten statt eines Autors und ansonsten die gleichen Attribute haben wie `Book`, wogegen `DVD` *kein* Feld `author` haben soll.
3. Verbessern sie die Klasse `Order` aus der Übungsserie 3 derart, dass eine Bestellung beliebig viele `IArticle`-Objekte enthalten kann. `Order` soll also nicht mehr auf (fünf) Bücher beschränkt sein.
4. Zeichnen Sie ein UML-Klassendiagramm aller involvierten Klassen und Interfaces.

Welche Methoden die Klasse `Order` bereitstellen muss, können Sie der Klasse `Store` entnehmen. Insbesondere muss `Order` eine Methode `getOrderedArticles()` besitzen. Definieren sie dessen Rückgabetyt als `Iterable<IArticle>!`

Überlegen Sie sich die Antwort zu folgender Frage: Welcher Nachteil entstünde, wenn die Methode `getOrderedArticles()` einen Array oder eine `ArrayList` zurückgeben würde (statt eines `Iterable`-Objektes)?

Aufgabe 5-3

1. Welches Problem tritt beim Ausführen des folgenden Programmcodes auf? Wie kann man es beheben?

```
int[] numbers = {1,2,3,5,8,13,21};
for(int i=2; i<=numbers.length; i++){
    System.out.println(numbers[i]);
}
```

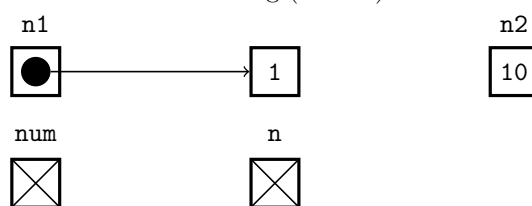
2. Gegeben sei die Klasse Num aus der Vorlesung:

```
public class Num{
    private int value;
    public Num(int update){ value = update; }
    public void setValue(int update){ value = update; }
    public String toString(){ return value + ""; }
}
```

Zeichnen Sie ein Diagramm, das die Wertzuordnungen für das folgende Programm illustriert (analog zu Abb. 6.5 im Buch bzw. Handout der Vorlesung).

```
1 public class Increment {
2     public static void main(String[] args) {
3         Num n1 = new Num(1);
4         int n2 = 10;
5         modify(n1);
6         modify(n2);
7     }
8     public static void modify(Num num){ num.setValue(100); }
9     public static void modify(int n) { n=100; }
10 }
```

Nach der Initialisierung (Zeile 4):



Bei Aufruf von `modify(Num)`:

...