Javakurs für Anfänger

Einheit 04: Variablenzugriff und Eingaben

Kyrill Schmid
Lehrstuhl für Mobile und Verteilte Systeme

Info

Nächste Woche am **21.11.2018 muss der Kurs einmalig entfallen**. Wir sehen uns in der Woche darauf am 28.11. wieder!

Heutige Agenda

1. Teil: Zugriff auf Instanzvariablen

- Praktisches Beispiel: Student
- Verwendung von Getter und Setter
- Programmieraufgabe: Kreisberechnung

2. Teil: Nutzereingaben

- EVA Prinzip
- Eingaben über die Konsole
- Eingaben über GUI
- Programmieraufgabe: Kreisberechnung II

Lernziele

- Mehr Übung mit Objekten und Methoden
- Getter und Setter kennenlernen
- Nutzereingaben implementieren

Eigene Aufgabe (letzte Stunde)

Versuchen Sie nun selbstständig die folgende Programmieraufgabe:

Schreiben Sie eine Klasse ***Student*** mit den folgenden Attributen und Methoden:

- Attribute:
  - Name (`String`)
  - Semester (`int`)
- Methoden:
  - `public int welchesSemester()`
    - Gibt das aktuelle Semester des Studenten zurück
  - `public void lernen()`
    - Soll auf der Konsole ausgeben: „… lernt gerade“.

Schreiben Sie innerhalb der Klasse eine Main-Methode und erzeugen Sie darin 2 Studenten mit jeweils einem Namen und der Semesterzahl.

Lassen Sie sich für beide Studenten die Semesterzahl auf der Konsole ausgeben.

Rufen Sie für beide Studenten die Methode lernen() auf.

Auf Instanzvariablen zugreifen

Zur Erinnerung:

- Ein direkter Zugriff auf Instanzvariablen von außerhalb der Klasse sollte vermieden werden
  - Durch die `private` Deklaration wird dieser Zugriff von Außen verboten

- *Was passiert also, wenn die Main-Methode nicht innerhalb der Klasse liegt?*

```java
public class Student {

    // Instanzvariablen

    private String name;
    private int semester;

    //Konstruktor
    public Student(String name, int semester){
        this.name=name;
        this.semester=semester;
    }

    //Methode mit Rueckgabewert int
    public int welchesSemester(){
        return semester;
    }

    //Methode ohne Rueckgabewert
    public void lernen(){
        System.out.println(this.name+" lernt gerade.");
    }
}
```

```java
public class Hauptprogramm {

    // Hier wieder die Main-Mathde zur Ausfuehrung des Programms
    public static void main(String[] args) {


        // Objekte erzeugen:
        Student peter = new Student("Peter",5);
        Student franzi = new Student("Franzi", 3);

        System.out.println("Student "+peter.name+" ist im "+ peter.welchesSemester()+". Semester");

        System.out.println(franzi.welchesSemester());

        //Aufruf der Methode lernen() auf den beiden Objekten
        peter.lernen();
        franzi.lernen();
    }
}
```

Auf `peter.name` kann jetzt nicht mehr direkt zugegriffen werden!

Auf Instanzvariablen zugreifen

Deshalb 2 Möglichkeiten zum setzen (schreiben):

- Bei Objekterzeugung über den Konstruktor (wie eben gesehen)
- Oder im weiteren Programmverlauf über spezielle „**setter**“-Methoden

Möglichkeit zum Lesen der Instanzvariablen:

- Über spezielle „**getter**“-Methoden

Innerhalb einer Klasse kann immer auf die privaten Instanzvariablen direkt zugegriffen (lesen u. schreiben) werden mit `objekt.instanzvariable`, bzw. `this.instanzvariable`

Setter-/Getter- Methoden allgemein

Allgemeiner Aufbau von Setter:

```
public void setIv_Name(IV_Typ iv_Name){
   this.iv_name = iv_name;
}
```

Allgemeiner Aufbau von Getter:

```
public IV_Typ getIv_Name (){
   return iv_name;
}
```

Diese eindeutigen (einfachen) Methoden können von Eclipse automatisch auf Basis der Instanzvariablen generiert werden

- *Source -> generate Getters and Setters…*

Programmieraufgabe

Lösen Sie bitte selbstständig die folgende Programmieraufgabe zu *Klassen & Objekten* in Java:

