Kapitel I: Algorithmen, Assertions und Schleifen

Frage: Wozu dienen Assertions? Wann werden sie ausgeführt? Was passiert dann?

Assertion = sollen Annahmen über den Zustand eines Programms verifizieren und sicherstellen, dass diese eingehalten werden → Korrektheits-Check

z.B.: `assert x >= 0;`

Programm überprüft die Bedingung und fährt fort wenn sie erfüllt ist. Andernfalls wird eine Exception ausgelöst.

Frage (Assertions auswerten):
Gegeben ist folgendes Programmstück zur Berechnung der harmonischen Zahlen.

```
int n= ... ;
assert n >= 0; (n darf nicht negativ sein)
float sum=0;
int i=1;
while (i <= n )
       {
       assert _____1_____
       sum = sum + 1.0/i;
       assert _____2_____
       i ++;
       assert _____3_____
       }
assert _____4____
```

Zur Erinnerung: Die harmonischen Zahlen sind definiert als:
h(i) == 1+1/2+1/3+...1/i
h(0) == 0
Kreuzen Sie in folgender Tabelle an, welche Assertions an welcher Stelle des Programms gelten:

1	2	3	4	Assertion
	✔			**sum <= n**
	✔			**sum >= 0**
	✔			**sum > 0**
✔				**sum == 0**
	✔			**sum != 0**
	✔	✔		**sum == h(i)**
				sum == h(i-1)
				sum == h(n)
				i >= 0
✔		✔		**i >= 1**
✔		✔		**i <=n**
				i < n
✔		✔		**i < n+1**
			✔	**i <= n+1**
			✔	**i > n**
			✔	**i == n+1**

Frage (Schleifeninvarinte aufrechterhalten):

```
int n = ...;
int i = 0;
int m = ?1?;
while (i < n) {
       assert m == n * i;
       m = m + n;
       ?2?;
       }
assert ?3?
```

?1?: Wie muss man m initialisieren, damit die Schleifeninvariante gilt?

→ muss 0 sein, da i beim ersten Durchlauf 0 ist

?2?: Welches Statement fehlt, um die Schleifeninvariante aufrechtzuerhalten?

i++;

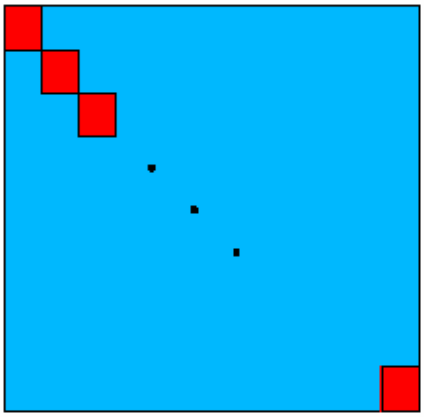
?3?: Welche Assertion gilt hier?

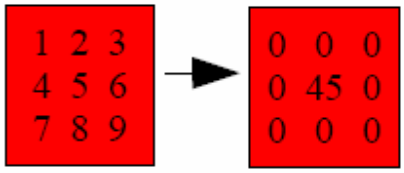
i == n

Frage (Matrix bearbeiten):
Schreiben Sie eine Methode (Funktion)

```
void sum3x3(int m [][]){...}
```

Eingabe: Die zweidimensionale Matrix m.
Zusicherung: m ist quadratisch (heisst: m hat gleich viele Spalten wie Zeilen). Außerdem ist die Anzahl der Spalten bzw. Zeilen durch 3 teilbar.
Gefordert: Entlang der Diagonale sollen jeweils 3x3 Matrix-Elemente zusammengefasst werden. Die Summe der 9 Zahlen kommt in die Mitte, die anderen 9 Elemente werden 0 gesetzt.
Fügen Sie Assertions in ihr Programm ein, die auch umgangssprachlich (als Kommentare) formuliert sein dürfen.

