Lösungen: Achtung: Der Kurstext wird laufend überarbeitet – daher können einige Aufgaben oder Lösungen nicht mehr zur jeweils aktuellen Textversion passen.

Aufgabe 1 (Objekte)

4 Punkte

Was ist der Unterschied zwischen Gleichheit und Identität?

Lösung

Zwei Objekte sind identisch, wenn sie nicht zwei Objekte, sondern nur eines sind. Identische Objekte sind auch gleich; darüber hinaus können auch nicht identische Objekte gleich sein, wenn die Gleichheitsrelation entsprechend definiert ist.

Aufgabe 2 (Aliasing)

4 Punkte

Was ist ein Alias und wie entsteht er?

Lösung

Ein Alias ist ein (zweiter) Name für ein Objekt. Er entsteht bei der Wertzuweisung unter Referenzsemantik.

Aufgabe 3 (Aliasing)

8 Punkte

Skizzieren Sie das Aliasing-Problem der objektorientierten Programmierung!

Lösung

Durch Aliase kann das Verbergen eines Objekts hinter der Schnittstelle eines anderen Objekts umgangen werden. So nützt es z. B. nichts, wenn in Smalltalk die Instanzvariablen eines Objekts von außen nicht zugreifbar sind, wenn das Objekt, auf das eine Instanzvariable verweist, einen Alias außerhalb des Objekts hat: Das Objekt ist damit über den Alias zugreif- und änderbar. Lediglich die Instanzvariable ist geschützt (so daß ihr Inhalt, der Verweis auf das Objekt, nicht geändert werden kann).

Aufgabe 4 (Typsysteme)

10 Punkte

Gegeben seien die folgenden drei Klassendefinitionen.

Typ	A

Typ	B
Supertyp	A

sowie

Typ	G
Typvariablen	T
Protokoll	

```
1     x: <T> ^ <Self>
2     x ^ <T>
```

Erläutern Sie, warum bei Vorliegen der Variablendeklarationen

```
3     | ga <G[A]> gb <G[B]> |
```

weder die Zuweisung

```
4     ga := gb
```

noch

```
5     gb := ga
```

zulässig sein kann.

Lösung

Dahinter verbirgt sich die Frage, ob `G[B]` ein Subtyp von `G[A]` (ob sich also die Subtypenbeziehung von `B` zu `A` auf entsprechende Typinstanzen vom selben parametrischen Typ überträgt) oder ob das Umgekehrte der Fall ist.

Wenn die erste Zuweisung erlaubt wäre, könnte man danach `ga x: A new` (was unstrittig typkorrekt ist) ausführen und danach mit `gb x` ein Objekt vom Typ `A` erhalten, was nicht sein darf, da es dem Typ von `gb`, `G[B]`, widerspricht. Im Fall der zweiten Zuweisung könnte man, wenn man davor `ga x: A new` ausgeführt hat, danach ebenso mit `gb x` ein Objekt vom Typ `A` erhalten.

Aufgabe 5 (Variablen)

5 Punkte

In Sprachen wie Smalltalk und Java ist es schwierig, wenn nicht gar unmöglich, eine Methode zu definieren, die den Inhalt zweier Variablen vertauscht. Erläutern Sie, warum.

Lösung

Sowohl in Smalltalk als auch in Java haben Variablen zwar Referenzsemantik (in Java nur bei Objekttypen), aber beim Methodenaufruf wird nach dem Prinzip des Call by value eine Kopie des Objektpointers übergeben. Zum Vertauschen wäre aber notwendig, daß Zeiger auf die Variablen übergeben werden, so daß der tatsächliche Variableninhalt geändert werden kann. Das ist aber in beiden Sprachen nicht vorgesehen. In C# und C++ ist es hingegen möglich.

Aufgabe 6 (Abstrakte und konkrete Klassen)

4 + 2 Punkte

a) Was versteht man unter einer abstrakten Klasse und wofür wird sie eingesetzt?

b) Was ist eine konkrete Klasse?

Lösung

a) Eine abstrakte Klasse ist eine Klasse, von der man keine Instanzen bilden kann. Sie wird eingesetzt, um Generalisierungen innerhalb von Programmcode auszudrücken.

b) Klassen, die nicht abstrakt sind, die also eigene Instanzen haben können, nennt man konkret.

Aufgabe 7 (Überladen von Methodennamen)

8 Punkte

Erläutern Sie, was man unter der Überladung von Methodennamen versteht und grenzen Sie den Begriff gegen den des Überschreibens ab.

Lösung

In Sprachen wie Java und C++ kann eine Klasse mehrere Methoden gleichen Namens, aber mit verschiedenen Parametertypen besitzen. Solche Methoden nennt man dann überladen. Auch Subklassen können neue, überladene Methoden definieren. Während beim Überladen eine *neue* Methode eingeführt wird, wird beim Überschreiben eine bereits bestehende redefiniert.

