Grundlagen der Informatik 1
WS 2008/09

Prof. Mühlhäuser, Dr. Rößling, Melanie Hartmann, Daniel Schreiber
http://proffs.tk.informatik.tu-darmstadt.de/gdi1

Übung 12 Version: 1.1 26.01.2009

---

1 Mini Quiz

☐ Die Reihenfolge der catch-Blöcke ist nicht von Bedeutung.

☐ Ein Compiler kann alle Arten von Fehlern (bis auf Laufzeitfehler) entdecken und signalisieren.

☐ Die beim Ausführen einer Methode auftretenden Ausnahmen können im catch Block behandelt werden.

☐ Fehler die während der Laufzeit des Programms auftreten müssen behandelt werden.

☐ Ausnahmen müssen unmittelbar in der Methode behandelt werden, in der sie auftreten.

☐ Man kann leicht nachweisen, dass ein Programm fehlerfrei ist.

2 Verständnis

1. Welche Gemeinsamkeiten besitzen die Ausnahmeklassen Exception und Error? Was unterscheidet sie? Welche von beiden sollte man abfangen?

2. Erklären Sie, was Syntax-, Laufzeit-, Intentions- und lexikalische Fehler sind. Welche dieser Fehler werden vom Compiler erkannt?

3. Wie hängen die Begriffe Throwable, throws und throw zusammen? Wo sollte man was verwenden?

4. Ist es sinnvoll, eigene Ausnahmeklassen zu definieren? Welche Vorteile ergeben sich hieraus?

3 Fehlersuche

1. Betrachten Sie folgenden Java-Code. Beheben Sie alle syntaktischen Fehler.

```
1 public static int[] reverseArray (int[] source) throw Exception {
2   int length = source.length();
3   int[] inverted;
4   int i=0;
5
6   try {
7     inverted = new int[length] ;
8
9     while (i < length){
```

```
10          inverted[i] = source[i];
11          i++;
12        }
13      }
14      catch (Exception) {
15        System.out.println("Caught Exception...");
16      }
17      catch (IndexOutOfBoundsException e) {
18        System.out.println("Caught Exception: " + e);
19      }
20      final {
21        inverted = new int[length()];
22        inverted = source ;
23      }
24      return inverted;
25  }
```

2. Was passiert generell beim Aufruf der Funktion reverseArray?

   Die Funktion wird von Java nicht kompiliert - auch **ohne** vorhandene syntaktische Fehler, weswegen? Wie können Sie die Funktion dennoch kompilieren, was müsste man dazu beheben? **Hinweis**: Betrachten Sie bitte nochmal die Mini Quiz Fragen...

3. Suchen und beheben Sie die vorhandenen Intentionsfehler (Fehlerquelle 'Mensch').

4 Funktionen höherer Ordnung in Java

Erinnern Sie sich noch an die Funktion fold aus dem Scheme-Teil der Vorlesung? Zur Erinnerung, der Vertrag lautet:

```
1 ;; fold : (X Y -> Y) Y -> ((list of X) -> Y)
```

Machen Sie sich nochmal kurz klar, welche Aufgabe die folgende Variante von fold erfüllt.

```
1 (define (fold f n)
2  (lambda (x)
3    (if (empty? x)
4      n
5      (f (first x) ((fold f n) (rest x ))))))
```

Wir wollen nun eine ähnliche Funktion in Java implementieren. Zunächst stellen wir fest, dass eine direkte Übersetzung nach Java nicht möglich ist, da Java keine Funktionen höherer Ordnung unterstützt. Mit Hilfe eines Interfaces kann jedoch ein solches Verhalten simuliert werden.

1. Deklarieren Sie ein Interface namens BinaryOp, das eine binäre Operation `apply` über ganzen Zahlen deklariert. Sowohl Argumente als auch der Rückgabewert sollen vom Typ int sein.

2. Schreiben Sie jetzt die Klassen Adder und Multiplier, die das Interface BinaryOp implementieren und in der Methode `apply` die Addition bzw. Multiplikation zur Verfügung stellen.