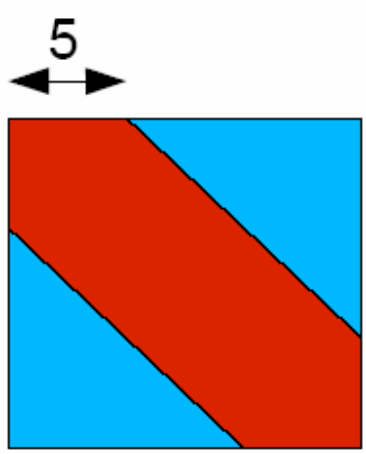




```
public class Matrix {

	final int l = 9;

	public static void main(String[] args){
		new Matrix();
	}

	public Matrix(){
		int c = 1;
		assert l%3 == 0;
		int [][] m  = new int [l][l];
		for(int j = 0; j<l; j++){
			for(int i = 0; i<l; i++){
				m[i][j]=c;
				c++;
				System.out.print(m[i][j]+",");
			}
			System.out.println();
		}
		sum3x3();
		ausgabe();
	}

	public void sum3x3(){
		int k=0,j=0,i=0,n=0;
		int[] summe = new int[l/3];

		for (n = 0; n < l/3; n++){
			for (i = k; i < k+3; i++){
				assert i < l;
				for (j =k; j < k+3; j++){
					assert j < l;
					summe[n] = summe[n] + m[j][i];

					assert summe[n] >= m[j][i];
					m[j][i]=0;
				}
			}
			m[j-2][i-2] = summe[n];
			k = k+3;
			assert k <= l;
		}
	}

	public void ausgabe(){
		int c = 1;
		for(int j = 0; j<l; j++){
			for(int i = 0; i<l; i++){
				System.err.print(m[i][j]+",");
			}
			System.err.println();
		}
	}
}
```

Frage (Matrix bearbeiten):
Schreiben Sie eine Methode (Funktion)
`int sum5x5(int m [][]){...}`

Eingabe: Die zweidimensionale Matrix m. Zusicherung: m ist quadratisch (heisst: m hat gleich viele Spalten wie Zeilen).
Gefordert: Die Funktion sum5x5 soll die Summe aller Elemente im roten Bereich liefern:



Frage (Assertions finden):

```
int n = ...;
assert n >= 0;
int i = 0;
int m = 0;
while (i != n) {
      assert m == n * i;
      m = m + 2 * n;
      i = i + 2;
      }
assert ???;
```

a) Stimmt die Schleifeninvariante?
Endlosschleife, wenn n eine ungerade Zahl ist, wird i niemals zu n → die Bedingung *i != n* gilt immer!

`(assert m == n*i;` **wird immer erfüllt)**

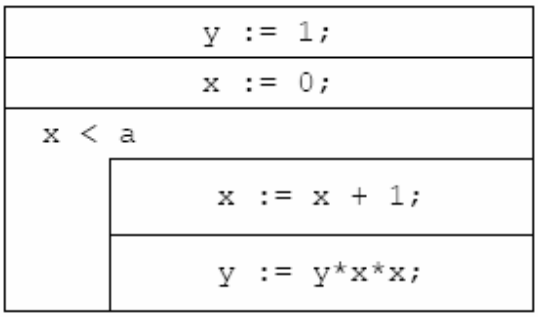
b) Welche Assertion gilt nach der Schleife (an der Stelle ???).
assert i == n;

c) Ist das Programm partiell korrekt? Terminiert es? Immer? Manchmal (in welchen Fällen)? Nie?
Terminiert nur wenn n eine gerade Zahl ist

d) Gibt es eine Assertion, die man anstelle von assert n>=0; einsetzen kann, sodass das Programmstück danach total korrekte ist?
z.B.: assert n%2 == 0;

Frage (Übersetzung eines Struktogramms in ein Codestück):

Gegeben ist folgendes Struktogramm:



Schreiben Sie ein entsprechendes Programm in Java, das a als Eingabe übernimmt (a>=0) und y retourniert.

```
public class Strukt {

	public static void main(String[] args){
		new Strukt();
	}

	public Strukt(){
		ret(3);
	}

	public int ret(int a){
		assert a >= 0 ;
		int x = 0 ; y = 1 ;

		while(x<a){
			x++;
			y = y*x*x;
			System.out.println(y);
		}
		return y;
	}
}
```

Frage: Kann man einen Verifikator bauen? Ein Verifikator ist ein Programm, das als Eingabe ein Computerprogramm P, eine Vorbedingung V und eine Nachbedingung N bekommt, und als Output true/false liefert, je nachdem ob P total korrekt ist in Bezug auf V und N (also true liefert genau dann wenn {V}P{N} total korrekt ist).

- kann gebaut werden, unter der Voraussetzung, dass P total korrekt ist (keine Endlosschleife, etc)
- V wird auf jeden Fall ausgeführt, N nur wenn P innerhalb einer endlichen Zeit terminiert

Kapitel II: Rekursion

Frage (rekursives Programm analysieren):

Gegeben ist folgendes rekursives Programm:

```
int theFunc(int x) {
      assert x>=0;
      if (x == 0) return 0;
      else return x * x + theFunc(x-1);
}
```

Beschreiben Sie, was es berechnet (entweder mit einer kurzen Formel oder umgangssprachlich).

angenommen x=5:

(5*5+(4*4+(3*3+(2*2+(1*1+(0)))))) → 55

Methode ruft sich immer wieder selbst auf und gibt Returnwert von `x * x + theFunc(x-1);` als Input zurück, bis Rückgabewert 0 ist

Frage (rekursives Programmstück schreiben):

Gegeben ist folgende Formel:

$$k(n)=\begin{cases} k(n-1)+\frac{1}{n^2} \text{ falls } n>0 \\ 0 \text{ falls } n=0 \end{cases}$$

Schreiben Sie eine rekursive Methode, die k(n) berechnet.

