

Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung

Aufgabe 6-1

Überprüfen und Auswerten von Ausdrücken

Präsenz

In dieser Aufgabe sollen Sie Ausdrücke auf Korrektheit untersuchen und gegebenenfalls auswerten. Als Grundlage für Ausdrücke im Kontext von Klassendeklarationen verwenden wir folgende EBNF-Grammatik mit der entsprechenden Erweiterung von `Value` um das Terminalsymbol `"null"`:

```
Expression = Variable |  
            Value |  
            Expression BinOp Expression |  
            UnOp Expression |  
            "(" Expression ")" |  
            MethodInvocation |  
            InstanceCreation
```

```
Variable = NamedVariable | FieldAccess  
NamedVariable = Identifier  
FieldAccess = Expression "." Identifier
```

```
MethodInvocation = Expression "." Identifier "(" [ActualParameters] ")"  
ActualParameters = Expression {" ," Expression}
```

```
InstanceCreation = ClassInstanceCreation  
ClassInstanceCreation = "new" ClassType "(" [ActualParameters] ")"
```

Wir verwenden die Klassendeklarationen von `Point` und `Line` aus der Vorlesung. Seien `Point p;`, `Line l;` und `int i;` lokale Variablendeklarationen. Gegeben seien folgende Ausdrücke:

1. `l.x`
2. `p.move(1,2) == null`
3. `l.length()`
4. `p == null`
5. `i == null`
6. `new point(3,4)`
7. `(new Point(1,1)).x`

- a) Geben Sie an, welche dieser Ausdrücke syntaktisch nicht korrekt und welche Ausdrücke syntaktisch korrekt, aber nicht typkorrekt sind (jeweils mit Begründung). Für typkorrekte Ausdrücke ist deren Typ anzugeben.

b) Gegeben sei der Zustand (σ, η) mit

$$\begin{aligned}\sigma &= [(i, 3), (p, @123), (l, @789)] \\ \eta &= \{< (@123, \text{Point}), [(x, 2), (y, 4)] >, < (@456, \text{Point}), [(x, 0), (y, 0)] >, \\ &\quad < (@789, \text{Line}), [(\text{start}, @123), (\text{end}, @456)] >\}\end{aligned}$$

i) Geben Sie die grafische Repräsentation dieses Zustands mit Zeigern an.

ii) Werten Sie alle syntaktisch korrekten und typkorrekten Ausdrücke bezüglich dieses Zustands aus.

Aufgabe 6-2

Überprüfen und Auswerten von Ausdrücken

Hausaufgabe

In dieser Aufgabe sollen Sie Ausdrücke auf Korrektheit untersuchen und gegebenenfalls auswerten. Als Grundlage für Ausdrücke im Kontext von Klassendeklarationen verwenden wir die EBNF-Grammatik aus Aufgabe 6-1.

Wir verwenden die Klassendeklarationen von `Point` und `Line` aus der Vorlesung. Seien `Point p;`, `Line l;` und `int i;` lokale Variablendeklarationen. Gegeben seien folgende Ausdrücke:

1. `i.y`
2. `p.i`
3. `l.start.y`
4. `p.GetX()`
5. `l.move(p)`
6. `i == p`
7. `new Line(3,4)`
8. `Point(3,4)`
9. `(new Line(new Point(i,i), p)).end.getY()`

a) Geben Sie an, welche dieser Ausdrücke syntaktisch nicht korrekt und welche Ausdrücke syntaktisch korrekt, aber nicht typkorrekt sind (jeweils mit Begründung). Für typkorrekte Ausdrücke ist deren Typ anzugeben.

b) Gegeben sei der Zustand (σ, η) mit

$$\begin{aligned}\sigma &= [(i, 3), (p, @123), (l, @789)] \\ \eta &= \{< (@123, \text{Point}), [(x, 2), (y, 4)] >, < (@456, \text{Point}), [(x, 0), (y, 0)] >, \\ &\quad < (@789, \text{Line}), [(\text{start}, @123), (\text{end}, @456)] >\}\end{aligned}$$

Werten Sie alle syntaktisch korrekten und typkorrekten Ausdrücke bezüglich dieses Zustands aus.