3. Implementieren Sie nun eine Methode mit folgender Signatur:

   ```
   1 int jfold(int[] a, BinaryOp op, int s)
   ```

   Diese Methode soll analog zu fold die ihr übergebene Operation *op* schrittweise auf alle Elemente des Arrays *a* anwenden. Dabei entspricht *s* dem Argument *n* in fold.

4. Erklären Sie nun, worin sich jfold und fold unterscheiden.

5 Polymorphie

Betrachten Sie folgenden Java Code:

```
1  interface MyInterface {
2    public int g();
3  }
4
5  class Base implements MyInterface {
6    public int i = 1;
7    public int g() { return f(); }
8    protected int f() { return i; }
9  }
10
11 class Derived extends Base {
12   public int i = 2;
13   protected int f() { return -i; }
14   protected int h() { return i + i; }
15   protected int p() { return i * i; }
16 }
17
18 public class Client {
19   public static void main (String args[]){
20     Base base = new Base() ;
21     Derived derived = new Derived();
22     MyInterface restricted = derived;
23
24     int temp = base.i;      // a)
25     temp = base.g();        // b)
26     base = derived;
27     temp = base.i;          // c)
28     temp = base.g() ;       // d)
29     temp = restricted.g(); // e)
30     restricted = base;
31     temp = restricted.g(); // f)
32   }
33 }
```

1. Welchen statischen und dynamischen Typ hat die Variable base zum Zeitpunkt a) bzw. c)?
2. Welchen Wert hat die Variable **temp** zu den Zeitpunkten a), b), c), d) e) und f)?
3. Betrachten Sie nun die folgenden Aufrufe:

```
1  Base base = new Base();
2  Derived derived = new Derived();
3  int temp = base.g();
4  derived = base;
5  base = new Derived();
6  MyInterface t = new Derived();
7  t = base;
8  temp = t.f();
9  derived = t;
10 base = t;
11 t = derived;
```

```
12 temp = t.p();
13 temp = t.g();
14 temp = derived.p();
```

Welche Aufrufe sind erlaubt, welche werfen einen Fehler? Treten die Fehler zur Laufzeit oder bereits beim Kompilieren auf? Es gibt keine Folgefehler - kommentieren Sie verbotene Aufrufe aus und ignorieren Sie diese bei der weiteren Fehlersuche.

Hausübung

Die Vorlagen für die Bearbeitung werden im GdI1-Portal bereitgestellt. Kommentieren Sie Ihren selbst erstellten Code. Die Hausübung muss bis zum Abgabedatum im GdI1-Portal abgegeben werden. Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe Ihrer Hausübung bestätigen Sie, dass Sie bzw. Ihre Gruppe alleiniger Autor des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen Quellen deutlich zitieren.

Abgabe: Bis spätestens Fr, 06.02.09, 16:00 Uhr.

6 Geografische Welten (10 Punkte)

Viele geografische Anwendungen bedienen sich heutzutage verschiedener Tricks aus der Informatik, woraus sich auch ein eigener Zweig entwickelt hat, die Geoinformatik. Wir schauen uns in dieser Hausübung eine stark vereinfachte Praxisanwendung aus diesem Gebiet an, in der es darum geht, die Küsten der Niederlande näher unter die Lupe zu nehmen.
In den Niederlanden gibt es immer wieder Sturmfluten, die das Land wegen seiner geringen Höhe überschwemmen. So hat die letzte große Sturmflut von 1953 fast ganz Zeeland (im Süden des Landes) verwüstet. Um sich gegen diese Naturgewalt zu schützen, behelfen sich die Küstenbewohner mithilfe von Deichen entlang der Nordseeküste. In dieser Aufgabe wollen wir eine solche Sturmflut auf einem gegebenen Gebiet simulieren.