Frage (Anzahl der Rekursionsaufrufe):

Die Fibonacci-Zahlen werden durch folgende rekursive Prozedur berechnet:

```
int count = 0;
...
int fib(int n) {
      assert n >= 0;
      count ++;
      if (n == 0 || n == 1) return 1;
      else return fib(n-1) + fib(n-2);
}
```

In der globalen Variable count wird die Anzahl der Aufrufe von fib mitgezählt. Wie gross ist der Wert von count nach einem Aufruf von fib(3) oder von fib(4) oder von fib(5)?

count = fib(n)*2 - 1

Frage (Partielle Korrektheit, Termination von Rekursionen):

Gegeben ist folgendes Programmstück

```
boolean istGerade(int i) {
      if (i==0) return true; else return istUngerade(i+1);
      }

boolean istUngerade(int i) {
      if (i==1) return true; else return istGerade(i+1);
}
```

Das Funktionspaar soll berechnen, ob die Eingabe i >=0 eine gerade bzw. ungerade Zahl ist.

Ist die Implementierung partiell korrekt? Terminiert sie? Immer? Nie? Auf welchen Werten?

- ist partiell korrekt
- terminiert nur wenn i ein negativer Wert ist, da i hochgezaehlt wird kann es andernfalls niemals 0 werden → bricht niemals ab

Frage (Rekursion, Abbruchbedingung):

Folgende Funktion soll in einem String alle “a” durch “X” ersetzen.

```
String replaceA(String s) {
      if (s.charAt(0) == 'x')       //ehemals XXX
      return true;                  //ehemals YYY
      else
      if (s.charAt(0) == 'a')
return 'X' + replaceA(s.substring(1,s.length()));
else
return s.charAt(0) + replaceA(s.substring(1,s.length()));
}
```

Ergänzen Sie die Passagen XXX und YYY.

Kapitel III: Komplexität, Sortieren

Frage: Was bedeutet "quadratischer Aufwand."?

Rechenzeitaufwand eines Algos in Abhängigkeit von n^2
(Ordnung des höchsten Terms)

Frage: Ein Algorithmus A hat die Komplexität O(n^2), ein anderer Algorithmus B hat die Komplexität O(n log n). Welcher Algorithmus ist schneller? Ist das immer so? Was kann über die Performance der Algorithmen A und B gesagt werden?

O(n log n) ist schneller, quasilinearer Aufwand (z.B.: Sortierung)
→ da die Berechnung n als Input hat, gilt das immer

$O(n^2)$ nimmt mehr Zeit u Speicher in Anspruch als O(n log n)

Frage: Ordnen Sie die folgenden Komplexitätsklassen aufsteigend nach ihrer Komplexität: Linear, Quadratisch, Polynomiell, Exponentiell

1. linearer Aufwand
2. quadratischer Aufwand
3. polynominaler Aufwand
4. exponentieller Aufwand

Frage: Beschreiben Sie umgangssprachlich das Sortierverfahren Bubblesort? Welche Komplexität hat das Verfahren? Ist das für alle möglichen Inputs der Fall?

Verfahren: benachbarte Elemente werden immer ausgetauscht, so wandern die größeren Elemente (wie Blasen…*wow*)nach einer gewissen Anzahl von Durchläufen nach oben, solange bis ab- oder aufsteigend sortiert ist

Komplexität: $O(n^2)$

Frage: Beschreiben Sie umgangssprachlich das Sortierverfahren Quicksort? Welche Komplexität hat das Verfahren? Ist das für alle möglichen Inputs der Fall? Wann ist das Verhalten von Quicksort schlechter? Wie hoch ist der Aufwand dann?

Verfahren: Elemente werden in 2 Teile geteilt; große Elemente auf die eine, kleine auf die andere Seite; die 2 Teile werden wieder und wieder geteilt, Elemente werden wieder der Größe nach links oder rechts gespeichert…solange bis ab- oder aufsteigend durchsortiert

Komplexität: O(n log n)…optimaler Aufwand → kleiner/größer-Prinzip

Schlechter wenn: die Trennelemente immer das Größte oder Kleinste aller Elemente sind

Frage: Beschreiben Sie umgangssprachlich das Sortierverfahren Heapsort (ganz grob in drei Sätzen. Die genaue Beschreibung der Vorgänge im Heap ist nicht erforderlich)? Welche Komplexität hat das Verfahren? Ist das für alle möglichen Inputs der Fall?