Aufgabe 8 (Statisches und dynamisches Binden)

2 + 4 + 6 Punkte

a) Was versteht man unter statischem Binden im Kontext von Methodenaufrufen?

b) Was versteht man unter dynamischem Binden im Kontext von Methodenaufrufen?

c) Nach welchem Verfahren wird bei dynamischem Binden die auszuführende Methode bestimmt?

Lösung

a) Beim statischen Binden wird ein Methodenaufruf schon zur Übersetzungszeit an eine Implementierung gebunden.

b) Beim dynamischen Binden wird die tatsächlich ausgeführte Methode erst zur Laufzeit bestimmt. Dieses Verfahren wird in objektorientierten Sprachen verwendet. Die Auswahl der Methode ist in diesen Sprachen nicht nur vom Nachrichtenselektor, sondern auch vom Empfängerobjekt abhängig.

c) Wenn eine Methode auf einem Empfängerobjekt aufgerufen wird, wird zunächst geprüft, ob die Methode im zur Klasse des Empfängers gehörenden Methodenwörterbuch enthalten ist. Wird die Methode gefunden, dann wird sie ausgeführt. Wird sie nicht gefunden, wird zunächst in der direkten Superklasse der Klasse des Objekts weitergesucht und dann in deren direkter Superklasse usw. bis zur Klasse `Object`.

Aufgabe 9 (Ko- und Kontravarianz)

4 +4 + 4 + 4 + 2 Punkte

a) Was versteht man im Kontext der Überschreibung von Methoden unter Kovarianz, was unter Kontravarianz?

b) Welche Bedingungen muss man an die Überschreibung von Methoden stellen, damit der überschreibende Typ zuweisungskompatibel zum überschriebenen bleibt?

c) Sind bei der Redefinition in Java Kovarianz und/oder Kontravarianz zulässig? Wenn ja, welche Art der Varianz ist an welchen Stellen zulässig?

d) Sind bei der Redefinition in Eiffel Kovarianz und/oder Kontravarianz zulässig? Wenn ja, welche Art der Varianz ist an welchen Stellen zulässig?

e) Welches Problem handelt sich Eiffel damit ein?

Lösung

a) Seien `A` und `B` zwei Typen, wobei `B` ein Subtyp von `A` ist. Sei m eine Methode in `A`, die in `B` redefiniert wird. Sei `PA` ein Parametertyp von `m` in `A` und `PB` der entsprechende Parametertyp von `m` in `B`. Wenn `PB` ein Supertyp von `PA` ist, so spricht man von Kontravarianz; die entsprechenden Parametertypen variieren gegenläufig zur Subtypbeziehung von `A` und `B`. Wenn `PB` ein Subtyp von `PA` ist, so spricht man von Kovarianz; die entsprechenden Parametertypen variieren gleichgerichtet zur Subtypbeziehung von `A` und `B`.

b) Wenn die Eingabeparameter einer redefinierten Methode kontravariant und die Ausgabeparameter kovariant redefiniert werden, dann bleibt Zuweisungskompatibilität des redefinierenden Typen mit dem redefinierten erhalten.

c) In Java ist keine Kontravarianz und auch keine Kovarianz für die Eingabeparameter erlaubt, d.h. Eingabeparameter können nur invariant redefiniert werden. Ausgabeparameter können dagegen ab der Java-Version 5.0 kovariant redefiniert werden.

d) In Eiffel wird beim Redefinieren von Methoden verlangt, dass die entsprechenden Parametertypen gleich sind oder kovariant redefiniert werden. Kontravarianz ist nicht erlaubt.

e) Die Substituierbarkeit in Gegenwart von kovarianten Überschreibungen ist eingeschränkt.

Aufgabe 10 (Verhaltenskonformität)

6 + 4 Punkte

Im Kurs wurde neben den syntaktischen Regeln, die ein Subtyp einhalten muß, auch von konformem Verhalten eines Subtyps gesprochen.

a) Erläutern Sie, was man unter Verhaltenskonformität im Kontext von Subtyping versteht.

b) Geben Sie ein Beispiel für einen Subtyp an, der sich nicht konform zu seinem Supertyp verhält.

