Grundlagen von Java

Aufgabe 1: Typen und Zuweisungen in Java

Welche der folgenden Java-Anweisungen sind fehlerhaft? Handelt es sich um einen Compiler- oder einen Laufzeitfehler?

Anmerkung: Folgefehler werden jeweils ignoriert.

```
class C {
     public static void main(String[] args) {
          short s = 7;
          int i = -12;
          unsigned int u = +12;
          long l = 123456789;
          float f = 3.1415;
          double d = 3.1415;

          s = i;
          i = s;
          s = 123456789;
          int i2 = (int) l;
          i = 12.4;
          i = 12L;
          i = (int) 12L;

          String str = "Hallo" + "Welt";
          str = args[-3];
          str = "i = " + i;
          str = str - i;

          boolean b = false;
          b = 12L == 12;
     }
}
```

Zusatz-Aufgabe 2: Installation und Einrichten des JDK

Hinweis: Diese optionale Aufgabe ist zur Einführung dafür gedacht, wie Sie Java auf der Kommandozeile (d.h. mit den Befehlen `javac` und `java`) verwenden können.

Teil 1 – Installation: Das JDK (Java Development Kit) für Java sollte bereits auf den Übungsrechnern im GLI-Labor installiert sein (zumindest in einer Windows-VM). Sofern Sie das JDK auf ihrem eigenen Rechner installieren oder von einer älteren Version updaten wollen, müssen Sie es sich von der Download-Seite (http://www.oracle.com/technetwork/java/javase/downloads/index.html) herunterladen und entsprechend installieren.

Wir benötigen für die Veranstaltung

- Das **JDK (Java Development Kit)**, dort ist der Java-Compiler und weitere Entwicklungs-Tools enthalten. Die JRE (Java Runtime Environment) reicht nicht!
- **mindestens Java Version 8** (oder eine spätere, nachfolgende Version)
- die Java Standard Version (SDK). Die Java Enterprise Version (JEE) ist eine erweiterte Version für verteilte Geschäftsanwendungen, die wir im Wahl-Modul „Fortgeschrittene Java-Programmierung“ benötigen. Jetzt können Sie jedoch erst einmal die vereinfachte SDK-Version verwenden.

Achtung:

- Seit einiger Zeit hat Oracle die Java-Lizenzen angepasst. Auf der Download-Seite von Oracle (s.o.) muss man sich deshalb jeweils persönlich registrieren und bestätigen, dass man sich an die Lizenz-Vereinbarung hält. (Bei kommerzieller Nutzung bestehen Einschränkungen!)
- Auch veralten die „offiziellen“ Java-Versionen von Oracle relativ schnell (Unterstützung bestimmter Versionen teilweise nur 6 Monate).
- Als **Alternative** ist deshalb die Java-Implementierung des **Open JDK Projektes** empfehlenswert. (Im GLI-Labor ist dies implementiert.) Allerdings ist diese Version nicht für „produktionstaugliche“ Anwendungen empfohlen.
    - Siehe https://openjdk.java.net/install/) für Installationshinweise.
    - Das Java 9 JDK kann z.B. unter https://jdk.java.net/java-se-ri/9 heruntergeladen werden. (oder https://jdk.java.net/17/ für Java 17).

Teil 2 – Einrichten des JDK: Damit Sie beim Aufruf des Java-Kommandos `javac` nicht jedesmal den gesamten Pfad angeben müssen (sonst findet Windows das Programm `javac.exe` nicht – dies ist leider standardmäßig nach der Installation des JDK der Fall, so auch auf den Fachbereichsrechern), sollten Sie den Suchpfad für Programme erweitern. Dazu müssen Sie die Umgebungsvariable PATH um den Ordner mit den Java-Programmen (`javac.exe` etc.) erweitern, je nach JDK-Version z.B.

```
C:\Programme\Java\jdk1.9.XYX\bin\
```

Teil 3 – Test: Laden Sie sich das Programm **`Obscure.java`** von der Webseite der Lehrveranstaltung.

Übersetzen und starten Sie das Programm mit dem JDK (durch direkten Aufruf des Java-Compilers).

Was tut dieses Programm?

Anmerkung: In nachfolgenden Übungsaufgaben können Sie auch eine integrierte Entwicklungsumgebung (siehe Aufgabe 3) verwenden.

Aufgabe 3: Umgang mit Eclipse – Java-Projekte erzeugen

Teil 1 – Installation: Falls Sie eine Entwicklungsumgebung wie z.B. Eclipse auf Ihrem eigenen Rechner ausführen wollen, installieren Sie die Entwicklungsumgebung auf Ihrem Rechner.

> **Anmerkung:** Statt Eclipse können Sie natürlich auch eine andere Entwicklungsumgebung (z.B. NetBeans, IntelliJ) verwenden.

