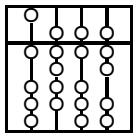
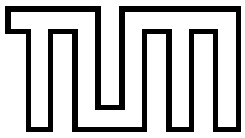
Name: | Vorname: | Matr.–Nr.:



TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



Lehrstuhl für Sprachen und Beschreibungsstrukturen WS 2006/2007
Praktikum Grundlagen der Programmierung 14. Februar 2007
Prof. Dr. Helmut Seidl

Klausur zu Einführung in die Informatik I

Hinweis: In dieser Klausur können Sie insgesamt 70 Punkte erreichen. Zum Bestehen der gesamten Prüfung (Zwischen- und Endklausur) benötigen Sie mindestens 40 von insgesamt 100 Punkten.

Aufgabe 1 Applets (10 Punkte)

Programmieren Sie ein Applet, dessen Hintergrundfarbe am Anfang blau ist. Bei einem Mausklick auf das Applet soll seine Hintergrundfarbe von blau nach grün bzw. von grün nach blau wechseln.

Hinweis: Zum Setzen der Hintergrundfarbe stellt die Klasse `Applet` die Methode

```
void setBackground(Color c)
```

bereit. Um auf ein Maus-Event zu reagieren müssen Sie einen geeigneten `MouseListener` implementieren. Dazu können Sie folgende Klasse `MouseAdapter` erweitern:

```
public class MouseAdapter implements MouseListener {
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
}
```

Aufgabe 2 IO und Exceptions **(3 + 7 = 10 Punkte)**

Ziel dieser Aufgabe ist es eine statische Methode

```
int berechneWert(String datei)
```

zu schreiben, die den Wert eines in einer Datei gespeicherten arithmetischen Ausdrucks berechnet. Dabei beschränken wir uns auf arithmetische Ausdrücke in denen lediglich **natürliche Zahlen** und **Addition** auftreten. Für den im nachfolgenden Beispiel angegebenen Ausdruck soll ein Aufruf der geforderten Methode 42 zurück liefern.

```
21 + 2 + 19
```

Dabei ist folgendes zu beachten:

- Ist das Format der Datei falsch, so soll bei einem Aufruf der Methode `berechneWert()` eine `FalschesFormatException` geworfen werden, die die Information enthält, beim Lesen des **wievielten Wortes** ein Fehler aufgetreten ist.
- Existiert keine Datei namens `datei`, so soll eine `DateiNichtVorhandenException` geworfen werden.

Zur Lösung dieser Aufgabe können Sie die nachfolgend beschriebene Klasse `DateiLeser` verwenden, mit der eine Datei Wort für Wort gelesen werden kann. Die Klasse `DateiLeser` ist gegeben und muss **nicht** von Ihnen implementiert werden.

- Der Konstruktor `DateiLeser(String datei)` erzeugt ein `DateiLeser`-Objekt für die Datei `datei`. Existiert eine solche Datei nicht, so wird eine `DateiNichtVorhandenException` geworfen.
- Um Wort für Wort lesen zu können, wird die Methode `String leseNaechstesWort()` zu Verfügung gestellt. Diese liefert also beim *i*-ten Aufruf das *i*-te Wort als String zurück. Existiert kein *i*-tes Wort, so wird `null` zurückgeliefert.

a) Definieren Sie die oben beschriebene `FalschesFormatException`.

b) Definieren Sie die oben beschriebene statische Methode `berechneWert()`.

Hinweis: Die statische Methode `int parseInt(String s)` der Klasse `Integer` liefert den `int`-Wert der als String repräsentierten Zahl `s` zurück. Falls der übergebene String `s` keinen `int`-Wert repräsentiert wird eine `NumberFormatException` geworfen.

Aufgabe 3 Tabellenkalkulation **(9 + 9 = 18 Punkte)**

In dieser Aufgabe sollen Sie ein vereinfachtes Modell einer Tabellenkalkulation in Java entwerfen. Grundelement der Tabellenkalkulation ist eine Tabelle, die aus Zellen besteht — organisiert in 30 Zeilen und 30 Spalten. In jeder Zelle kann ein Ausdruck stehen. Ein Ausdruck kann dabei folgendes sein:

- eine `int` -Konstante (Klasse `Const`), z.B. 5;
- Die Summe zweier Ausdrücke (Klasse `Add`);
- das additive Inverse eines Teilausdrucks (Klasse `UnMinus`) oder
- ein Verweis auf eine andere Zelle (Klasse `Ref`) der Tabelle.

a) Implementieren Sie die geschilderte Klassenhierarchie zur Repräsentation von Ausdrücken. Verwenden Sie eine gemeinsame Basisklasse `Expr`, die eine abstrakte Methode

```
public abstract int eval()
```

zur Auswertung des Ausdrucks deklariert.

