Fragen, Antworten, Kommentare zur aktuellen Vorlesung

Es gibt ein Video zur Aufgabe mit der dynamischen Polymorphie (Rahmen):
https://youtu.be/sZeT6XygNLU

Zur Klasse Viereck und der Aufgabe gibt es das folgende Video: https://youtu.be/zoou_8OX0YU. Wie immer gilt, beachten Sie, dass es viele andere auch sehr gute Lösungen geben kann. Wird z. B. nach einem Rechteck mit bestimmten Eigenschaften gesucht, reicht es oft, die Seitenvarianten A-B und C-D bzw. B-C und A-D zu betrachten (was zum länglichen if wird und so ohne Schleife geht).

Zum Game of Life (Blatt 12) habe ich ein Video online gestellt: https://youtu.be/x4nR0n9NzcQ. Es werden einige Ansätze diskutiert, Fehler gemacht und korrigiert sowie die optionale Aufgabe (teilweise) gelöst.

Frage: Müssen wir in der Klausur Exceptions kennen?

Antwort: Die Antwort steht im vorherigen Fragen-und-Antworten-Dokument, ja. Der Tipp ist immer sich das Aufgabenblatt 12 anzusehen. Die letzte Aufgabe aus der Probeklausur könnte durch etwas mit Exceptions ersetzt werden, ist aber nur eine von vielen Möglichkeiten und vielleicht kommen Exceptions nicht vor.

Frage: Müssen wir die Klassen Interaktionsbrett und EinUndAusgabe auswendig können?

Antwort: Das Interaktionsbrett wird in der Klausur nicht vorkommen. Von der Klasse EinUndAusgabe sollten Sie die Methode ausgeben() kennen.

Hinweis: Bisher haben wir für eine Sammlung mit beliebig vielen Objekten einer Klasse K immer die `ArrayList<K>` genutzt. Ein Ansatz der prinzipiell in Ordnung ist, aber noch viele Optimierungsmöglichkeiten hat. Hier wird nicht der Fall betrachtet, dass eine Liste mit den Eigenschaften, dass alle Werte eine Reihenfolge haben und jedes Objekt mehrfach vorkommen kann, genauer betrachtet, was zu anderen Arten von Sammlungsklassen führt. Hier soll das Detail betrachtet werden, dass immer ein Interesse nach der einfachsten und trotzdem allgemeinsten Lösung besteht. Im Fall der `ArrayList` kann festgestellt werden, dass unser Interesse nur daran besteht, dass die Elemente in einer Reihenfolge eingefügt, durchlaufen und auch gelöscht werden können. Diese allgemeinen Eigenschaften sind in Java im Interface `List` zusammengefasst. Zu `List` gibt es verschiedene Implementierungen und können natürlich weitere ergänzt werden. Da die Art der Liste für uns bisher egal war, ist an jeder Stelle, an der ein Typ anzugeben ist, nicht `ArrayList` sondern `List` anzugeben (Variablendeklarationen, Parameterlisten), da so die genaue Art der Liste später geändert werden kann. Konkret sollten Sie z. B. ab jetzt statt

```
ArrayList<Hund> hunde = new ArrayList<Hund>();
```

genauer folgendes schreiben.

```
List<Hund> hunde = new ArrayList<>();    // import java.util.List;
```

Auf der rechten Seite kann Hund weggelassen werden, das ist aber nur eine abkürzende Schreibweise und hat sonst keine Bedeutung. Diese Vereinfachung dürfen Sie natürlich in der Klausur nutzen, dies wird aber nicht vorausgesetzt. Generell bietet Java mittlerweile einige abkürzende Schreibweisen, die allerdings das Grundkonzept nicht verändern. Sollten Sie über das Schlüsselwort `var` stolpern, können Sie es bewusst ignorieren, da es zu einem unsauberen Programmierstil führt, da es zwar weniger zu tippen gibt, es für später den Code lesende Personen (und um die geht es fast ausschließlich in der Programmierung) aber die Lesbarkeit verringert.

Hinweis: Java bietet einige weitere Sprachkonstrukte, die jenseits der Einführung in die Programmierung sind. Zwei Beispiele zeigt der folgende Code.

```
public class Main {
  public static void main(String[] s){
    var x = 42;
    var y = x + 1;
    System.out.println(y);
    var z = y = x = 42;
    z = (y = (x = 42));
    int[] array = {21, 28, 35, 42, 49};
    for(var pos = 0; (z = array[pos]) < 42; pos ++){
      System.out.println(z);
    }
  }
}
```

Die zugehörige Ausgabe lautet:

```
43
21
28
35
```

Das Beispiel zeigt, dass der Typ bei Variablen nicht explizit angegeben werden muss, wenn der Compiler auf den Typen schließen kann. Das ist für Basistypen und Klassen möglich. Statt des Typen wird das Schlüsselwort `var` genutzt. Generell gibt es einige Programmiersprachen, die ohne explizite Typangaben auskommen, was Vor- und Nachteile hat. Leider wird `var` gerne zur schlechten Programmierung genutzt, um das Tippen langer Klassennamen zu verhindern. Dieser Code wird dadurch unwartbar und ist schlecht für Projekte, was leider erst bei der nächsten Bearbeitung auffällt. Ein Kompromiss ist der auch in ungetypten Sprachen genutzte Ansatz den Typen in den Variablennamen zu integrieren, z. B. `xInt`. Dies erhöht die Tipparbeit, macht Programme aber lesbarer, da die Information zusätzlich direkt erkennbar ist.

Weiter zeigt das Beispiel das eine Zuweisung ein Ausdruck ist, also zu einem Wert, genauer dem zugewiesenen Wert, ausgewertet werden kann. Dabei muss eine Zuweisung `x=42` vom Zuweisungsbefehlt `x=42;` unterschieden werden, da Befehle keine Ausdrücke sind. Die gezeigten Klammern deuten die Auswertungsreihenfolge an, die ohne Klammern die gleiche ist. Die Schleife zeigt eine Nutzungsmöglichkeit einen Wert zuzuweisen und diesen Wert in einer Prüfung zu nutzen. Der Ansatz erhöht die Lesbarkeit nicht.