Programmierung

UNIVERSITÄT KOBLENZ · LANDAU

Institut für Informatik

Prof. Dr. Ralf Lämmel
Dr. Manfred Jackel

SS 2009

Nachklausur

Modul „Programmierung“

13.11.2009

Name																									
Vorname																									
E-Mail-Adresse																			@uni-koblenz.de						
Matrikelnummer																									

Studiengang:
☐ Computervisualistik (Diplom)
☐ Informatik (Diplom)
☐ Computervisualistik (BSc)
☐ Informatik (BSc)
☐ B Ed _____
☐ Informationsmanagement (BSc)

☐ Diese Prüfungsleistung ist mein 2. Versuch

☐ Diese Prüfungsleistung ist mein 3. Versuch

Jede Aufgabe bringt 12 Punkte, außer Aufgabe 6 (Mutiple Choice), die 16 Punkte wert ist.

	1	2	3	4	5	6	7	GESAMT
Punkte								

Auswertung:

Aufgabe 1 "Parsing" - Grundlegende Konzepte

Die folgende EBNF-Grammatik definiert einfache Ausrücke mit Multiplikation und Addition mit der "üblichen" Priorität (Multiplikation vor Addition).
Erweitern Sie die Grammatik um einen Operator ↑, der das Potenzieren einführt (3↑2 ergibt 9) und geben Sie zusätzliche Regeln an, die dem Potenzieren eine höhere Priorität geben als der Multiplikation. Verwenden Sie das zusätzliche Nonterminal `powExpr`.

```
expr       : multExpr ( ('+' | '-') multExpr)*;
multExpr   : atom ('*' atom)*;
atom       : INT | ID | '(' expr ')';
```

```
expr       : multExpr ( ('+' | '-') multExpr)*;
multExpr   : powExpr ('*' powExpr)*;

powExpr    : atom ('↑' atom)*
atom       : INT | ID | '(' expr ')';
```

Aufgabe 2 "Combinator Libraries" (Grundlegende Konzepte)

Aus der abstrakten Klasse

```
public abstract class Acceptor {
     public abstract boolean accept(Input i);
}
```

seien zwei Acceptorklassen

```
public class IntAcceptor extends Acceptor

public class CommaAcceptor extends Acceptor
```

abgeleitet, deren accept-Methode Integerliterale resp. das Komma-Zeichen „,“ erkennen.

Schreiben Sie Code für ein Parser-Objekt `integerList`, das eine Komma-separierte Liste von Integerzahlen parset (z.B. „1,2,3,4“ ohne die Anführungszeichen). Eine Liste enthalte immer mindestens eine Integer. Verwenden Sie Parserkombinatoren wie Seqience, Star, Optional etc.

```
     // Code here

Acceptor integerList=

                new Sequence( new IntAcceptor(),
                                new Star(new Sequence(new
CommaAcceptor(),new IntAcceptor())));
```

```
}
```

Aufgabe 3 "XML Processing DOM" (Codeverständnis)

Gegeben Sei die XML-Datenstruktur

```
<Buch>
     <Autor> Donald E. Knuth</Autor>
     <Titel>The Art of Computer Programming</Titel>
</Buch>
```

Erzeugen Sie mit Hilfe des DOM-API DOM-Datenobjekt buch, das diese Datenstruktur darstellt. Die genauen Bezeichnungen der Typen und Methoden der DOM API werden nicht verlangt. Zur Erfüllung der Aufgabe reicht es, wenn Sie denn ungefähren Ablauf der Konstruktion des DOM-Baumes durch eine Code-Sequenz beschreiben können.

```
Element buch = doc.createElement("Buch");

Element autor = doc.createelement("Autor");
autor.setNodeValue = "Donald E. Knuth";
Element titel = doc.createElement("Titel");
titel.setNodeValue("The Art of Computer Programming";
buch.appendChild(autor);
buch.appendChild(titel);
```

Aufgabe 4 "Generics" (Grundlegende Konzepte)

Gegeben sei der einstellige generische Funktor

```
public interface UnaryFunction <X,Y> {
       Y apply(X x);
      }
```

und die nachstehende generische Baumklasse MyTree<X > mit Knoteninformation info vom Typ X und linkem und rechtem Unterbaum l,r. Die Methode tmap soll den Baum erzeugen, der entsteht, wenn man den Funktor auf alle Knoten des Baumes anwendet.