Verfahren: Baumstruktur, jeder Knoten hat unter sich nur wertniedere; Rekursion: solange in einem unteren Element ein höherer Wert steht, wird ausgetauscht, solange bis der größte Wert ganz nach oben gewandert ist.

in der Praxis: Elemente sind in Array gespeichert, jeder Knoten i findet seinen linken Nachfolger bei: 2i+1 und seinen rechten bei: 2i+2, diese haben ihren Vorgänger bei (i-1)/2

Komplexität: O(n log(n))

Kapitel IV: Threads

Frage: Wozu dienen Threads? Was kann man damit programmieren/modellieren?

= Prozesse im Prozess, Programfragmente die parallel zu anderen Threads laufen können
Wozu: Nebenläufigkeit, System kann 2 oder mehrere Vorgänge gleichzeitig (quasi gleichzeitig) ausführen

Frage: Sei eine Klasse wie folgt implementiert:

```
public class MyThread extends Thread {
      public void run() {
            while (true) { ... }
      }
}
```

Im Hauptprogramm steht:

```
MyThread t = new MyThread();
```

Was ist der Unterschied zwischen einem Aufruf von t.start(); und t.run();
`t.start()`: Thread wird erzeugt und initialisiert, ruft dann selbst run() auf um den Code auszuführen
`t.run()`: erzeugt keinen neuen Thread, normaler Methodenaufruf im laufenden Thread, startet Thread-Body

Was passiert? Warum?

Was passiert bei der Sequenz t.run(); t.start();

Thread wird gestartet mit run, start erzeugt einen neuen

Was passiert bei der Sequenz t.start(); t.run();

start erzeugt neuen Thread und ruft selbst run auf

Frage: Sei eine Klasse wie folgt implementiert:

```
public class MyThread extends Thread {
      public void run() {
            int q = (int) Math.random() * 10;
            for (int i = 0; i < q; i++)
                  try { sleep(1000) } catch (InterruptedException e) {};
            }
      }
```

Im Hauptprogramm steht:

```
MyThread t = new MyThread();
for (int j = 0; i <= 100; i++)
{MyThread t = new MyThread(); t.start();}
```

Was passiert? Wieviele Threads werden gestartet?

t.start wird 101 mal aufgerufen → 101 Threads

Wieviele laufen nach 5 Sekunden (ca.? genau?)? Wieviele laufen nach 10 Sekunden?

- **abhängig davon, wie schnell alle Threads erzeugt werden bzw. wieder terminieren (Anzahl der Pausen abhängig von der Zufallszahl q)**
- **nach 5 Sekunden sollten alle Threads erzeugt sein, daher sollten ca 101 Threads laufen, abzüglich der Threads deren q < 5 war**

Frage: Wazu dient das Keyword synchronized? Was genau blockiert es?

Durch synchronized kann Methode oder Block innerhalb einer Methode geschützt werden. Wird eine mit *synchronized* beschriebene Methode (zB: public synchronized void doThat(){}) durchlaufen, sperrt Java das Objekt, zu dem die Methode gehört, für alle anderen synchronized Methoden.

Frage (Synchronisation):

Eine Klasse implementiert einen Timestamp:

```
public class TimeStamp {
      int minute; int second;

      public TimeStamp () {
            minute = 0; second = 0;
      }

      public void addSeconds(int p) {
            second = second + p;
            minute = minute + second / 60;
            second = second % 60;
      }
}
```

Welche Gefahr besteht, wenn meherer Threads mit TimeStamps arbeiten? Wie kann man dieses Risiko ausschließen?

Gefahr: mehrere Threads auf die Klasse TimeStamp zu und überschreiben immer wieder eben gesetzte Werte
Lösung: synchronized → nur ein Thread darf TimeStamp bearbeiten

Frage: Wozu dienen (ganz allgemein ausgedrückt) die Befehle wait() und notify()?

wait(): hält einen Thread an
notify(): reaktiviert einen wartenden Thread

Frage: Was bewirkt folgendes Codestück?

```
try { wait(); } catch (InterruptedException e) {}
?1?;
```

versucht aktuellen Thread anzuhalten

Was wird ausgeführt? Wird die Anweisung ?1? ausgeführt? Was muss passieren, damit ?1? ausgeführt wird?

