Objektorientierte Softwareentwicklung

- Softwaretechnik -

SS 2010

Jähnichen
Mertgen

Ausgabe: Mittwoch, 28.04.10
Abgabe: Freitag, 07.05.10

1. Übungsblatt: LOS

Ziel dieser Aufgabe ist es, einen Interpreter für eine objektorientierte Programmiersprache zu implementieren: **LOS** = *Lua Objekt Sprache*. Die Spezifikation der Sprache wird in mehreren Schritten vorgestellt. Den ersten Schritt, welcher in dieser Aufgabe umzusetzen ist, machen wir mit der Einführung von Objekten und Klassen inklusive einfacher Vererbung und Type-checking. LOS soll auf Basis der aktuellen Lua-Version 5.1.4 entwickelt werden.

Objekte, Klassen und Vererbung

Objekte und Klassen in LOS sollen durch Lua-Tables implementiert werden. Objekte speichern nur ihren Zustand, ihre Methoden (= Funktionen mit implizitem erstem Argument `self`) sollen in Klassen ausgelagert werden.

Klassen

Klassen werden mit folgender Syntax erzeugt:

```
1 Class{'MyClass', SuperClass,
2   attribute1 = String,
3   attribute2 = MyClass
4 }
```

`Class` ist kein Schlüsselwort von Lua, sondern eines von eurer Sprache LOS. Die obige Syntax ist ein Funktionsaufruf mit einer vier-elementigen Liste als Argument - `f{}` ist in Lua lediglich eine Abkürzung für `f({})`.

1. *MyClass* ist der Name der zu definierenden Klasse. Die erzeugte Struktur soll anschließend unter diesem Namen global verfügbar sein.

2. Der zweite Bestandteil (hier `SuperClass`) ist optional. Wenn hier eine bereits definierte Klasse angegeben wird, dann soll diese bei der neuen Klasse als Oberklasse (`_super`)

eingetragen werden. Zyklen in der Vererbungshierarchie sind nicht erlaubt. Wird ein Wert übergeben, der keine gültige Oberklasse darstellt, soll ein Fehler ausgegeben werden.

3. Im Anschluss kann optional eine beliebige Anzahl von Attributen deklariert werden. Eine Attributdeklaration besteht aus einem Schlüssel (Name des Attributs) und einem Wert (Typ des Attributs). Als Typ für ein Attribut kommen die Basistypen *String*, *Number* und *Boolean* in Frage, sowie alle selbstdefinierten Klassen. Stellt ein Schlüssel-Wert-Paar keine gültige Attributsdeklaration dar, soll ein Fehler ausgegeben werden.

    - Die genannten Basistypen existieren in Lua natürlich nicht als "Typ". Sie sollen lediglich durch einen Textstring `'string'`, `'number'` bzw. `'boolean'` kodiert werden. Überlegt euch: was stellt `String` im vorangehenden Beispiel in Lua eigentlich dar?
    - Der Typ `MyClass` im Beispiel existiert ebenfalls noch nicht, denn wir legen die Klasse in diesem Moment ja erst an. Eine einfache Lösung wäre, die Klasse vor dem Aufruf von `Class` als leere Table anzulegen. Wie kann man diesen Vorgang automatisieren?
    - Sollte eine Klasse ein Attribut gleichen Namens wie eine ihrer Oberklassen deklarieren, so muss gelten, dass die Attribute mit dem gleichen Typ deklariert sind (die erneute Deklaration ist redundant und hat praktisch keinen Effekt), andernfalls soll ein Fehler ausgegeben werden.

Im Anschluss an die Klassendefinition können der Klasse Methoden hinzugefügt werden. Dies geschieht mit folgender Syntax (Beispiel):

```
1 function MyClass:hello()
2   print('hello')
3 end
```

Es dürfen mehrere Methoden gleichen Namens hinzugefügt werden - was allerdings nur bedeutet, dass die jeweils zuletzt definierte Methode die vorigen komplett ersetzt.

Objekterzeugung

Zur Instanziierung von Objekten soll an Klassen die Methode `new` als Standard-Konstruktor zur Verfügung stehen:

```
1 obj = MyClass:new()
```

Der Konstruktor wird automatisch von LOS bereit gestellt und muss nicht vom Programmierer eines LOS-Programms definiert werden. Weitere Konstruktoren werden in LOS nicht unterstützt.

Ein spezielles Feld `_class` im Objekt soll die Referenz auf seine Klasse halten. (Für `_class` wie auch für alle anderen "*_versteckten*" Felder gilt die Konvention, dass sie nur zu internen Zwecken dienen. Der Programmierer eines LOS-Programms hat keine Kenntnis von ihnen.)

Nach der Initialisierung sollen sämtliche Attribut-Felder eines Objekts mit Default-Werten für den jeweiligen Typ belegt sein:

- 0 für *Number*
- Leerer String für *String*
- `false` für *Boolean*
- `nil` für Objekt-Referenzen

Attribute und Methoden

Für jede Zuweisung an ein Attribut soll gelten, dass die Klasse des Objekts das Attribut deklariert oder von einer Oberklasse geerbt haben muss und dass der zuzuweisende Wert dem deklarierten Typ oder einem konformen Subtyp entspricht (erbende Subklassen sind konform zu ihren Oberklassen). Andernfalls soll ein Fehler ausgegeben werden.

Wird ein Feature eines Objekts referenziert, so muss zwischen Attributen und Methoden unterschieden werden. Attribute (die Werte, mit denen Attributvariablen belegt sind) finden sich beim Objekt selbst, Methoden in der Klasse. Bei gleichem Namen sollen Attribute Vorrang vor Methoden haben. Wird ein Feature weder im Objekt noch in der direkten Klasse gefunden, soll die Liste der Oberklassen durchsucht werden, bis das Feature gefunden wird.

Fehler und Ausgabe

Bei korrekten Programmen soll LOS keine Ausgaben produzieren, welche nicht vom Anwenderprogramm ausgelöst werden (d.h. insbesondere, dass keine Debugging-Informationen ausgegeben werden). Bei Programmen mit Verletzungen der LOS-Spezifikation soll mit der Funktion `error` eine aussagekräftige Fehlermeldung ausgegeben werden, welche über Problem und Ursache informiert. Darüber hinaus ist keine Fehlerbehandlung gefordert.

Bearbeitung und Abgabe

Die Aufgabe soll in Gruppen von 3 Personen bearbeitet werden. Die Abgabe erfolgt in eurem Gruppenforum. Legt dazu einen neuen Beitrag mit dem Betreff *Abgabe Übung 1* an und fügt die Dateien gepackt in einem einzelnen `zip`-Archiv als Anhang ein. Im Kopf jeder Datei müssen (als Kommentar) Namen und Matrikelnummern aller aktiv mitarbeitenden

Gruppenmitglieder stehen. Mit der Aufgabe wird ein kleines Testprogramm bereit gestellt, das natürlich mit eurer LOS-Version ausführbar sein muss. *Wir empfehlen, dass ihr eure LOS-Implementierung darüber hinaus mit eigenen Tests prüft! Wir werden das tun* ☺