6.1 Grundstruktur (0.5P)

Erstellen Sie zunächst die Klasse `GeoSimulation`. Diese soll als interne Datenstruktur area ein `char[][]` verwenden; diese Datenstruktur repräsentiert eine Karte des Gebiets. Eine korrekte Karte ist ein rechteckiges Gebiet, wobei jeder Teil durch eines von drei Symbolen repräsentiert wird: '#' für ein Stück Deich, '0' für ein Stück Land und '~' für ein Wassergebiet. Implementieren Sie die folgenden Methoden der Klasse `GeoSimulation`:

- Konstruktor GeoSimulation(**char**[][] area)
- Getter und Setter Methoden für die interne area Datenstruktur
- Eine Methode toString(), die eine Stringrepräsentation der Area ausgibt. Eine solche Repräsentation kann z.B. folgendermaßen aussehen:

```
1 ~~~~~#00000
2 ~~~~#000000
3 ~~~#0000000
4 ~~~~#000000
```

6.2 Gültige Karten (2P)

Damit eine Karte gültig ist, müssen mindestens ein Deich, ein Stück Land und Wasser vorhanden sein. Schreiben Sie dazu die Methoden **boolean** hasLeveeElement() (*levee* ist die amerikanische Bezeichnung für einen Deich), **boolean** hasGroundElement() und **boolean** hasWaterElement(). Sobald eine neue Area angegeben wird (d.h. im Konstruktor oder durch setArea()) soll geprüft werden, ob die Karte gültig ist und ansonsten eine entsprechende Exception geworfen werden (NoLeveeElementException, NoGroundElementException bzw. NoWaterElementException). Diese Exceptions sollen von der Exception InvalidAreaException erben.

6.3 Deichbruch (3P)

Implementieren Sie die Methode **void** floodArea(**int** x, **int** y), die die Koordinaten eines möglichen Deichbruchs entgegennimmt und berechnet, welche Teile des Gebiets danach unter Wasser stehen. Dazu sollen alle Landelemente ('0') der aktuellen Karte, die von der Überschwemmung betroffen sind, durch Wasserelemente ('~') ersetzt werden. Dabei erfolgt die Überschwemmung nur horizontal und vertikal, ausschließlich diagonal angrenzende Elemente sind nicht betroffen. Die Koordinaten entsprechen dabei den Zählern des Arrays, d.h. mit den Koordinaten (x,y) ist das Element area[x][y] gemeint.
Hinweis: Sie können sich beim Lösen am Floodfill-Algorithmus (siehe `http://de.wikipedia.org/wiki/Floodfill`) orientieren.
Beispiel: floodArea(4,2)

```
1  0#000#              0#~~~#
2  0#00#~              0#~~#~
3  #000#~      -->     #~~~~~
4  000#~~              ~~~#~~
5  000#~~              ~~~#~~
```

floodArea soll zwei verschiedene Exceptions auslösen können:

- InvalidCoordinatesException , wenn die angegebenen Koordinaten nicht gültig sind, d.h. außerhalb der Karte liegen.
- NoLeveeException, wenn die angegebenen Koordinaten nicht auf ein Deichstück zeigen.

6.4 Gültige Karten II (2.5P)

Ein weitere Bedingung für die Gültigkeit der Karte ist, dass darauf ein dichter Deich dargestellt ist. Schreiben Sie dazu die Methode **boolean** hasLeakProofLevee(), die genau dies bietet. Sie dürfen dabei davon ausgehen, dass alle Wasserelemente zusammenhängen, es also nicht zwei getrennte Seen oder Meere gibt. Ergänzen Sie den Konstruktor und setArea um diese Überprüfung, und lösen Sie gegebenenfalls eine LeakyLeveeException aus, die auch von InvalidAreaException erbt.
Hinweis: Sie können zur Lösung dieser Aufgabe einen ähnlichen Ansatz wie in der vorigen Aufgabe verwenden...

6.5 Benutzerinteraktion (2P)

Schreiben Sie eine Klasse GeoDialog (das Gerüst wird im Portal bereitgestellt), die den Benutzer mit dem GeoSimulator interagieren lässt. Die Klasse soll die Area mit vorgegebenen Werten initialisieren und den Benutzer über eine InvalidAreaException mit einer passenden Fehlermeldung informieren, außer

es handelt sich um eine LeakyLeveeException. Danach soll die Klasse das aktuelle Gebiet anszeigen, vom Benutzer die x- und y-Koordinaten eines möglichen Deichbruchs erfragen, diesen simulieren und das resultierende Gebiet anzeigen. Dabei auftretende Fehlermeldungen sollen dem Benutzer auch kurz mitgeteilt werden.