try block wird ausgeführt, → wait() **zur Zeit tut sich gar nichts!**
?1? wird nicht ausgeführt, erst wenn notify() aufgerufen wird

Kapitel V: GUI

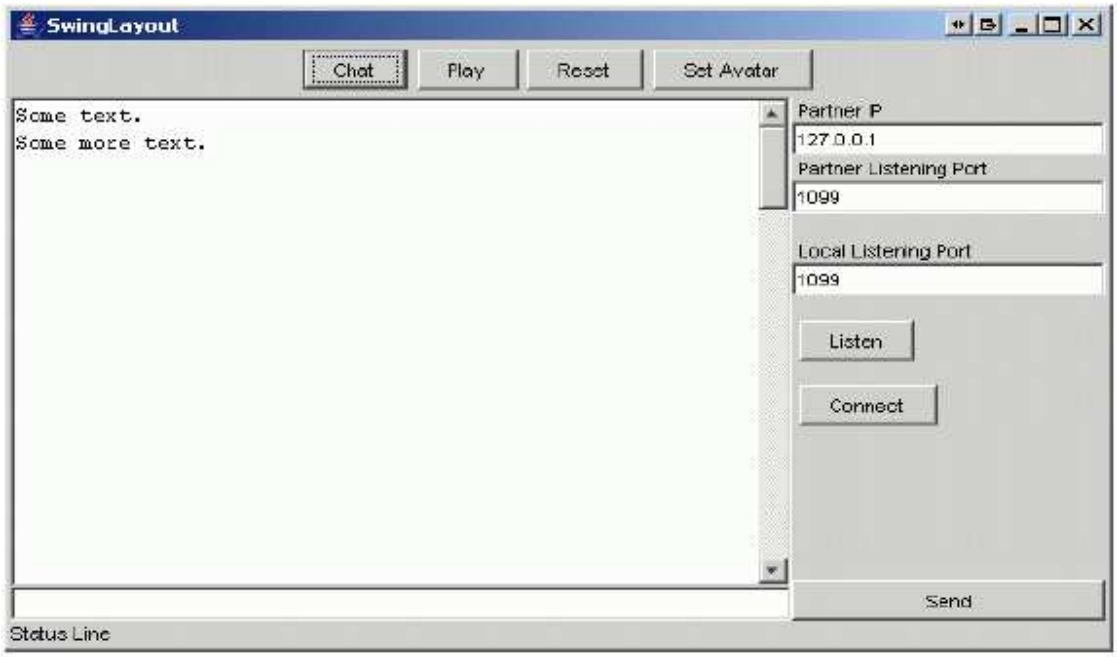
Frage (Aufbau eine Swing GUIs): Was sind Swing Komponenten, was sind Swing Container. Nennen Sie Beispiele für beide. Wie werden sie verwendet.

Komponenten: Klassen, die zur Gestaltung einer grafischen Oberfläche eingesetzt werden können; zB.: JPanel, JLabel, JFrame,...

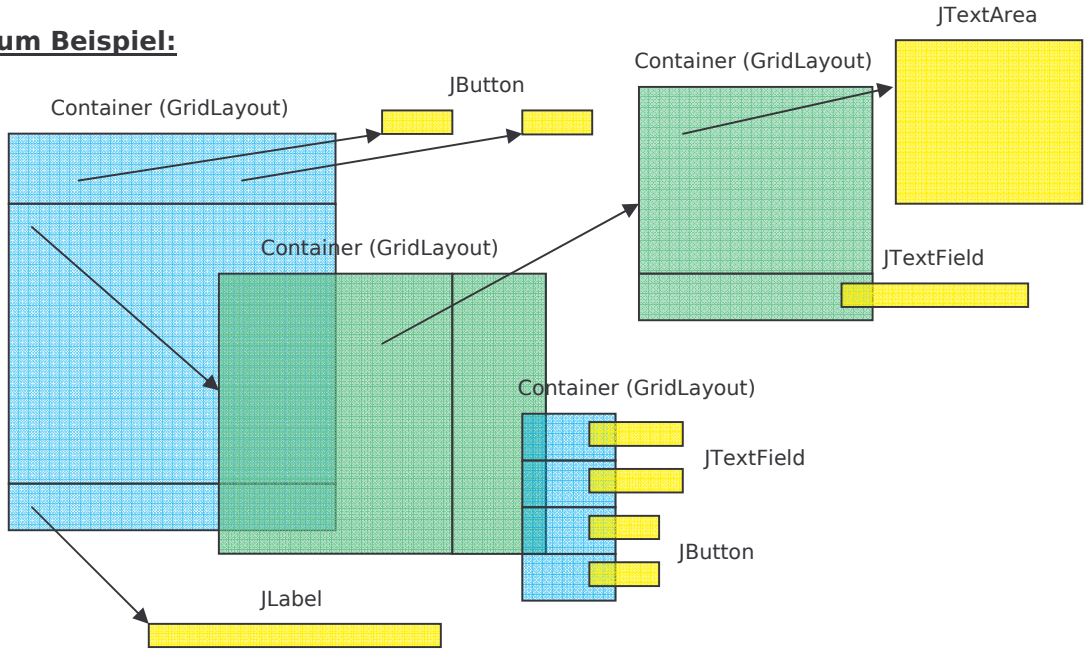
Container: stellt Methoden zur Verfügung um Komponenten hinzuzufügen oder zu entfernen, realisiert mit Layout-Manager Klassen die Anordnung/Positionierung der Komponenten

```
Container c = new Container();
c = getContentPane();    //Objekte warden beim Swing nicht direct dem Fenster, sondern
                         //dem ContentPane zugefügt, mit getContentPane() wird ein
                         //Container (ContentPane) zurückgegeben auf dem mit add
                         //Elemente platziert werden können
c.setLayout(new BorderLayout(0, 0));      //Layout festlegen mit LayoutManager
JPanel head = new JPanel();               //neue Komponente erzeugen
c.add(head, BorderLayout.NORTH);          //hinzufügen der Komponente
```

