Einführung in die Programmierung für NF MI

Übung 07

Inhalt

- Wiederholung Kommentare
- Wiederholung Arrays
- Einführung in Objekte

Wiederholung Kommentare

- Kommentare immer vor den zu kommentierenden Inhalt (ohne Leerzeile)
- Javadoc-Kommentare /** */
  vor jede Klasse und jede Funktion
- Mehrzeilige Kommentare /* */
  für ergänzende Erklärungen
- Einzeilige Kommentare //
  für kurze Anmerkungen zu einer Zeile

Wiederholung Arrays

- Fragen?

- Was ist ein Array?
- Was sind besondere Eigenschaften von Arrays (generell und in Java)?
- Wie funktionieren Array in Java? (Erstellung und Zugriff)

Einführung in Objekte

- Durch Objekte können reale Dinge oder Datenstrukturen in der Programmierung abgebildet werden
- Objekte haben dabei eigene Attribute (Variablen) und Methoden / Funktionen
- Attribute von Objekten können natürlich selbst Objekte sein

Einführung in Objekte

- Beispiel-Objekt: Kugelschreiber
- Attribut
  - ausgefahren (bool)
- Methoden
  - istAusgefahren (bool)
    gibt Zustand von ausgefahren zurück
  - fahreKugelschreibermineEinAus (void)
    fährt je nach Zustand die Mine ein oder aus

Einführung in Objekte

- Ein großer Vorteil der objektorientierten Programmierung ist die Trennung von logischen Elementen
- So ist es im Kugelschreiberbeispiel z.B. beim Aufruf der Funktionen nicht relevant, ob es sich um einen Kugelschreiber mit Druckkopf oder zum Drehen handelt
  - dies wird intern im Objekt gehandelt

Einführung in Objekte

- Beispielobjekt: Adresse
- Eine Adresse besteht aus mehreren Attributen, z.B.
  - Straße
  - Hausnummer
  - Postleitzahl
  - Ort

Einführung in Objekte

- Beispielobjekt: Punkt
- Um einen Punkt im zweidimensionalen Raum zu definieren werden zwei Attribute benötigt:
  - X-Koordinate
  - Y-Koordinate

Einführung in Objekte

- Beispielobjekt: Kreis
- Um einen Kreis im zweidimensionalen Raum zu definieren werden drei Attribute benötigt:
  - X-Koordinate
  - Y-Koordinate
  - Radius
- Oder
  - Den bereits definierten Punkt und Radius

Klassen

- Ein Objekt bezeichnet ein einzelnes „Ding“, z.B. einen definierten Kugelschreiber
- Eine Klasse ist die Verallgemeinerung (z.B. alle Kugelschreiber), die beschreibt, welche Attribute ein Objekt dieser Klasse haben kann, ohne diesen Werte zuzuweisen
- Ein Objekt ist eine **Instanz seiner Klasse**

Klassen

- Klassen geben also den Rahmen für die Objekte vor
- Ein Objekt einer Klasse wird mit dem **new-Operator** erzeugt
- Beispiel:

```
String zeichenkette =
   new String ("Hallo");
```

Konstruktoren

- Durch die Verwendung des new-Operators wird der **Konstruktor** einer Klasse aufgerufen
- Der Konstruktor ist eine spezielle Methode einer Klasse und dient zur Initialisierung von Attributen eines Objektes
- Der Konstruktor ist wie eine Methode aufgebaut, die den Namen der Klasse hat

Konstruktoren

- Eine Klasse kann mehrere Konstruktoren haben, diese müssen sich jedoch in ihren Übergabeparametern unterscheiden
- Welcher der Konstruktoren dann verwendet wird, hängt von der Art und Anzahl der Parameter ab, die bei der Erzeugung übergeben werden
- Datentyp und Anzahl muss übereinstimmen

Zugriff

- Um auf die Attribute oder Methoden eines Objektes zuzugreifen bedient man sich der Punktnotation
- Beispiel

```
String zeichenkette =
  new String ("Hallo");

int laenge =
  zeichenkette.length();
```

Sichtbarkeiten

- Wenn Variablen in Java ohne spezielle Anweisung erzeugt werden, sind sie im gesamten Package sichtbar
- Die Sichtbarkeit kann und sollte durch die Attribute *public*, *private* und *protected* beeinflusst werden

Sichtbarkeiten

Sichtbarkeit	Innerhalb der Package	Abgeleitete Klassen	Außerhalb der Package
private	unsichtbar	unsichtbar	unsichtbar
default	sichtbar	unsichtbar	unsichtbar
protected	sichtbar	sichtbar	unsichtbar
public	sichtbar	sichtbar	sichtbar

Getter- und Setter-Methoden

- Normalerweise werden innerhalb eines Objektes alle Variablen auf die Sichtbarkeit *private* gesetzt
- Damit sind sie „von außen“ nicht beeinflussbar und vor Zugriff geschützt
- Um dennoch auf sie zugreifen zu können (falls nötig), werden sogenannte Getter- und Setter-Methoden benutzt

Getter- und Setter-Methoden

- Diese Methoden haben nur einen Sinn:
  - Getter-Methoden geben den Wert ihrer Variable zurück
  - Setter-Methoden ändern den Wert ihrer Variable auf den übergebenen Wert
- In Sonderfällen kann es sein, dass in den Getter- und Setter-Methoden weitere Dinge passieren, z.B. Observer benachrichtigen

Getter- und Setter-Methoden

- Namenskonvention ist
  - variablenTyp **getVariablenname**()
  - void **setVariablenname**(variablenTyp neuerWert)
- Getter- und Setter-Methoden können in Eclipse automatisch erzeugt werden
- Wenn Übergabeparameter und Klassenvariable denselben Namen haben, wird mit `this.` auf die Klassenvariable zugegriffen

Getter- und Setter-Methoden

- Ein Beispiel für Konstruktoren sowie Getter- und Setter-Methoden bietet die Klasse *Punkt.java*
- Ein weiteres Beispiel, wie Objekte als Attribute von Objekten verwendet werden, bietet die Klasse *Kreis.java*
- Die Klasse *Beispiel.java* bietet Beispiele für den Umgang mit Objekten

Interne Funktionen

- Der Sinn von Objekten besteht natürlich nicht nur darin, sie zu erstellen, sondern mit ihnen arbeiten zu können
- Von anderen Klassen sind dazu nur die notwendigen Methoden sichtbar, was in der Klasse aber geschieht bleibt verborgen
- Dieses Prinzip wird als „Data Hiding“ bezeichnet

Interne Funktionen

- Falls Hilfsmethoden verwendet werden, werden diese als *private* gekennzeichnet
- Beispiel und Inhalt dieser Übung:
  - Die als Übungsaufgabe programmierte Klasse *Primzahlcheck* wird dazu verwendet, einem Nutzer auszugeben, ob eine Zahl prim ist oder nicht
  - Dazu werden interne Hilfsfunktionen von sichtbaren Zugriffsfunktionen getrennt

Fragen zum Übungsblatt?