Prof. Dr. M. Huhn
Fakultät Informatik
SS 2020

# Programmieren – Laboraufgabe 1

**– Der Spielstart –**

## Hinweise

- Legen Sie bitte für jede Aufgabenserie ein eigenes Projekt mit dem Projektnamen `ProgSS20A`$i$ an, wobei $i$ die Aufgabennummer ist und $i \in \{1, \ldots, 5\}$. Ihr Code liegt im Unterordner `src`. Als Paketnamen verwenden Sie dort bitte `de.ostfalia.prog.ss20`.

Das zuvor beschriebene Spiel wird ab jetzt als *Standardspiel* bezeichnet. Es wird durch die Aufgaben stückweise implementiert. Beim Entwurf ist darauf zu achten, die Implementierung offen für Erweiterungen und Änderungen zu halten. Insbesondere sollen einige Spieleigenschaften - wie die Startpositionen einiger Figuren - beim Spielstart anpassbar sein.

## Spielbeginn

### Aufgabe 1-1 (Schlümpfe wandern auf Pfaden, 5 Punkte)

Ziel dieser Aufgabe ist es, dass die Spieler ihre Schlümpfe durch den Schlumpfwald wandern lassen können. Im Schlumpfwald gibt es zwar schon die Fliege „Bzz“ und den Oberschlumpf „Doc“, aber beide sind unbeweglich. Eine Begegnung mit ihnen ist für die Schlümpfe bedeutungslos.

a) Entwerfen Sie eine objektorientierte Programmstruktur, die die Aufgaben 1 bis 3 abdeckt und stellen Sie sie in der Form eines UML-Klassendiagramms im Labor vor.

   Sie können in der Informatik-Lounge Rückmeldungen zu Ihren Entwurf einholen, um den Refactoring-Aufwand gering zu halten.

b) Implementieren Sie die Aufzählungstypen (enum) Farbe und Richtung. Achten Sie bitte darauf, dass die Werte - wegen der Testbarkeit - in der gegebenen Reihenfolge vorkommen müssen:

   ```
   Farbe {GELB, ROT, BLAU, GRUEN}
   ```

   ```
   Richtung {WEITER, ABZWEIGEN}.
   ```

c) Implementieren Sie geeignete Klassen, so dass

   - 1 bis 4 Spieler (repräsentiert durch ihre Farben) angelegt werden können,
   - diese eine Datenstruktur für das Spielbrett bereitstellen,
   - für jeden Spieler[1] werden 4 Schlümpfe auf dem Startfeld positioniert[2]. Um die Testbarkeit zu gewährleisten, heißen die Schlümpfe der Farbe `BLAU` `"BLAU-A"`, `"BLAU-B"`, `"BLAU-C"` und `"BLAU-D"`, analog für die Farben `GELB, ROT, GRUEN`.
   - Ihre Simulation soll so flexibel gestaltet sein, dass Sie nicht nur das Standardspiel generieren können, sondern auch Variationen davon. Dazu gehören insb.:
   - die Fliege „Bzz“ und der Oberschlumpf können am Anfang des Spiels auf einem beliebigen, aber regelkonformen Feld positioniert werden.

[1] repräsentiert durch seine Farbe
[2] über den Konstruktor `public ZombieSchluempfe (Farbe... farben)` siehe nächste Seite

**Abnahme: 17.03.2020**
**5 Punkte**

- die Position ausgewählter Figuren kann auf dem Spielbrett über einen `String` gemäß der im Interface beschriebenen Kodierung vorgegeben werden.
  Beispiel: `"BLAU-A:30,␣BLAU-B:28,␣BLAU-C:30,␣BLAU-D:28,␣GELB-A:30,␣GELB-B:28"` beschreibt, dass Schlumpf `"BLAU-A"` auf Feld 30, Schlumpf `"BLAU-B"` auf Feld 28 usw. positioniert wird. Die im String nicht genannten Figuren werden auf ihren Standard-Startfeldern positioniert.
- Für die Verzweigungsfelder im Standardspielbrett gilt:

  **Feld 3:** `Richtung.WEITER` führt zu Feld 4. `Richtung.ABZWEIGEN` führt zu Feld 8.

  **Feld 31:** `Richtung.WEITER` führt zu Feld 32. `Richtung.ABZWEIGEN` führt zu Feld 36.
- Schlümpfe können entlang der Pfade auf dem Spielbrett gezogen werden.
- Der Oberschlumpf wird zu Beginn des Spiels auf einem Feld positioniert, bewegt sich aber niemals.

Folgende Teile der Spielregeln können in dieser Aufgabe ignoriert werden:

- Pilz- und Blütenstaub-Felder können ignoriert werden.
- Das Beenden eines Zugs auf einem Flussfeld wird nicht getestet.
- Die Schlumpfine kommt nicht vor.
- Es treten während des Spielverlaufs keine Zombieschlümpfe auf.

Es ist das zentrale Interface `IZombieSchluempfe` in einer Klasse `ZombieSchluempfe` zu implementieren, d.h. alle Methoden aus dem Interface `IZombieSchluempfe` sind zu überschreiben. Zusätzlich sind folgende Konstruktoren vorzusehen:

- `public ZombieSchluempfe (Farbe... farben)`
- `public ZombieSchluempfe (String conf, Farbe... farben)`

Weitere Hinweise:

- Überschreiben Sie in allen Ihren Klassen die Methode `toString()`, um eine sinnvolle Ausgabe zu ermöglichen.
- In dieser Aufgabe genügt es, wenn Sie Ihre Methoden in einer `main`-Methode aufrufen und unter Verwendung von `toString()` sinnvolle Ausgaben in der Konsole erhalten. Es wird kein Konsolen-Dialog erwartet.
- Die Generierung von zufälligen Zahlen kann mit Hilfe der Methode Math.random() oder der Klasse Random (java.util.Random) simuliert werden.

Das Interface `IZombieSchluempfe` finden Sie in Moodle `https://moodle.ostfalia.de/mod/resource/view.php?id=27376`.