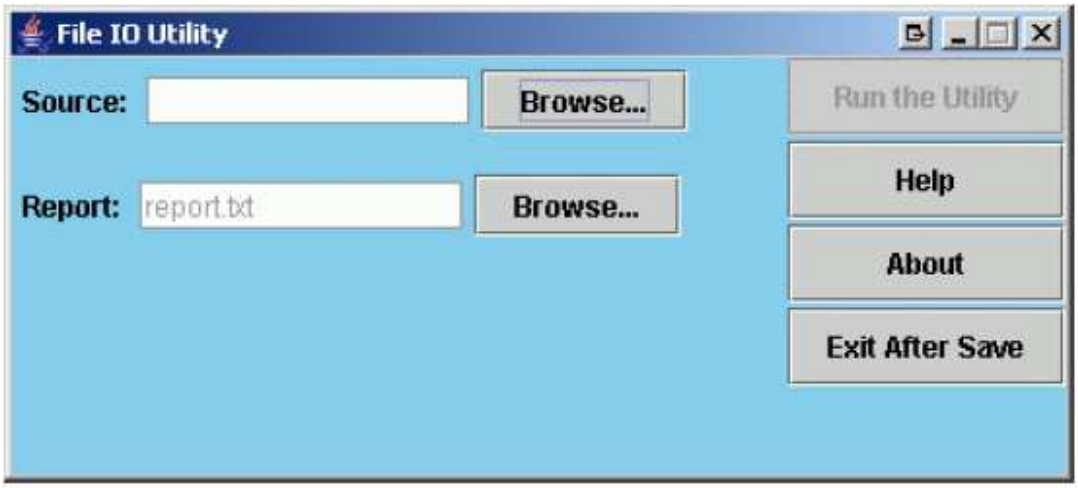
Frage (GUI): Wie könnte eine Hierarchie von Swing Containers und Components aussehen, mit dem folgendes User-Interface erzeugt wird (umgangssprachliche Beschreibung genügt, Skizze erwünscht):



zum Beispiel:



Frage (GUI): Wie könnte eine Hierarchie von Swing Containers und Components aussehen, mit dem folgendes User-Interface erzeugt wird (umgangssprachliche Beschreibung genügt, Skizze erwünscht):



ICH SPARS MIR – SIEHE OBEN….bzw Matzinger Folien zur GUI!!

Frage: Beschreiben Sie: Was ist ein Mouse Listener, was ein Mouse Motion Listener? Was ist ein Listener innerhalb der Programmiersprache Java (eine Variable, eine Klasse, ein Objekt, eine Methode, ein Interface, ...)?

MouseListener: stellt folgende Methoden zur Verfügung:

```
mousePressed();
mouseReleased();
mouseClicked();
mouseEntered();
mouseExited();
```

Reaktion auf die unterschiedlichen Events werden vom Empfänger implementiert

MouseMotionListener: Methoden: `mouseDragged(); mouseMoved();`
Methoden ebenfalls vom Empfänger implementiert

Listener: Interface, stellt Methoden zur Verfügung die von der jeweiligen Klasse implementiert werden

Kapitel VI: Design Patterns

Frage: Was sind “Design Patterns”? Wozu sind sie nützlich?

