Name: | Vorname: | Matr.–Nr.:

Technische Universität München
Fakultät für Informatik
Prof. Dr. Seidl

WS 2004/2005
12. Februar 2005

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 Fehlerbehandlung mit Exceptions (2 + 5 + 8 = 15 Punkte)

Betrachten Sie folgendes Programmstück:

```
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
String userinput;
int a, b;

System.out.println("1. Zahl eingeben: ");
userinput = in.readLine();
a = Integer.parseInt(userinput);

System.out.println("2. Zahl eingeben: ");
userinput = in.readLine();
b = Integer.parseInt(userinput);

int result = a / b;
System.out.println("Das Ergebnis ist " + result);
```

a) Was berechnet dieses Programmstück?

b) Welche Ausnahmen/Exceptions können hier auftreten? Wo können diese auftreten?

c) Erweitern Sie das Programmstück um eine sinnvolle Fehlerbehandlung.

Aufgabe 2 Vererbung **(5 + 6 + 6 = 17 Punkte)**

Gegeben seien folgende Java-Klassen:

```
class A {
    public void f() { System.out.println("f in A"); }
    public void g() { System.out.println("g in A"); f(); }
}
class B extends A {
    public void f() { System.out.println("f in B"); }
    public void h() { System.out.println("h in B"); f(); g(); }
}
class C extends A {
    public void h() { System.out.println("h in C"); f(); g(); }
}
interface I {
    public void m();
}
abstract class D {
    public abstract void n(boolean b);
    public void o() { System.out.println("o in D"); n(false); }
}
class E extends D {
    public void n(boolean b) {
        System.out.println("n in E");
        if(b) o();
    }
    public void m() { System.out.println("m in E"); }
}
class F extends E implements I {
    public void n(boolean b) {
        System.out.println("n in F");
    }
}
```

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

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

(i) `A a = new A();`
(ii) `B b = new B(); A t = b;`
(iii) `B b = new A();`
(iv) `D d = new D();`
(v) `D d = new F();`
(vi) `F f = new F(); I i = f;`

c) Welche Ausgaben produzieren die folgenden Anweisungen?

(i) `A a = new A(); a.g();`
(ii) `B b = new B(); b.h();`
(iii) `C c = new C(); c.g();`
(iv) `B b = new B(); A a = b; a.g();`
(v) `D d = new E(); d.n(true);`
(vi) `F f = new F(); f.o();`

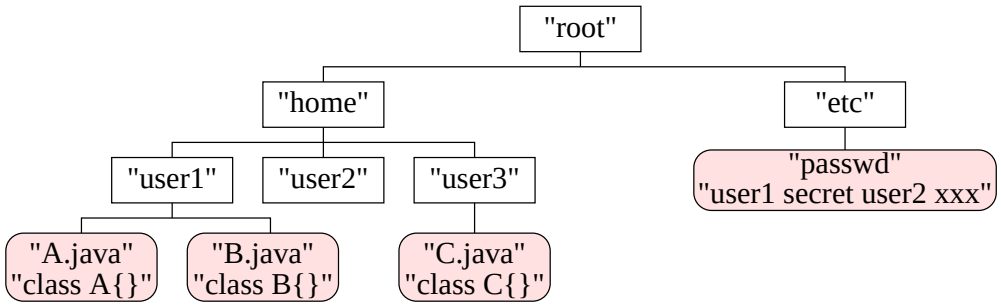
Aufgabe 3 Datenstrukturen **(4 + 8 + 8 = 20 Punkte)**

Entwickeln Sie ein System von Klassen, mit denen sich Dateien in Verzeichnissen organisieren lassen. Eine Datei besteht aus ihrem Namen vom Typ `String` und ihrem Inhalt, der der Einfachheit halber auch vom Typ `String` ist. Ein Verzeichnis besteht aus seinem Namen vom Typ `String` und aus einer (möglicherweise leeren) Liste von Unterverzeichnissen und Dateien. Zur Implementierung können Sie die Klasse `List<E>` benutzen, die wie folgt definiert ist:

```
public class List<E>{
    public E info;
    public List<E> next;

    public List(E e){
        info = e;
        next = null;
    }
    public List(E e, List<E> n){
        info = e;
        next = n;
    }
    public void insert(E e){
        next = new List<E>(info,next);
        info = e;
    }
}
```

Ein Beispiel-Filesystem ist unten abgebildet. Verzeichnisse sind durch Rechtecke und Dateien durch abgerundete graue Rechtecke dargestellt:



a) Beschreiben Sie Ihre Datenstruktur mit einem geeigneten UML-Klassendiagramm. Führen Sie dazu Klassen `File` und `Directory` ein. Methoden sind nicht verlangt. Alternativ, können Sie auch den Java-Code Ihrer Klassen direkt angeben.

b) Implementieren Sie eine Methode `public void makeFile(String name,String inhalt)` der Klasse `Directory`, die eine Datei mit Inhalt im aktuellen Verzeichnis anlegt. Das aktuelle Verzeichnis ist das `Directory`-Objekt, dessen Methode aufgerufen wurde. Wenn ein Verzeichnis oder eine Datei mit dem selben Namen im aktuellen Verzeichnis existiert, soll nichts gemacht werden.

c) Implementieren Sie eine Methode `public Directory changeDir(List<String> path)` der Klasse `Directory`, die als Argument einen Pfad `path` (als Liste von `String`s) erhält. Wenn `path` zu einem Unterverzeichnis *d* aus dem aktuellen Verzeichnis führt, soll *d* zurückgeliefert werden (z.B. führt `new List<String>("home",new List<String>("user1"))` im "root"-Verzeichnis zum "user1"-Verzeichnis). Sonst soll `null` zurückgeliefert werden.

Aufgabe 4 Threads **(7 + 7 + 4 = 18 Punkte)**

Implementieren Sie eine einfache Ampel, die die Abläufe an einem Fußgängerüberweg regelt. Die Ampel soll dabei den Fußgängern bzw. Autos eine gewisse Zeit die Berechtigung geben, die Straße zu überqueren, bzw. den Fußgängerüberweg zu passieren.

a) Implementieren Sie zunächst einen Thread `Ampel`. Dieser habe folgende Eigenschaften:

   - Die Ampel habe nur zwei Zustände: Entweder können die Fußgänger gehen oder die Autos passieren.
   - Im Betrieb wechselt die Ampel permanent zwischen diesen Zuständen, wobei sie in jedem Zustand eine Minute bleibt.

b) Geben Sie die Implementierung eines Threads `Fussgaenger` an. Der Fußgänger sei durch einen Namen und die Ampel, an der er steht, gekennzeichnet. Er soll solange warten, bis die Ampel ihm das überqueren erlaubt.

c) Schreiben Sie eine `main`-Methode, die eine Ampel und zwei Fußgänger startet.