Laden Sie daher von

> http://www.eclipse.org/downloads/

die **Eclipse IDE for Java Developers** (oder **für Java EE Developers** – die erweiterte Version für Java Enterprise) herunter. Achten Sie auf die passende Version (32 oder 64 bit)!

Nach dem Entpacken des zip-Files kann das Programm **eclipse.exe** direkt gestartet werden.

> **Hinweis:** Eclipse wird lediglich in einem beliebigen Ordner entpackt und kann von dort gestartet werden. Allerdings erfolgt dadurch auch keine Eintragung in der Registry und es wird kein Link im Programme-Menü bzw. auf dem Desktop erzeugt – diese müsste man bei Bedarf ggf. selbst selber anlegen.

Teil 2 – Projekt erzeugen: Erstellen Sie ein neues Java-Projekt *("File → New → Java Project").*

- Fügen Sie dem src-Ordner des Projektes (im default-Package) eine neue Java-Klasse hinzu. *("New → Class")*
- Sofern diese Klasse eine passende main-Methode besitzt, können Sie dieses Programm durch Rechtsklick mit der Maus auf die Java-Quelldatei mittels *"Run As → Java Application"* starten.

Sie können dafür z.B. die Datei Obscure.java (siehe Aufgabe 2) oder ein einfaches "Hello World"-Programm nehmen.

Aufgabe 4: Elementare Java Programmierung – Schiffe versenken

Schreiben Sie ein Programm, mit dem Sie gegen den Computer „Schiffe versenken“ spielen können.

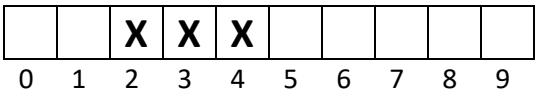
Ziel des Spiels: Versenken Sie alle Schiffe des Gegners mit möglichst wenigen Versuchen. Am Ende erhält der Spieler eine Bewertung, wie gut er war.

Da wir momentan noch keine Kommunikation verwenden können, um mehrere Programme gegeneinander antreten zu lassen, werden wir zunächst gegen den Computer spielen. Ferner müssen wir in diesem frühen Stadium des Semesters auf eine grafische Benutzeroberfläche verzichten, also arbeitet diese Version auf der Kommandozeile.

Setup: Wenn das Spielprogramm gestartet wird, platziert der Computer die Schiffe auf dem Spielfeld. Zur Vereinfachung beschränken wir uns zunächst auf ein eindimensionales Spielfeld, auf welchem das/die Schiffe fest platziert werden, d.h. das Spielfeld kann fest vorgegeben werden.

Falls Sie noch Zeit und Lust haben, können Sie Ihr Programm in Aufgabe 4 entsprechend erweitern.

Wie Sie spielen: Der Computer fordert Sie auf einen Tipp (eine Position bzw. Zelle) abzugeben, den Sie auf der Kommandozeile eingeben. Als Antwort auf Ihren Tipp sehen Sie auf der Kommandozeile als Ergebnis „Vorbei“, „Treffer“ oder „Versenkt“. Wenn das Schiff (bzw. alle Schiffe) versenkt sind, wird ihre Bewertung, d.h. die Anzahl ihrer Versuche ausgegeben.



Teil 1: Erstellen Sie eine Klasse **`SchiffeVersenken1D`**. Erstellen Sie (in dieser Klasse) die folgenden Methoden:

- **`String toString(boolean[]spielFeld)`** liefert eine textuelle Darstellung des Spielfeldes. Diese Funktion kann u.a. zum Testen verwendet werden, um sich die (noch nicht versenkten) Schiffe anzuzeigen.
  - Überlegen Sie sich, wie Sie das Wasser und wie die Schiffe repräsentieren wollen.

- **`int eingabeTipp()`** liest ihren Rateversuch, d.h. eine mögliche Position, von der Kommandozeile ein.

- **`void run()`** implementiert schließlich das Spiel:
  - Zunächst wird das Spielfeld als festes Array erzeugt, d.h. das Schiff auf dem Spielfeld platziert.
  - Im eigentlichen Spielablauf wird der Benutzer jeweils zur Eingabe eines Rateversuches aufgefordert, bis das Schiff versenkt ist.

Fügen Sie schließlich die folgende **`main`**-Funktion zu ihrer Klasse hinzu: (Anmerkung: Später werden wir lernen, warum wir nicht einfach alles direkt in der main-Funktion ausführen können.)

```
public static void main(String[] args) {
     SchiffeVersenken1D spiel = new SchiffeVersenken1D();
     spiel.run();
}
```

Teil 2: Testen Sie Ihr Programm.

Zusatz-Aufgabe 5 (Nicht ausgelastet?)