Denkmuster, Lösung für öfter auftretendes Problem

Definition(en):

- Ein Design Pattern beschreibt die **grundsätzliche Lösung eins wiederkehrenden Problems**. Diese Lösung ist ‘millionenfach’ wieder verwendbar ohne das sich zwei Lösungen gleichen.
- Ein Design Pattern identifiziert, erklärt und bewertet grundlegende **Entwurfsaspekte** in objekt-orientierten Systemen
- Ein Design Pattern ist ein Weg zur Verpackung und **Wiederverwendung von Wissen**

Frage: Was ist kennzeichnet ein “Singleton”? Wie kann man es implementieren?

Singleton ist eine Klasse, von der nur ein einziges Objekt erzeugt werden darf, stellt globale Zugriffsmöglichkeit auf dieses Objekt zur Verfügung und instanziert es beim ersten Zugriff automatisch.

Implementierung:

```
public class Singleton {

	private static Singleton instance = null;

	public static Singleton getInstance(){
		if (instance == null) {
			instance = new Singleton();
		}
		return instance;
	}

	private Singleton(){
	}
}
```

Kapitel VII: MVC

Frage: Erklären Sie kurz das MVC-Modell? Wozu dient es? Wo wird es eingesetzt?

Anstatt den Code in eine einzelne Klasse zu packen, werden beim MVC Konzept drei Bestandteile eines grafischen Elements unterschieden:

Model: enthält Daten des Dialogelements und speichert Zustand
View: grafische Darstellung
Controller: Verbindungsglied zwischen beiden. Empfängt Tastatur- u. Mausereignisse und stößt erforderlichen Maßnahmen zur Änderung von Model und View an.

eingesetzt bei: Swing-Dialogelementen

Frage: Welche Komponente des MVC sollte von der grafischen Umgebung, in der die Applikation implementiert wird, unabhängig sein? Warum?

Model: stellt Daten zur Verfügung, kann mehrere Views besitzen, die sich alle auf die im Model gespeicherten Daten beziehen. Ändern sich diese, werden alle Views aktualisiert

- der Controller ist oft sehr einfach strukturiert, dass er gleich mit in die View gepackt wird

Kapitel VIII: Formale Sprachen

Frage: Was bedeutet der Begriff “Syntax einer Sprache”?

– Beschreibung der korrekten Form einer Sprache

Frage (T-Diagramme):

Auf Ihrem Computer ist ein C-Compiler installiert. Aus dem Internet bekommen Sie einen in C geschriebenen Java-Compiler und eine in C geschriebene Java Virtual Machine. Aus einem Vorprojekt besitzen Sie einen in Java geschriebenen Compiler, der Programme der Programmiersprache Z in Java übersetzt. Zeichnen Sie in T-Diagrammen,
wie Sie ein Programm in der Programmiersprache Z auf Ihrem Computer zum Laufen bringen.

Frage: Eine Grammatik ist wie folgt definiert:

Terminalsymbole (Lexicals): ID, NUM, +, -, (,), ,
Nonterminale: expr, op, list
Regeln:

expr → ID (list)
expr → expr op expr
expr → (expr)
expr → ID
expr → NUM
op → +
op → -
list → expr
list → expr , list

Startsymbol: *expr*

Finden Sie Ableitungen für folgende Ausdrücke:

ID (ID + NUM)
NUM + ID - NUM
ID ((NUM – ID) + ID , ID , ID + NUM , ID (ID + NUM))

Sind folgende Ausdrücke ableitbar? Oder nicht?

ID ( )
ID (- NUM + ID)
ID (ID (ID (ID (NUM))))

Frage: Schreiben Sie obige Grammatik in Bakkus-Naur Normalform.

Frage: Gegeben ist folgende Grammatik in Bakkus-Naur Normalform

```
Statement:
      Ifstatement
      ID = Expression ;

Ifstatement:
      if ( Expression ) Statement [ else Statement ]

Expression:
ID { + ID }
```

Schreiben Sie ein nach dieser Definition korrektes Statement, in dem mindestens zweimal das Keyword if vorkommt.

Frage: Gegeben ist folgende reguläre Grammatik:

Terminalsymbole: a, b, c
Nonterminalsymbole: A, B, C
Regeln:

$A \rightarrow a\ B$
$B \rightarrow b\ C$
$B \rightarrow c\ A$
$A \rightarrow \varepsilon$

Startsymbol: A

Zeichnen Sie einen Automaten, der dieselbe Sprache (aus den Symbolen a,b,c erzeugt).
Schreiben Sie eine regular Expression, die dieselbe Sprache erzeugt.
Beschreiben Sie umgangssprachlich, wie diese Sprache aussieht, i.e. welche Ausdrücke in dieser Sprache sind.