Aufgabe 6-3

Deklaration von Klassen und Methoden

Präsenz

In dieser Aufgabe soll eine Klasse `Figur` deklariert werden, die einfache geometrische Figuren repräsentiert. Beispielsweise können ein Rechteck oder ein Kreis eine geometrische Figur sein. Jede geometrische Figur hat einen Mittelpunkt vom Typ `Point` und eine Farbe vom Typ `String`. Außerdem kann man angeben, ob die Figur mit dieser Farbe ausgefüllt ist.

a) Deklarieren Sie eine Klasse `Figur`, die oben genannte Eigenschaften realisiert. Die Klasse soll die folgenden Konstruktoren und zwei Methoden anbieten:

- einen Konstruktor mit formalen Parametern zur Initialisierung der Attribute,
- einen weiteren Konstruktor mit formalen Parametern zur Initialisierung der Attribute, so dass für den Mittelpunkt als Parameter dessen x- und y-Koordinaten übergeben werden,
- eine Methode **bewegen** ohne Rückgabetypp, die den Mittelpunkt einer Figur soweit verschiebt, wie es zwei formale Parameter **dx** und **dy** vom Typ **int** fordern,
- eine Methode **getMittelpunkt()** mit dem Rückgabetypp **Point**, die den Mittelpunkt einer Figur zurückgibt.

*Hinweis: Um die Klasse **Point** verwenden zu können, speichern Sie die Datei **Point.java** aus dem ZIP-Archiv des Übungsblatts im gleichen Ordner wie Ihre Klasse **Figur**. Diese Datei enthält die Klassendeklaration der Klasse **Point**, wie sie in der Vorlesung gezeigt wurde.*

- b) Mit folgendem Programmcode können Sie Ihre Klasse **Figur** testen (die Datei **FigurTest.java** muss im gleichen Ordner wie Ihre Klasse **Figur** gespeichert werden):

```

1  /**
2   * Diese Klasse testet die Implementierung der Klasse Figur.
3   * Es wird eine Figur figur1 erstellt sowie eine Figur figur2.
4   * Anschliessend wird die Figur figur1 bewegt.
5   */
6  public class FigurTest {
7      public static void main(String[] args) {
8          Point p = new Point(2, 1);
9          Figur figur1 = new Figur(p, "rot", true);
10         Figur figur2 = new Figur(p, "schwarz", false);
11
12         System.out.println("figur1: Mittelpunkt=("
13             + figur1.getMittelpunkt().getX() + ","
14             + figur1.getMittelpunkt().getY() + ")");
15         System.out.println("figur2: Mittelpunkt=("
16             + figur2.getMittelpunkt().getX() + ","
17             + figur2.getMittelpunkt().getY() + ")");
18
19         figur1.bewegen(2, 2);
20
21         System.out.println("figur1: Mittelpunkt=("
22             + figur1.getMittelpunkt().getX() + ","
23             + figur1.getMittelpunkt().getY() + ")");
24         System.out.println("figur2: Mittelpunkt=("
25             + figur2.getMittelpunkt().getX() + ","
26             + figur2.getMittelpunkt().getY() + ")");
27     }
28 }

```

Geben Sie für dieses Programm jeweils den Zustand des Speichers (Stack und Heap) nach Zeile 10 und nach Zeile 19 an. Sie können die Zustände entweder in der abstrakten Syntax mit Objektreferenzen Ihrer Wahl angeben (siehe Aufgabe 6-1 und 6-2) oder die grafische Repräsentation mit Zeigern verwenden.

- c) Welches Problem fällt Ihnen bei der Analyse der Zustandsänderung auf und wie können Sie es beheben?

Aufgabe 6-4

Deklaration von Klassen und Methoden

Hausaufgabe

In dieser Aufgabe soll eine Klasse **Fahrzeug** deklariert werden, die beliebige Fortbewegungsmittel repräsentiert. Beispielsweise können ein Fahrrad, ein Auto oder eine Eisenbahn ein Fahrzeug sein. Jedes Fahrzeug hat eine aktuelle Position vom Typ **Point**, eine bestimmte Anzahl an Rädern, ein Leergewicht und eine aktuelle Geschwindigkeit.