Gehen Sie zur Implementation von `Ref` davon aus, dass die Klasse `Tabelle` die in b) beschriebene Methode `int evalZelle(int i, int j)` zu Verfügung stellt.

b) Implementieren Sie die Klasse `Tabelle`. Auf den Inhalt der Tabelle darf nur über die folgenden Methoden zugegriffen werden:

- `int evalZelle(int i, int j)`
  Der Aufruf `evalZelle(i,j)` gibt den Wert des in Zelle (i, j) gespeicherten Ausdrucks zurück. Um doppelte Berechnungen zu vermeiden, soll ein bereits berechnetes Ergebnis für jede Zelle gespeichert werden. Nachfolgende Aufrufe von `evalZelle()` liefern das gespeicherte Ergebnis zurück.
- `void setExpression(int i, int j, Expr e)`
  Der Aufruf `setExpression(i,j,e)` speichert den Ausdruck `e` in Zelle (i, j) ab. Damit werden alle gespeicherten Ergebnisse potentiell ungültig und müssen daher zurückgesetzt werden.

Aufgabe 4 Vererbung **(3 + 5 + 6 = 14 Punkte)**

Wichtig: Lösen Sie diese Aufgabe direkt auf **diesem Blatt**!

Gegeben seien die im **Anhang** angegebenen Interface- und Klassen-Definitionen.

a) Stellen Sie die Klassen-Hierarchie mithilfe eines UML-Klassen-Diagramms dar.

b) Welche der folgenden Zuweisungen sind erlaubt und welche nicht?

	Erlaubt!	**Nicht erlaubt!**
(i) `B x = new C();`	☐	☐
(ii) `I x = new B();`	☐	☐
(iii) `J x = new A();`	☐	☐
(iv) `D<B> x = new D<A>();`	☐	☐
(v) `D<A> x = new D<B>();`	☐	☐
(vi) `D<A> x = new D<A>();x.t = new B();`	☐	☐
(vii) `J x = new C();`	☐	☐
(viii) `A x = (new D<A>()).d2(new B());`	☐	☐
(ix) `B b1 = new B(); B b2 = b1.copy();`	☐	☐
(x) `J x = new B(); (new D<A>()).d1(x);`	☐	☐

c) Welche Ausgaben produzieren die folgenden Anweisungen?

Hinweis: In manchen Fällen kommt es zu einer Exception.

(i) `I x = new B(); x.i();`

.....

.....

.....

.....

(ii) `I x = new C(); x.i();`

.....

.....

.....

.....

(iii) `B x = (B) new A(); x.i();`

.....

.....

.....

.....

(iv) `J x = new B(); B y = (B) x; y.i();`

.....

.....

.....

.....

(v) `D<A> x = new D<A>(new C()); x.d2(x.t);`

.....

.....

.....

.....

(vi) `D<B> x = new D<B>(); x.d1(new A());`

.....

.....

.....

.....

Aufgabe 5 Thread-Synchronisation **(18 Punkte)**

In dieser Aufgabe sollen 3 Threads miteinander kommunizieren. Weiterhin soll es eine globale `int`-Variable `zahl` geben, auf die alle Threads zugreifen sollen.

- Thread 1 soll wiederkehrend eine Sekunde schlafen und dann die Variable `zahl` um eins erhöhen.
- Thread 2 und 3 sollen immer dann den **aktuellen** Wert der Variablen `zahl` ausgeben, wenn dieser durch 7 teilbar ist. Sorgen Sie dafür, dass ein Wert der Variablen `zahl` von jedem Thread höchstens einmal ausgegeben wird. Thread 2 und 3 sollen über jede Änderung des Wertes der Variablen `zahl` benachrichtigt werden und dann die Chance erhalten diesen auszugeben.

Implementieren Sie geeignete Klassen für die Threads sowie eine `main`-Methode, die die oben geschilderte Situation initialisiert.

Anhang zu Aufgabe 4

Gegeben seien folgende Interface- und Klassen-Definitionen:

```
interface I {
    void i();
}
interface J {
    void j();
}
class A implements I {
    public void i() { System.out.println("i() in A"); }
    public A copy() { return this; }
}
class B extends A implements J {
    public void j() { System.out.println("j() in B"); }
}
class C extends A implements J {
    public void i() {
        super.i();
        System.out.println("i() in C");
    }
    public void j() { System.out.println("j() in C"); }
}
class D<T extends I> extends C{
    public T t;
    public D(){ }
    public D(T t){
        System.out.println("initialisiere t");
        this.t = t;
    }
    void d1(I i){
        i.i();
        System.out.println("d1(I) in D");
        }
    void d1(A a){
        a.i();
        System.out.println("d1(A) in D");
    }
    void d1(B b){
        b.i();
        System.out.println("d1(B) in D");
    }
    T d2(T t){
        d1(t);
        return t;
    }
}
```