Frage: Schreiben Sie einen Automaten (eine regular expression) für float-Konstanten in Java.

Frage: Gegeben sei folgender regulärer Ausdruck: d([a-c][0-2][a-c])x)*d

Schreiben Sie einen äquivalenten Automaten dazu. Geben Sie zwei Worte an, die vom Automaten akzeptiert werden (bzw. den regulären Ausdruck matchen) und zwei Worte, die nicht akzeptiert werden.

Frage: Betrachten Sie folgenden Java Code.

```
import java.util.regex.*;

Pattern p = Pattern.compile("(ab)*c");
Matcher m1 = p.matcher("ababc");
Matcher m2 = p.matcher("abcabc");
Matcher m3 = p.matcher("ababcdeab");
Matcher m4 = p.matcher("aababc");
boolean b1 = m1.matches();
boolean b2 = m2.lookingAt();
boolean b3 = m3.matches();
boolean b4 = m4.lookingAt();
```

Welchen Wert bekommen die Variable b1 bis b4?

Kapitel IX: Networks, Sockets, ...

Frage: Beschreiben Sie kurz die Kommunikation mit TCP und UDP? Was ist der Unterschied?

!!!!!Rechnernetze1!!!!!!

zur Erinnerung
TCP, *verbindungsorientiert*, stellt virtuellen Kanal zwischen 2 Rechnern her, auf Zuverlässigkeit ausgelegt
UDP, *verbindungslos*

Frage: Welche der Befehle TCP write, read, UDP write, UDP read können den ausführenden Thread blockieren, wenn der Partner am Ende der Leitung nicht richtig reagiert.

TCP write bzw read: TCP ist verbindungsorientiert, wartet daher bei write/read auf Reaktion des Empfängers, wenn keine Reaktion → blockiert

Frage: Beschreiben Sie den Aufbau einer TCP-Verbindung auf der Client und auf der Serverseite (umgangssprachlich oder in Struktogrammen. Genauer Code ist nicht erforderlich).

ganz kurz

Server listen()
Client connect()
Server accept()
Client request()
Server reagiert()
disconnect()

Frage: In welchem Layer des OSI-Modells befinden Sie sich, wenn Sie mittels Java Sockets kommunizieren? Nennen Sie mindestens zwei etablierte Protokolle dieses Layers.

im 4. Layer
TCP, IP, UDP

Frage: Gegeben ist folgendes Codestück. Wo könnte es blockieren? Was muss die Gegenseite tun, damit eine Kommunikation stattfindet.

```
try {
ServerSocket serv = new ServerSocket(1099);
Socket s = serv.accept();          //könnte blockieren wenn Client
                                   //nicht verbindet, solange wartet Server
InputStream in = s.getInputStream();
int len;
byte[] b = new byte[256];
in.read(b);
// XXX
} catch (Exception e){}
```

Frage: Welche Close-Statements fehlen an der Stelle XXX? Umgangssprachliche Beschreibung genügt.

in.close();
serv.close();

Frage: Wieviele Threads müssen auf jeweils einer Seite der TCP-Verbindung laufen, damit man über diese asynchron kommunizieren kann (als damit der Datenfluss in die eine Richtung unabhängig ist vom Datenfluss in die andere Richtung)?

1. einer für Server.listen()
2. und einer für sendenden Client

Frage: Eine Implementierung einer TCP-Verbindung verwendet in allen denkbaren Situationen folgende Disconnect-Funktion:

Variante 1:

```
public void disconnect() {
        try { instream.close(); } catch (Exception ex) {};
        try { outstream.close(); } catch (Exception ex) {};
        try { sock.close(); } catch (Exception ex) {};
        try { servsock.close(); } catch (Exception ex) {};
}
```

Variante 2:

```
public void disconnect() {
        try { instream.close();
                outstream.close();
                sock.close();
                servsock.close(); } catch (Exception ex) {};
}
```

Welches ist die bessere Implementierung? Warum? Was kann bei der schlechteren Implementierung passieren?

besser: Variante 1, jede Anweisung steht in einem try-Block, sollte eine Exception ausgelöst werden, werden die anderen Anweisungen weiter ausgeführt (im Gegensatz zu Variante 2)

Frage: Beschreiben Sie kurz das Wesen von Multicast-Verbindungen? Nennen Sie ein Beispiel für eine Applikation wo diese sinnvoll eingesetzt werden könnten.

Prinzip: einer sendet zu mehreren
z.B.: Audio/Video-Streaming