- a) Deklarieren Sie eine Klasse **Fahrzeug**, die oben genannte Eigenschaften definiert. Die Klasse soll den folgenden Konstruktor und drei Methoden anbieten:

- einen Konstruktor mit formalen Parametern zur Initialisierung der Attribute, so dass für den Mittelpunkt als Parameter dessen x- und y-Koordinaten übergeben werden,
- eine Methode **beschleunigen** ohne Rückgabety, die die aktuelle Geschwindigkeit um einen bestimmten Wert erhöht bzw. erniedrigt (bei negativer Beschleunigung). Der Wert soll als Parameter übergeben werden können,
- eine Methode **fahren** ohne Rückgabety, die das Fahrzeug auf eine neue Position verschiebt, wie es zwei formale Parameter **dx** und **dy** vom Typ **int** fordern,
- eine Methode **getAktuelleGeschwindigkeit** mit dem Rückgabety **double**, die die aktuelle Geschwindigkeit des Fahrzeugs zurückgibt.

*Hinweis: Um die Klasse **Point** verwenden zu können, speichern Sie die Datei **Point.java** aus dem ZIP-Archiv des Übungsblatts im gleichen Ordner wie Ihre Klasse **Fahrzeug**. Diese Datei enthält die Klassendeklaration der Klasse **Point**, wie sie in der Vorlesung gezeigt wurde.*

- b) Mit folgendem Programmcode können Sie Ihre Klasse **Fahrzeug** testen (die Datei **FahrzeugTest.java** muss im gleichen Ordner wie Ihre Klasse **Fahrzeug** gespeichert werden):

```

1  /**
2   * Diese Klasse testet die Implementierung der Klasse Fahrzeug.
3   * Es wird ein Fahrzeug auto erstellt sowie ein Fahrzeug fahrrad.
4   * Anschliessend wird das Fahrzeug auto beschleunigt.
5   */
6  public class FahrzeugTest {
7      public static void main(String[] args) {
8          Fahrzeug auto = new Fahrzeug(1, 2, 4, 1234.5, 100.0);
9          Fahrzeug fahrrad = new Fahrzeug(3, 4, 2, 33.3, 18.3);
10
11         System.out.println("Auto: aktuelle Geschwindigkeit="
12             + auto.getAktuelleGeschwindigkeit());
13         System.out.println("Fahrrad: aktuelle Geschwindigkeit="
14             + fahrrad.getAktuelleGeschwindigkeit());
15
16         auto.beschleunigen(11.1);
17
18         System.out.println("Auto: aktuelle Geschwindigkeit="
19             + auto.getAktuelleGeschwindigkeit());
20         System.out.println("Fahrrad: aktuelle Geschwindigkeit="
21             + fahrrad.getAktuelleGeschwindigkeit());
22
23         fahrrad.fahren(5, 3);
24     }
25 }

```

Geben Sie für dieses Programm jeweils den Zustand des Speichers (Stack und Heap) nach Zeile 9, nach Zeile 16 und nach Zeile 23 an. Sie können die Zustände entweder in der abstrakten Syntax angeben (siehe Aufgabe 6-1 und 6-2) oder die grafische Repräsentation mit Zeigern von Hand malen und abfotografieren oder ein beliebiges Zeichenprogramm (z.B. PowerPoint) verwenden. Bitte geben Sie nur **.txt**-, **.jpg**- oder **.pdf**-Dateien ab!

Besprechung der Präsenzaufgaben in den Übungen ab 23.11.2018. Abgabe der Hausaufgaben bis Mittwoch, 05.12.2018, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung).

- Erstellen Sie zu jeder Aufgabe eine Klasse, die den Namen trägt, der in der Aufgabe gefordert ist.
- Der Java-Code in ihrer Abgabe muss als separate **.java**-Datei abgegeben werden. Wir benötigen **nicht** Ihre **.class**-Dateien.