Programmierparadigmen

Wintersemester 2012/13
Torsten Görg, Sandro Degiorgi

5. Übung
Einsendung bis spätestens 07.01.2013, 13:00 Uhr
Besprechungen am 09./10./11. Januar 2013

Aufgabe 5.1 ..... 14 Punkte

In der Vorlesung wurde der Unterschied zwischen Referenz- und Wertesemantik vorgestellt. Untersuchen Sie das folgende Java-Programm. Zeichnen Sie analog zu den Beispielen im Skript auf Seite 5-30 für die mit `A`, `B`, `C`, `D` und `E` markierten Programmstellen jeweils eine Skizze in Box-Pointer-Notation, die alle Objekte mit ihren Werten und Referenzen auf andere Objekte zeigt, wie sie bestehen, wenn das Programm bis zu dem angegeben Punkt ausgeführt ist. Enthält ein Objekt mehrere Komponenten, so ist das zugehörige Rechteck mit waagerechten Strichen in Felder zu unterteilen und in jedes Feld der Wert einer Komponente einzutragen.

```
01  class Node {
02     private int value;
03     private Node next;
04
05     public Node( int value )   { this.value = value;
06          if( value != 0 ) this.next = new Node( value / 2 ); else this.next = this; }
07     public int getValue()   { return value; }
08     public void setValue( int value )   { this.value = value; }
09     public Node getNext()   { return next; }
10     public void setNext( Node next )   { this.next = next; }
11  }
12
13  class Aufgabe_5_1 {
14     public static void main( String args[] ) {
15        Node var1;
16        Node var2 = new Node( 5 );
17        // A
18        int var3 = 23;
19        var1 = var2;
20        var2.getNext().setValue( var3 );
21        var1.setValue( 7 );
22        // B
23        var2 = new Node( var3 - 27 );
24        var2.getNext().getNext().setNext( var2.getNext() );
25        var2.getNext().getNext().getNext().setValue( var1.getValue() );
26        // C
27        int var4 = var2.getNext().getValue();
28        var2 = var2.getNext().getNext();
29        var3 = 9;
30        var1.getNext().getNext().setNext( new Node( 3 ) );
31        // D
32        var2.getNext().setNext( var1.getNext() );
33        var1.getNext().getNext().setNext( var2 );
34        // E
35     }
36  }
```

Aufgabe 5.2 ..... 23 Punkte

Untersuchen Sie den folgenden Programmcode mit Java-Klassen für Tiere:

```
01  class Tier
02  {
03     private int anzahlBeine;
04     public Tier( int anzahlBeine ) { this.anzahlBeine = anzahlBeine; }
05     public Tier() {}
06  }
07
08  class Vogel extends Tier
09  {
10     public Vogel() {}
11     static protected int anzahlEier = 0;
12     public boolean flugfaehig = true;
13     public Vogel( boolean flugfaehig ) { super( 2 ); this.flugfaehig = flugfaehig; }
14     public void legeEier() {}
15     static public int eieranzahl() { return anzahlEier; }
16  }
17
18  class Amsel extends Vogel
19  {
20     public Amsel() {}
21     public void legeEier() { anzahlEier += 3; /* weiterer, amselspezifischer Code */ }
22  }
23
24  class Pinguin extends Vogel
25  {
26     public Pinguin() { super( false ); }
27     public void legeEier() { anzahlEier += 7; }
28  }
29
30  class Katze extends Tier
31  {
32     public Katze() { super( 4 ); }
33  }
34
35  class Dinosaurier extends Tier
36  {
37     public boolean flugfaehig = false;
38     public Dinosaurier( int anzahlBeine, boolean flugfaehig )
39        { super( anzahlBeine ); this.flugfaehig = flugfaehig; }
40     public void legeEier() { /* Code zum Legen von Eiern */ }
41  }
```

1. Identifizieren Sie im obigen Programmcode unter Angabe der zugehörigen Zeilennummern alle Klassendefinitionen, Konstruktoren, Instanzattribute (fields), Klassenattribute (`static` fields), Instanzmethoden (methods), Klassenmethoden (`static` methods) und expliziten Aufrufe eines Basisklassenkonstruktors.
2. Welche Klassenhierarchie bilden die Klassen in diesem Programm? Geben Sie dazu alle Vererbungsbeziehungen zwischen den Klassen an.
3. Welchen Fehler gibt es in der Implementierung der Klassenhierarchie? Wie und wo hätte man darüber hinaus die Klassenbibliothek eleganter entwerfen können?
4. Nehmen Sie nun an, dass Zeile 01 modifiziert würde zu

   ```
   class Tier extends Vogel
   ```

   Warum ist es vernüftig, dass dies im Kontext des restlichen Programms nicht erlaubt ist?

5. Erweitern Sie das vorliegende Programm um eine Klasse für Säugetiere und betten diese neue Klasse geeignet in die Klassenhierachie ein.

6. Betrachten Sie nun die folgende Nutzung der Klassenbiliothek:

```
01   static void erstelleZoo()
02   {
03      Vogel tier1 = new Pinguin();
04      Vogel tier2 = new Tier( 5 );
05      Tier tier3 = new Amsel();
06      Tier tier4 = new Tier( 8 );
07      Dinosaurier tier4 = new Dinosaurier( 2 );
08      Dinosaurier tier5 = new Dinosaurier( 4, false );
09      tier1.legeEier();
10      tier1.flugfaehig = true;
11      tier4.anzahlBeine = 4;
12      tier3.legeEier();
13      int eier1 = tier5.eieranzahl();
14      int eier2 = Vogel.eieranzahl();
15      tier5.legeEier();
16   }
```

   Welche der Anweisungen in den Zeilen 03 bis 15 sind zulässig und welche nicht? Begründen Sie bei den nicht zulässigen Anweisungen, warum diese jeweils inkorrekt sind.