Lösung

a) Die Grundregel für Subtyping besagt, daß ein Objekt von einem Subtyp an allen Programmstellen vorkommen kann, an denen Supertyp-Objekte zulässig sind. Dazu muß der Subtyp die Eigenschaften des Supertyps besitzen. Diese Eigenschaften betreffen sowohl syntaktische Bedingungen, beispielsweise für die Deklaration von Attributen und Methodensignaturen, als auch das Verhalten des Subtyps, das durch seine Methodenimplementierungen bestimmt wird. Ein Subtyp verhält sich konform zu seinem Supertyp, wenn seine Methodenimplementierungen dasselbe Verhalten aufweisen, wie durch seinen Supertyp vorgegeben.

b) Im Folgenden setzen wir die Implementierung eines Stapels in Form eines Typs `Stack` mit dem üblichen Verhalten voraus: Seine `push`-Methode legt das als Parameter übergebene Objekt oben auf dem Stack ab, seine `pop`-Methode nimmt das oberste Element vom Stack und gibt es zurück. Das Element, das zuletzt auf dem Stack abgelegt wurde, wird also mittels `pop` als erstes wieder vom Stack herunter genommen. Die `top`-Methode der Klasse `Stack` liefert das oberste Stack-Element zurück, läßt es aber, im Gegensatz zu `pop`, auf dem Stack. Bilden wir jetzt einen Subtyp von `Stack`, der die Implementierung der Methode `pop` so abändert, daß sie dasjenige Element zurückgibt und aus dem Stack löscht, das ganz unten auf dem Stack liegt, realisiert er eine FIFO-Queue. Er verhält er sich also nicht konform zu seinem Supertyp `Stack`.

Aufgabe 11 (Polymorphie)

4 + 12 Punkte

In dieser Aufgabe geht es um die verschiedenen Varianten von Polymorphie, die Sie im Kurs kennengelernt haben.

a) Erläutern Sie zunächst allgemein den Begriff der Polymorphie. Was versteht man darunter im Kontext von Programmiersprachen?

b) Nennen Sie drei im Kurs genannte Formen von Polymorphie und erklären Sie deren Bedeutung.

Lösung

a) Polymorphie bedeutet Vielgestaltigkeit. Im Zusammenhang mit Programmiersprachen spricht man von Polymorphie, wenn Programmkonstrukte oder Programmteile für Objekte (bzw. Werte) mehrerer Typen verwendbar sind.

b) Im Kurs werden u. a. folgende Arten von Polymorphie beschrieben:

1. *Inklusionspolymorphie:* Programmkonstrukte bzw. Programmteile, die für Objekte eines Typs `T` formuliert sind, sind unverändert auch für Objekte eines Typs `S` verwendbar, wenn `S` ein Subtyp von `T` ist. D. h., Objekte vom Typ `S` können an allen Programmstellen verwendet werden, an denen Objekte vom Typ `T` zulässig sind. Insbesondere können mit Hilfe der Inklusionspolymorphie inhomogene Behälter realisiert werden, d. h. Behälter, die Objekte verschiedener Typen enthalten können.

2. *Parametrische Polymorphie:* Dies ist eine Art der Polymorphie, bei der man Programmkonstrukte oder -teile — in objektorientiertem Kontext sind das dann meist Klassen und Methoden — mit Typen parametrisiert. So kann man z. B. Behältertypen mit ihren Elementtypen parametrisieren. Der einmal für diesen Behälter geschriebene Programmcode kann durch einfaches Instanziieren des Typparameters für jeden beliebigen Elementtyp verwendet werden. Parametrische Polymorphie bietet weniger Flexibilität als Inklusionspolymorphie. Beispielsweise lassen sich mit Sprachen, die nur parametrische Polymorphie unterstützen, keine inhomogenen Behälter realisieren. Andererseits gestattet parametrische Polymorphie bei vielen Anwendungen eine leistungsfähigere Typanalyse zur Übersetzungszeit.

3. *Beschränkt parametrische Polymorphie:* Beschränkt parametrische Polymorphie verbindet Inklusionspolymorphie mit parametrischer Polymorphie. Sie ermöglicht es, Typparameter von parametrischen Typen durch Angabe von Supertypen zu beschränken. Für einen durch einen Typ `T` beschränkten Typparameter `E` dürfen nur solche Typen `S` eingesetzt werden, die Subtypen von `T` sind. Bei der Implementierung solcher beschränkt parametrischer Typen durch Klassen kann dann das Wissen ausgenutzt werden, daß Objekte eines Typs `S`, der für einen

Typparameter E eingesetzt wird, der selbst nach oben durch einen Supertyp T beschränkt ist, mindestens über alle Eigenschaften von T verfügen. Diese Eigenschaften können dann in der Implementierung benutzt werden.

Aufgabe 12 (Probleme der objektorientierten Programmierung)

8 Punkte

Was versteht der Kurs 01814 unter dem Begriff *Fragile-base-class Problem*?

Lösung

Allgemein versteht man darunter, daß Änderungen einer Klasse in der Vererbungshierarchie neben den für die geänderte Klasse gewünschten Effekten auch unerwünschte Effekte in anderen Klassen haben können. Diese unerwünschten Effekte können offensichtlich (die Änderungen werden an eine Subklasse vererbt, wo sie aber nicht sinnvoll sind) oder verborgen sein. Letzteres ist häufig bei der sog. offenen Rekursion der Fall, nämlich wenn eine zuvor (durch dynamisches Binden) aus Methoden der Superklasse aufgerufene, überschreibende Methode in der Subklasse nicht mehr aufgerufen wird.