Programmierung für Mathematik HS12
Übung 2

27. September 2012

1 Aufgabe: OOP

1.1 Ziele

- Die verschiedenen Initialisierungen kennen.
- Lernen wie Variablen inkrementiert werden.
- Die verschiedenen Zuweisungen kennen.
- Typumwandlungen kennen lernen.

1.2 Aufgabenstellung

1.2.1 Verschiedene Deklarationen und Initialisierungen

Betrachten Sie die nächsten Code Snipets und bestimmen Sie welche syntaktisch fehlerhaft sind. Wie lautet die Ausgabe für die korrekten Code Fragmente? Begründen Sie Ihre Antwort.

Snipet1

```
int a = 2 ;
```

Snipet2

```
int a;
int a = 4;
```

Snipet3

```
int a ;
a=1++;
```

Snipet4

```
int a;
a = 9;
a++;
```

Snipet5

```
String s = "w" + "oooo".toUpperCase() + "t";
```

Snipet6

```
in a,b:
a  = b = 12;
b= 3+10*10;
```

Snipet7

```
String s;
String w;
s=w;
```

Snipet8

```
String num;
int i=10;
num=i;
```

1.2.2 Inkrementieren

Ordnen Sie die einzelnen Zeilen um korrekte Zuweisungen zu erhalten:

=	a	+	1
+	a	+	
	a	+1	=

1.2.3 Zuweisungen

Einer Variabel kann gleich bei der Deklaration oder zu einem späteren Zeitpunkt einen Wert zugeordnet werden. Schreiben Sie für beide Fälle ein Programm.

1.2.4 Typumwandlungen

Bestimmte primitive Typen sind zueinander soweit kompatibel, dass einer Variabel eines Typs der Wert einer Variabel eines anderen Typs zugewiesen werden kann. Finden Sie heraus welche Typen aus folgender Liste einander zugewiesen werden können.

- int
- long
- String
- float
- boolean
- double
- byte
- char

Überlegen Sie sich ob die so definierte Relation zwischen primitive Typen symmetrisch oder transitiv ist?

1.2.5 Was ist der Hauptunterschied zwischen primitiven & Referenztypen?

Erklären Sie den Unterschied zwischen primitiven und Referenztypen anhand eines Beispieles.

2 Aufgabe: Klassen

2.1 Ziele

- Eine Klasse mit Zustand (Instanzvariablen) und Methoden mit Übergabewerten implementieren können.
- Das Verhalten der erstellten Klasse mittels eines TestDrivers überprüfen.

2.1.1 Aufgabenstellung

Schreiben Sie eine Klasse mit dem Namen *Adder* und den Methoden *reset()*, *add()* und *sum()*.

- Die Methode *add()* soll einen einzigen Parameter vom Typ *long* erwarten und diesen zur Summe aller bisher erhaltenen Werte hinzuaddieren.
- *sum()* soll die Summe der bisher erhaltenen Werte auf dem Bildschirm ausgeben.
- *reset()* setzt das Object zurück, sprich die Summe der bisher erhaltenen Werte werden wieder auf 0 gesetzt.

Erzeugen Sie in der *main()*-Methode eines TestDrivers einige Objekte der Klasse *Adder* und testen Sie die *reset()*, *add()* und *sum()* Methoden.

Adder Klasse

```
class Adder {

  ...

    void add(...) {
      ...
    }

    void reset() {
      ...
    }

    void sum() {
      ...
    }

}
```

3 Aufgabe: Grundzüge der Objektorientierten Programmierung

3.1 Ziele

- Die Begriffe der Objektorientierten Programmierung kennen lernen.

3.2 Aufgabenstellung

3.2.1 Begriffe

Erklären Sie folgende Begriffe in jeweils einen Satz:

- Klasse
- Methode
- Instanz
- Datenkapselung