Erweiterung1 : Erweitern Sie Ihr Programm um die Möglichkeit, ein Schiff zufällig vom Computer auf dem Spielfeld platzieren zu lassen, indem Sie die folgende **`setup`**-Methode erstellen:

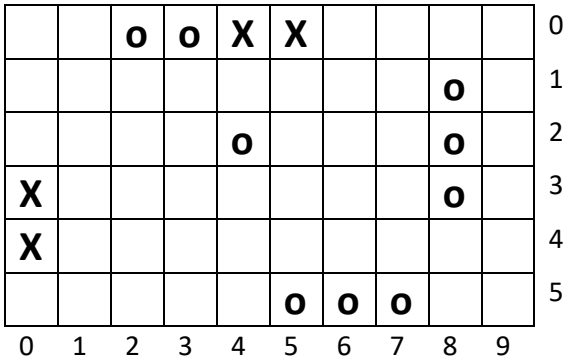
- **`boolean[] setup()`** erzeugt ein (eindimensionales) Spielfeld der Größe 10, auf dem an zufälliger Position ein Schiff der Größe 3 positioniert ist.
  Erstellen Sie dazu die folgenden Hilfsfunktionen:
  - **`boolean[] erzeugeSpielfeld(int feldGroesse)`** erzeugt ein eindimensionales leeres Spielfeld der angegeben Größe (ohne Schiffe).
    Hinweise:
    - Ein neues Feld können Sie mit **`new boolean[<Länge>]`** erzeugen.
    - Überlegen Sie sich, wie Sie das Wasser und wie die Schiffe repräsentieren wollen.
  - **`boolean setSchiff(boolean spielFeld[], int pos, int schiffsGroesse)`** positioniert ein Schiff bestehend aus **`schiffsGroesse`** Feldern ab Position **`pos`** auf dem Spielfeld.
    Falls dies nicht möglich ist (wann tritt hier ein Fehler auf?), soll das Spielfeld unverändert bleiben und **`false`** als Fehlercode zurückgeliefert werden.

Erweiterung2: Erweitern Sie Ihr Programm um die Möglichkeit, mehrere Schiffe mit unterschiedlicher Größe zu platzieren.

Bei dieser Gelegenheit bietet es sich auch gleich an, das Spielfeld nicht als ein Feld von `boolean`-Werten zu verwalten, sondern einen expliziten Aufzählungstyp zu verwenden:

```
enum Feld {WASSER, SCHIFF, TREFFER};
```

Immer noch nicht ausgelastet? Wie wäre es, wenn Sie statt eines eindimensionalen Spielfelds ein zweidimensionales Spielfeld verwenden? Für die geratene Position müssten Sie dann jeweils Zeile und Spalte einlesen (oder könnte man das auch anders lösen?)



(Beispieldarstellung: Leer = Wasser, **o** = Schiff (noch nicht getroffen), **X** = Treffer)

Zusatz-Aufgabe 6 (für Java-Experten): Java-Rätsel: Java-Typen

Falls Sie noch Zeit und Lust haben, können Sie auch noch die folgende Aufgabe bearbeiten. Die Java-Rätsel auf diesem und folgenden Aufgabenblättern sind für diejenigen unter Ihnen gedacht, welche gerne anspruchsvolle Aufgaben lösen wollen, mit denen Sie noch tiefer in die Feinheiten von Java einsteigen können.

Hinweise:

- Bei den Java-Rätsel Aufgaben sollten Sie zunächst „im Kopf“ versuchen, das Programm zu analysieren, ohne es am Rechner einzugeben.

  Anmerkung: Wenn Sie verstehen, was das Programm voraussichtlich macht, haben Sie schon einmal den „normalen“ Stoff verstanden. Bei den Rätselaufgaben gibt es dann teilweise noch Besonderheiten, die zu unerwarteten Ergebnissen führen.

- Erst danach sollten Sie untersuchen, was Java mit dem Programm macht: Was ist die Ausgabe des Programms? Oder gibt es gar einen Compiler-Fehler?
- Versuchen Sie dann zu ergründen, warum das Programm es sich (vermutlich) anders verhält, als Sie ursprünglich gedacht haben.
- Bei Fragen wenden Sie sich an den Dozenten (oder schauen Sie in der Beispiellösung nach).

a) Rätsel (Long-Dision): Was gibt das folgende Programm aus – und warum?

```
public class LongDivision {
    public static void main(String[] args) {
        final long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
        final long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
        System.out.println(MICROS_PER_DAY / MILLIS_PER_DAY);
    }
}
```

b) Rätsel (Casting): Was gibt das folgende Programm aus – und warum?

```
public class Multicast {
    public static void main(String[] args) {
        byte b = (byte) -1;
        char c = (char) b;
        int i = (int) c;
        System.out.println("i = " + i);
    }
}
```