Programmieren Sie tmap aus!

```
public class MyTree<X> {
      X info;
      MyTree<X> l, r;

      public MyTree(MyTree<X> l, X info, MyTree<X> r) {
            this.l = l;
            this.info = info;
            this.r = r;
      }

      <Y> MyTree<Y> tmap(UnaryFunction<X, Y> f) {

            // Your code goes here



            return new MyTree<Y>(l.tmap(f), f.apply(info), r.tmap(f));









      }

}
```

Aufgabe 5 "Aspect Oriented Programming" (Grundlegende Konzepte)

Gegeben sei folgendes Programmfragment:

```
if (y>x) thencase(x,y); else elsecase(y,x);
```

Geben Sie einen Aspekt an, der zählt, wie oft der then-Teil und wie oft der else-Teil durchlaufen wird. Die Variablen x und y sind vom Typ int und thencase und elsecase seien Methodenaufrufe.

```
public aspect Count {

	static int countThen = 0, countElse = 0;

	before(): call(void thencase(int,int)) {
		countThen++;
	}

	before(): call(void elsecase(int,int)) {
		countElse++;
	}









}
```

Aufgabe 6 Threads (Grundlegende Konzepte)

Gegeben seien 2 Felder a und b und 2 Threads, die jeweils exklusiven Zugriff auf beide Felder brauchen um die Methoden doit1(a,b) resp. doit2(a,b) dann auszuführen. Um den exklusiven Zugriff auf a resp. b zu gewährleisten, benutzen wir die Semaphore asema und bsema.

Welche Vorkehrungen (Aufruf von Semaphoremethoden) muss jeder Thread tätigen, damit garantiert kein Deadlock entsteht und somit jeder Thread die Chance hat, doit1 resp, doit2 aufzurufen

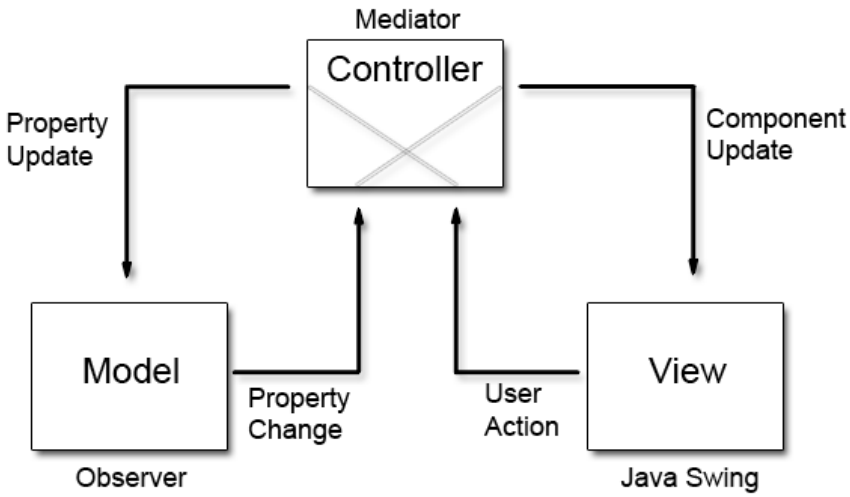
```
int[] a,b;
Semaphore asema= new Semaphore(1); //allowing 1 exclusive access
Semaphore asemb= new Semaphore(1);
```

// Thread 1
//get a,b exclusively for this thread
 	
// gleiche Reihenfolge der
 	
asema.aquire();
bsema.aquire();
 	
doit1(a,b)
// cleanup code here
 	
asema.release();
bsema.release();
	// Thread 2
	//get a,b exclusively for this thread
	
	aquire-Aufrufe auf beiden Seiten
	
	asema.aquire(); //
	bsema.aquire();
	
	doit2(a,b)
	// cleanup code here
	
	asema.release();
	bsema.release();

Aufgabe 7 “Model View Controller” (Grundlegende Konzepte)

Durch folgende 3 Anweisungen wird je ein Model-, View- und Controller-Objekt erzeugt (Apple Cocoa).

```
MyController controller = new MyController();
MyModel myModel = new MyModel();
MyView myView = new MyView(controller);
```



Skizzieren Sie durch mindestens 2 Anweisungen, welche Objekte sich bei einem anderen Objekt registrieren müssen, damit

- Property-Änderungen des Modells zu den Views gelangen und
- GUI-Aktionen in der View mithilfe des Controllers Property-Änderungen des Modells übersetzt werden.

Alternativ können Sie die notwendige Vorgehensweise auch verbal beschreiben.

```
controller.addView(myView); //Der View muss ich beim Controller
registrieren

controller.addModel(myModel); // Der Kontroller erhält eine Referenz
auf das Modell
```