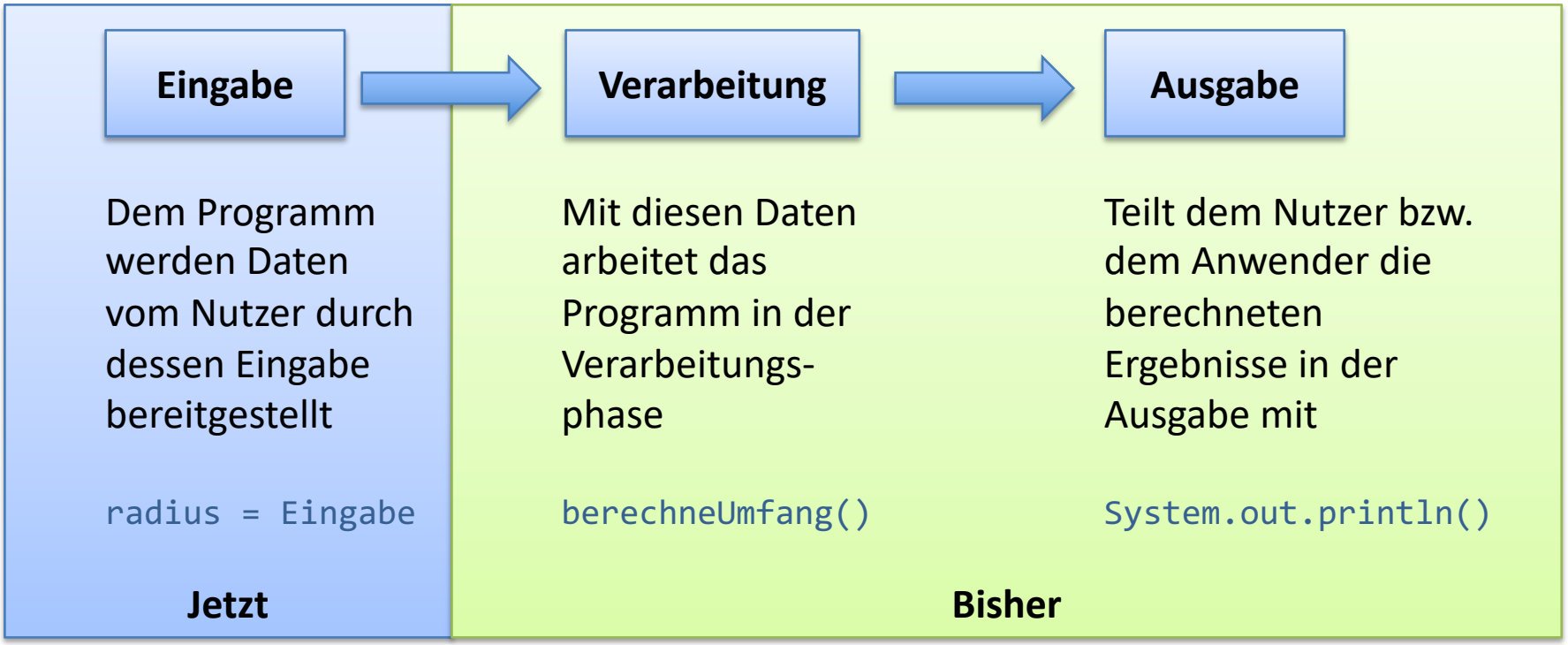
- Erstellen Sie ein neues Projekt „Uebung03“ mit einer Klasse `Kreis`
- Die Klasse `Kreis` besitzt:
  - Ein Attribut `radius`, welches den Radius des Kreises als Kommazahl `Double` speichert
  - Einen Konstruktor mit leerer Parameterliste, der den Radius mit 0 initialisiert
  - Einen zweiten Konstruktor, dem als Parameter eine Kommazahl zur Initialisierung des Radius übergeben wird.
  - Getter und Setter für den Radius
  - Eine Methode `double getUmfang()`, welche den Umfang des Kreises berechnet und zurückliefert
    - Hinweis: Kreisumfang = 2*pi*radius
    - Hinweis: pi kann als `double` mit 3.14 oder mittels `Math.PI` angegeben werden
  - Eine Methode `double getFlaeche()`, welche die Fläche des Kreises berechnet und zurückliefert
    - Hinweise: Kreisfläche = r^2*pi

- Erstellen Sie ein Testprogramm `KreisTest`, das:
  - Einen Kreis mit `radius` = 5 erzeugt
  - Und anschließend den Radius, den Umfang und die Fläche auf der Konsole ausgibt.

