# Klausur zur Vorlesung: Einführung in die objektorientierte Programmierung mit Java WS1415

Alexander Bazo 19. Februar 2015

---

## Allgemeine Hinweise

1. Die Bearbeitungszeit beträgt 60 Minuten. Sie könne 50 Punkte erreichen.
2. Verwenden Sie für das Lösen der Aufgaben die Programmiersprache Java. Alle gegebenen Code-Fragmente sind ebenfalls in Java verfasst. Halten Sie sich an die bekannten Regeln zur Codequalität.
3. Schreiben Sie Ihren Namen, Vornamen, Studiengang und Ihre Matrikelnummer leserlich unten auf jedes Angabenblatt bevor Sie mit der Bearbeitung beginnen! Blätter ohne diese Angaben werden nicht gewertet. Wenn Sie die Rückseite eines Blattes verwenden, notieren Sie dies bitte auf der Vorderseite. Kennzeichnen Sie eindeutig, zu welcher Aufgabe eine Lösung gehört.
4. Benutzen Sie keine Bleistifte, keine rotschreibenden Stifte und kein TippEx (oder ähnliche Produkte).
5. Die Klausur ist als *Closed Book*-Klausur angelegt. Sie dürfen keine mitgebrachten Quellen oder Notizen zur Bearbeitung der Aufgaben verwenden. Technische Hilfsmittel sind ebenfalls nicht erlaubt.
6. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Klausuraufsicht (Hand heben). Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für den gesamten Hörsaal durchgegeben. Aussagen unter vier Augen sind ohne Gewähr.
7. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung zu einer Aufgabe ergeben also null Punkte.

# Teil 1: Theorie

## 1) Variablen (2 Punkte)

Kreuzen Sie alle syntaktisch korrekten Variablendeklarationen an.

- ☐ `public static final int NUMBER;`
- ☐ `private void String s;`
- ☐ `private double d;`
- ☐ `float 2f;`

## 2) Schleifen (4 Punkte)

Beschreiben Sie kurz den Aufbau und einen möglichen Verwendungszweck einer `for`-Schleife.

## 3) Codequalität (4 Punkte)

Nennen Sie zwei Merkmale **schlechten** Programmcodes und beschreiben Sie kurz jeweils ein Problem, das mit diesen verbunden ist.

# Teil 2: Codeverständnis

**Hinweise:** Die Methode `System.out.println(String s)` gibt den String-Parameter `s` auf der Konsole aus. Ein Zeichen `(char)` an einer beliebigen Position einer String-Instanz kann über die Methode `charAt(int index)` der String-Klasse ausgelesen werden.

## 4) Codeanalyse (5 Punkte)

Geben ist die Methode `doStuff(String[] words)`. Beschreiben Sie kurz den Aufbau der Methode und nennen Sie anschließend den wahrscheinlichen Einsatzzweck. Gehen Sie davon aus, dass der gegebene Code korrekt ist.

```
1 public double doStuff(String[] words) {
2     int counter = 0;
3     for (int i = 0; i < words.length; i++) {
4         counter += words[i].length();
5     }
6     return (double) counter / words.length;
7 }
```


Name, Studiengang & Matrikelnummer:

## 5) Fehler finden (5 Punkte)

Finden Sie die Konventionsverstöße, Syntaxfehler und Bugs in der folgenden Methode. Markieren Sie die fehlerhaften Stellen im Code und beschreiben Sie den jeweiligen Fehler kurz unter Angabe der Zeilennummer. Schlecht gewählte Bezeichner für Methoden und Variablen sowie fehlerhafte Einrückungen zählen nicht als Fehler. Im Code sind fünf Fehler zu finden.

```
1  import java.util.ArrayList;
2
3  public class CharPrinter {
4      private static final int targetChar = 0;
5      private ArrayList<String> inputStrings;
6
7      public CharPrinter(String string1, String string2) {
8          inputStrings.add(string1);
9          inputStrings.add(string2)
10     }
11
12     public void printChars() {
13         for(int i=0; i <= inputStrings.size(); i++) {
14             String tmp =  inputStrings.get(i);
15             if(tmp.length() > 0) {
16             int c = targetChar;
17             String prefix = "Char "+c+" in word "+i+ ":";
18             System.out.println(prefix, tmp.charAt(targetChar));
19             }
20         }
21     }
22 }
```


Name, Studiengang & Matrikelnummer:

# Teil 3: Code implementieren

## 6) Wegpunkte und Distanzen (15 Punkte)

Gegeben ist die Klasse `WayPoint`. Diese Klasse beschreibt einen Punkt auf einer zweidimensionalen Karte anhand einer x- und y-Koordinate. Gehen Sie davon aus, dass dieser Code korrekt ist und beachten Sie die Hinweise am Ende der Aufgabenstellung (Seite 6).

```
1  public abstract class WayPoint {
2      private double xPos;
3      private double yPos;
4
5      public WayPoint(double xPos, double yPos) {
6          this.xPos = xPos;
7          this.yPos = yPos;
8      }
9
10     public double getX() {
11         return xPos;
12     }
13
14     public double getY() {
15         return yPos;
16     }
17 }
```

**6a) Implementieren Sie eine neue Klasse** `ExtendedWayPoint` **die von** `WayPoint` **erbt** und über eine öffentliche Methode `distanceTo` verfügt. Die Methode bekommt einen Parameter vom Typen `WayPoint` übergeben und berechnet die Distanz zu diesem. Das Ergebnis wird als `double`-Wert zurück gegeben.