7. Ändern Sie die vorliegenden Klassenhierarchie für Tiere so ab, dass die Methode `legeEier` mit der Anweisung `tier.legeEier();` für jedes `Tier`-Objekt eines beliebigen Tieres, dass Eier legen kann, aufrufbar ist, wobei die Variable `tier` immer mit demselben Typ statisch typisiert ist.

Aufgabe 5.3 ..... 13 Punkte

In praktisch allen objekt-orientierten Sprachen gibt es Klassendefinitionen, Konstruktoren, Instanzattribute (fields), Klassenattribute (`static` fields), Instanzmethoden (methods) und Klassenmethoden (`static` methods).

1. Informieren Sie sich unter den angegebenen Quellen, wie die Programmiersprachen Ada, C++ und Java diese Einheiten umsetzen.
   - Ada http://en.wikibooks.org/wiki/Ada_Programming/Object_Orientation
   - C++ http://www.cplusplus.com/doc/tutorial/classes/
   - Java http://docs.oracle.com/javase/specs/jls/se7/html/index.html, Kap. 8

2. Geben Sie je Sprache eine Klassendefinition mit Konstruktor, Instanzattributen, Klassenattributen, Instanzmethoden und Klassenmethoden an und kennzeichnen Sie diese jeweils.

3. Bezüglich der Konzepte im vorigen Aufgabenteil: Welche Gemeinsamkeiten und Unterschiede zwischen den Sprachen, bzw. welche Besonderheiten der einzelnen Programmiersprachen, sehen Sie?

4. In der Vorlesung wurden die Konzepte Klassenhierarchie und Vererbung, Erweiterbarkeit von Unterklassen, Polymorphie und die Möglichkeit dynamischer Methodenbindung in der OOP vorgestellt. Geben Sie zu den Sprachen Ada, C++ und Java jeweils konkret an, wie diese Konzepte umgesetzt werden.

Aufgabe 5.4 ..... 17 Punkte

Auf der Webseite der Veranstaltung finden Sie einen Python Skript mit dem Namen `geo.py`. Details zu Objektorientierung in Python finden Sie unter http://www.tutorialspoint.com/python/python_classes_objects.htm, speziell zu dem hier genutzten decorator `@staticmethod` unter http://docs.python.org/2/library/functions.html#staticmethod. Bitte nehmen Sie zur Kenntnis, dass diese Aufgabe **nichts** mit Duck-Typing zu tun hat.

1. Implementieren Sie ein möglichst äquivalentes Programm in Java.
2. Welche der von Ihnen nun implementierten Objekte sind aktive Objekte bzw. passive Objekte? Erklären Sie kurz, was der Unterschied zwischen den Objektarten ist. Geben Sie ggf. je ein Ausführungsmodell an, welches die jeweilige Art von Objekten nutzt.
3. Wandeln Sie das Programm nun in eine Ada Version mit `tagged-Types` um. Bitte nutzen Sie bei dieser Implementierung die Präfixform zum Aufruf der Methoden der jeweiligen Objekte. Details zur Objektorientierung in Ada finden Sie z.B. unter http://en.wikibooks.org/wiki/Ada_Programming/Object_Orientation.
4. Ändern Sie nun das Programm so ab, dass die Aufrufe nun in Parameterform durchgeführt werden können. Geben Sie auch die jeweiligen Aufrufe explizit an.

Aufgabe 5.5 ..... 13 Punkte

1. Geben Sie zu jeder der folgenden Aussagen eine Funktions-Signatur in C++ an, wobei die Funktionssignatur jeweils einen Parameter hat und dieser `x` heißt. Zu Risiken und Nebenwirkungen von C++ lesen Sie bitte die Packungsbeilage (http://www.possibility.com/Cpp/const.html) oder fragen Sie Ihren Kommilitonen oder Tutor.

   Die Methode akzeptiere ...

   a) einen formalen Parameter vom Typ `int`.
   b) einen konstanten formalen Parameter vom Typ `int`.
   c) einen änderbaren Zeiger auf einen änderbares Objekt vom Typ `int`.
   d) einen konstanten Zeiger auf ein änderbares Objekt vom Typ `int`.
   e) einen änderbaren Zeiger auf einen konstantes Objekt vom Typ `int`.
   f) einen konstanten Zeiger auf ein konstantes Objekt vom Typ `int`.

2. Welche konstanten bzw. variablen Objekte und konstanten bzw. variablen Views sind bei den Funktionen im ersten Aufgabenteil zu identifizieren?

   - ... a read-only handle ... (konstanter View)
   - ... a changeable variable ... (variables Objekt)
   - ... a changeable handle ... (variabler View)
   - ... a read-only variable ... (konstantes Objekt)

3. Geben Sie an, welche der folgenden Programmzeilen in den Rümpfen welcher der sechs Funktionen korrekt wären.

   a) `x  = 0;`
   b) `*x = 0;`
   c) `int *y = new int; x = y;`
   d) `int *y = new int; x = &y;`

   Geben Sie den Effekt jeder der von Ihnen als valide eingestuften Programmzeile an.

4. Geben Sie zu den Funktions-Definitionen äquivalente Prozedurdefinitionen in Ada an. Um die C++ Semantik exakt abbilden zu können, werden Sie Hilfskonstrukte nutzen müssen.