Benutzereingabe während des Programmablaufs

- Wir wollen das Kreis-Attribut `radius` nicht fest im Programmcode verbauen (*hardcoded*), sondern es erst zur Laufzeit durch eine Benutzereingabe einlesen!

Der Dialog mit dem Anwender (EVA Prinzip)



Nutzereingaben

Nutzereingabe über die Tastatur:

- Über die Konsole (Standardeingabe)
- Über eine grafische Benutzerschnittstelle **GUI** (*graphical user interface*)

1. Benutzereingabe über Konsole:

- `System.in` liefert uns den `InputStream` der Standardeingabe
  - analog zu `System.out` für `OutputStream` der Standardausgabe
- Wir bedienen uns der Java Klasse `Scanner` aus dem Package `java.util`, der wir im Konstruktor den `InputStream` der Standardeingabe übergeben:
  - `Scanner scan = new Scanner(System.in);`
- Mit den Methoden des Scanners können Eingabewerte gelesen werden
  - `scan.next()`: gibt nächstes Token als String zurück
    - Eine Zahl als Eingabe ist zunächst auch String
    - Kann mittels `Integer.parseInt(eingabe)` zu Integer umgewandelt werden
    - Analog für Kommazahl-Eingabe: `Double.parseDouble(eingabe)`
  - `Scan.nextInt()`: gibt nächstes Token als int zurück

Benutzereingabe über Konsole

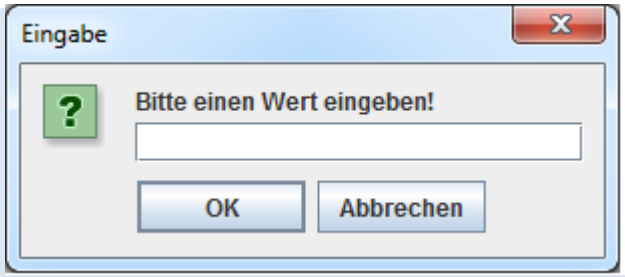

```
// Java Beispiel für Nutzereingabe über die Konsole

import java.util.Scanner; // Klasse Scanner muss importiert werden

// Objekt der Klasse Scanner mit Standardeingabe erzeugen
Scanner scan = new Scanner(System.in);

// Nutzer zur Eingabe auffordern:
System.out.println(„Bitte geben Sie einen Wert ein: “);

// Nutzereingabe lesen und als String speichern
String eingabe = scan.next();

// Nutzereingabe ausgeben
System.out.println(„Sie haben “+eingabe+“ eingegeben“);

// Oder Nutzereingabe in Integer umwandeln, falls Eingabe eine korrekte Zahl
int zahl_eingabe = Interger.parseInt(eingabe)
```

Programmieraufgabe:
Schreiben Sie Ihr Programm zur Kreisberechnung von vorhin so um, dass der Benutzer aufgefordert wird den Radius in der Konsole einzugeben. Das Programm soll dann den eingegebenen Wert verwenden, um den Radius, den Umfang und die Fläche auf der Konsole auszugeben!

Benutzereingabe über GUI

Die Klasse `JOptionPane` aus dem Packet `javax.swing` besitzt die Methode `showMessageDialog`:

- Zeigt ein grafisches Fenster mit einer Eingabeaufforderung
- Liefert nach klicken auf „ok“ die Eingabe als String zurück




```
// Beispiel für Nutzereingabe über GUI

// Klasse JOptionPane muss importiert werden!
import javax.swing.JOptionPane;

public class MeineKlasse{
   public static void main(String[] args){

      // Fenster zur Werteingabe erzeugen und Eingabe als String speichern
      String eingabe = JOptionPane.showInputDialog(„Bitte einen Wert eingeben!“);

      // Dann kann man wieder mit der Eingabe als String arbeiten. Bsp.:
      System.out.println(„Ihre Eingabe war “+eingabe);
   }
}
```

Programmieraufgabe:
Verwenden Sie nun das graphische Interface für die Abfrage des Radius und lassen Sie sich wieder den Radius, den Umfang und die Fläche des Kreises ausgeben.