**6b) Ergänzen Sie anschließend eine weitere Klasse** `MapHelper`**.** Diese verfügt über eine öffentliche, statische Methode `getRouteDistance`. Die Methode erwartet als Parameter ein Array vom Typen `ExtendedWaypoint`. In der Methode berechnen Sie die Länge der Route, die durch die Wegpunkte im Array beschrieben wird und geben das Ergebnis als `double`-Wert zurück. Nutzen Sie für die Berechnung die Methode aus dem ersten Aufgabenteil.

**Für das übergebene Array gilt**: Die Wegpunkte sind im Array in der korrekten Reihenfolge gespeichert. Ein Array `[A,B,C]` beschreibt den Weg von `A` über `B` nach `C`. Die zurückgelegte Strecke setzt sich aus den Teilstücken `A -> B` und `B -> C` zusammen. Ihre Lösung muss für Arrays mit einer beliebigen Anzahl an Wegpunkten funktionieren.

Für die Berechnung der Distanz `d` zwischen zwei Punkten gilt $d = \sqrt{dx^2 + dy^2}$. Dabei ist `dx` der Abstand (Differenz) der beiden x-Koordinaten und `dy` der Abstand der y-Koordinaten. In Java können Sie die Wurzel einer Zahl `a` mit der Methode `Math.sqrt(double a)` ziehen.

Name, Studiengang & Matrikelnummer:

## 7) Schiffe versenken (15 Punkte)

Gegeben sind die Klasse `Battleships` sowie die beiden Enums `Orientation` und `FieldState`. Gehen Sie davon aus, dass dieser Code korrekt ist. Die Klasse Battleships stellt das Grundgerüst für das bekannte Spiel *Schiffe versenken* (im Englischen: *Battleships*) dar. Das Spielfeld besteht aus einer zweidimensionalen Karte mit mehreren Feldern. Die Anzahl der Felder auf der x- bzw. y-Achse wird durch die Konstante `GRID_SIZE` angeben. Das Spiel nutzt ein zweidimensionales Array zur Repräsentation dieser Karte. Jedes Feld kann dabei immer nur einen von zwei möglichen Zuständen haben: frei (`FREE`) oder mit einem Schiffsteil belegt (`BLOCKED`). Dazu wird das Enum `FieldState` verwendet.

```
1  public class Battleships {
2      private static int GRID_SIZE = 5;
3      private FieldState[][] map;
4
5      //Initalisiert die Karte. Alle Felder werden mit dem Enum-
           Wert Free belegt.
6      public Battleships() {
7          map = new FieldState[GRID_SIZE][GRID_SIZE];
8          for(FieldState[] row: map) {
9              Arrays.fill(row, FieldState.FREE);
10         }
11     }
12
13     public boolean addShip(int xPos, int yPos, int size,
           Orientation orientation) {
14         // Hier gehoert Ihr Code hin
15     }
16 }
17
18 public enum FieldState {
19     FREE,
20     BLOCKED
21 }
22
23 public enum Orientation {
24     HORIZONTAL,
25     VERTICAL
26 }
```

Auf der Karte können Schiffe platziert werden. Jedes Schiff besteht dabei aus einer zusammenhängenden Linie von einem oder mehren Feldern der Karte. Jedes Schiff beginnt auf einem bestimmten Feld und setzt sich entweder nach rechts (horizontal) oder nach unten (vertikal) fort. Abbildung 1 zeigt zwei solche Schiffe. Die gepunkteten Felder kennzeichnen ein Schiff mit der Länge `3`, das auf dem Feld `(2,2)` beginnt und horizontal ausgerichtet ist. Die schraffierten Felder kennzeichnen ein Schiff mit

der Länge 2, das auf dem Feld `(0,3)` beginnt und vertikal ausgerichtet ist.

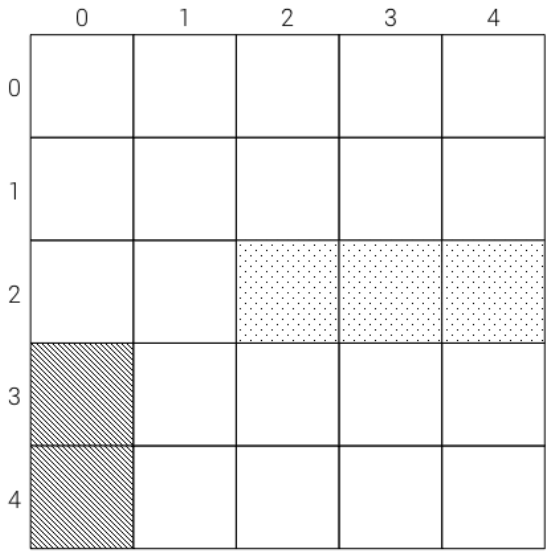


Abbildung 1: Schematische Darstellung der Karte mit zwei Schiffen.

**Ergänzen Sie den Rumpf der vorgegebenen Methode** `addShip`**.** Diese Methode bekommt Startposition, Länge und Orientierung eines neuen Schiffs übergeben. Überprüfen Sie zuerst, ob das Schiff auf der Karte platziert werden kann. Dafür muss sowohl die Startposition als auch jedes Segment des Schiffes auf die gegebene Karte passen. Anschließend aktualisieren Sie das Array `map` und ändern den Zustand jeden Feldes, das vom Schiff genutzt wird auf `Blocked`. Wenn das Schiff korrekt platziert werden konnte, gibt die Methode `true`, andernfalls `false` zurück.

**Hinweis:** Beim Platzieren des Schiffes müssen Sie nicht überprüfen, ob die genutzten Felder bereits von einem anderen Schiff verwendet werden.

Name, Studiengang & Matrikelnummer: