LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

mobile and distributed systems group

Javakurs für Anfänger

Einheit 03: Wiederholung und Nutzereingaben

Lorenz Schauer
Lehrstuhl für Mobile und Verteilte Systeme

Heutige Agenda

1. Teil: Wiederholung

- Klassen, Objekte, Attribute und Methoden
- Das Schlüsselwort `this`
- Programmieraufgabe: Kreisberechnung

2. Teil: Erweiterter Programmablauf

- Nutzereingaben
  - Das EVA-Prinzip
- Eingaben über Konsole und GUI

Praxis:

- Die Klasse Kreis schreiben
- Die Klasse Kreis um Eingabeaufforderung erweitern

Lernziele

- Mehr Übung mit Objekten und Methoden
- Nutzereingaben implementieren
- Mit vorhandenen Klassen arbeiten

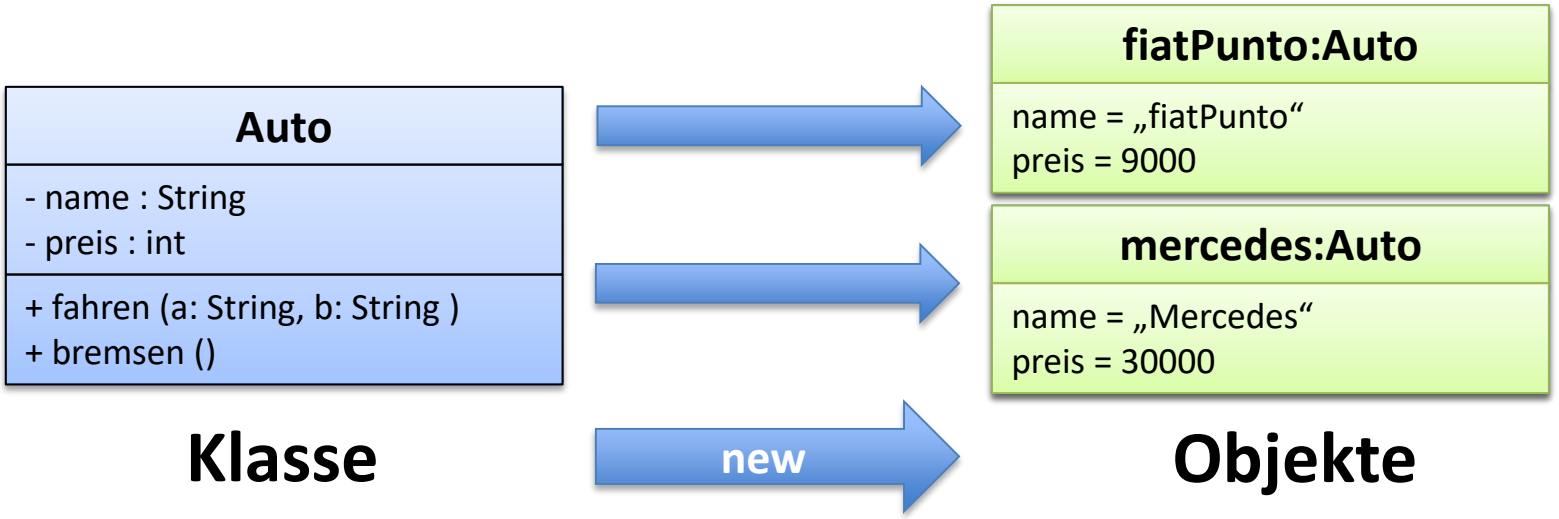
Wiederholung: Klassen & Objekte

Klasse:

- Stellt ein **Konzept** bzw. **Bauplan** dar
- Beschreibt dadurch einen Teil der Realität (**Attribute** und **Methoden**)

Objekt (= Instanz einer Klasse):

- Wird beim Ausführen des Programms (gemäß nach dem Bauplan der Klasse) erzeugt und spätestens beim Beenden wieder verworfen
- Bekommt **Werte** für seine Attribute



Wiederholung: Attribute & Methoden

Attribute werden durch **Instanzvariablen** definiert

- Direkten Zugriff von außerhalb der Klasse vermeiden!
  - Prinzip der Datenkapselung
  - Daher: `private` Deklaration
  - Beispiel:
    - `private String name;`
    - `private int preis;`

Das Verhalten wird durch **Methoden** definiert:

```
<Modifier> Rückgabetyp Methodenname(Parametertyp Parameter1,…){
        //Methodenrumpf
}
```

Verwendung und Manipulation eines Objekts über dessen Methodenaufrufe

Beispiel Autohaus (letzte Stunde)

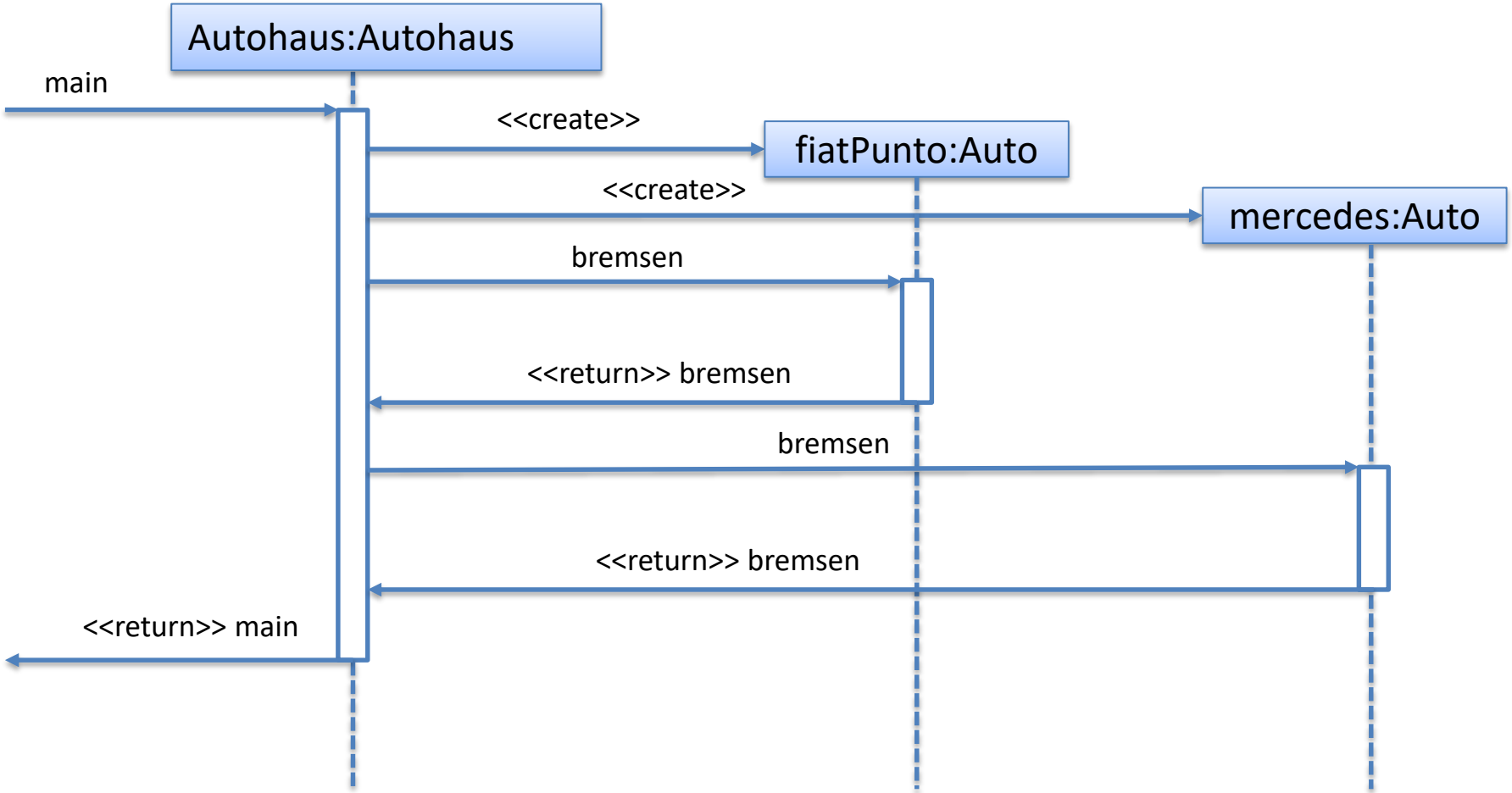

```
public class Autohaus {
    public static void main(String[] args) {

        Auto fiatPunto = new Auto("fiatPunto", 9000);

        Auto mercedes = new Auto("Mercedes",30000);


        fiatPunto.bremsen();

        mercedes.bremsen();
    }
}
```

Objekte der Klasse Auto erzeugen
Dabei werden die Instanzvariablen mit den entsprechenden Werten belegt

Auf den Objekten können die entsprechenden Methodenaufrufe erfolgen

Verwendung von Objekten

Programmablauf als Sequenzdiagramm am Beispiel Autohaus



Wiederholung: Das Schlüsselwort `this`

Das Schlüsselwort `this` wird als Selbstreferenz bezeichnet

- Verweist immer auf das eigene Objekt
- Um auf Instanzvariablen zuzugreifen: `this.instanzvariable`
  - Beugt versehentliche Verwechslungen mit lokalen Variablen oder Parametern vor
- Beispiel:

```
public class Person{
//Instanzvariablen
   private String meinName;
   private int alter;


   public Person(String name){
      meinName = name;
   }
}
```

Instanzvaribale **meinName** wird durch Parameter **name** nicht überschattet!
Daher: kein `this` nötig aber möglich!

Richtig wäre: `this.meinName = meinName;`

```
public class Person{
//Instanzvariablen
   private String meinName;
   private int alter;


   public Person(String meinName){
      meinName = meinName;
   }
}
```

Instanzvaribale **meinName** wird durch Parameter **meinName** überschattet!
Daher: *this* unbedingt nötig, um Instanzvariable zu belegen!

Programmieraufgabe zur Wiederholung

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

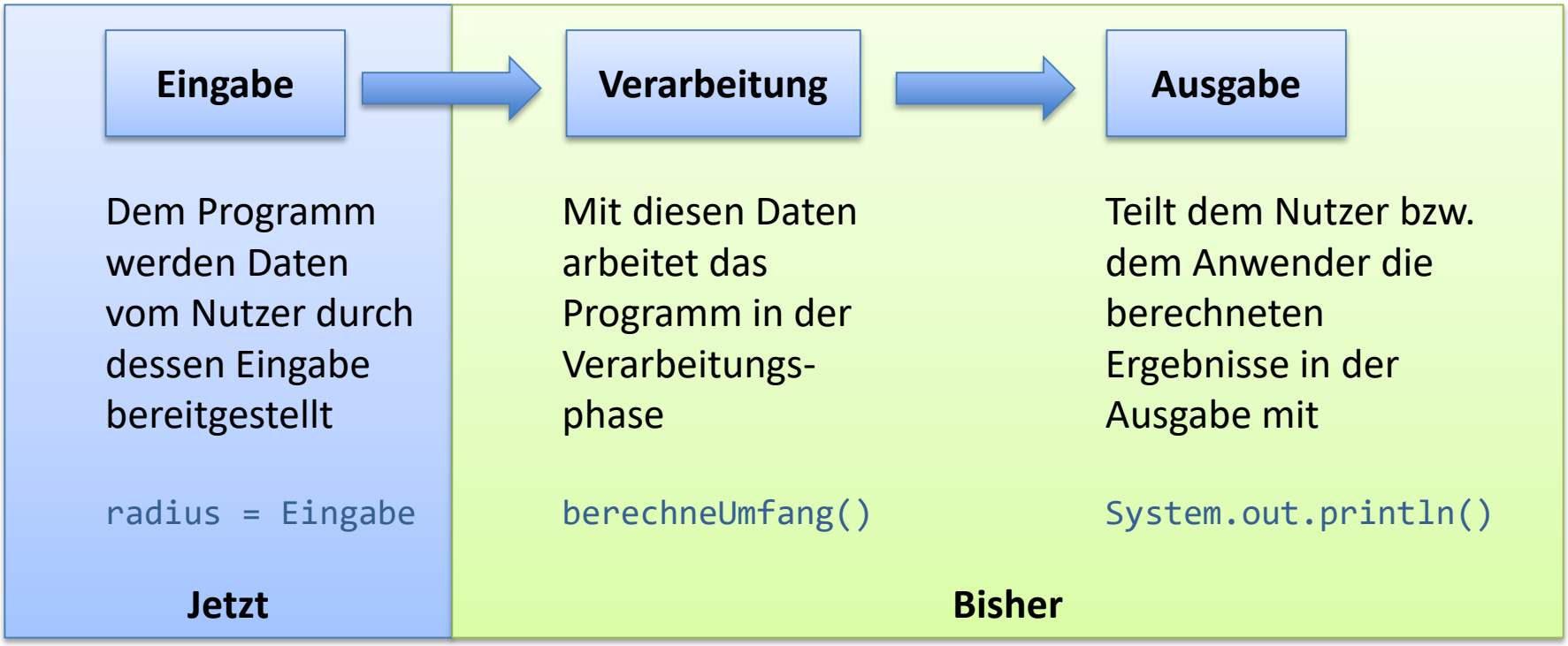
- Erstellen Sie ein neues Projekt „Uebung03“ mit einer Klasse `Kreis`
- Die Klasse `Kreis` besitzt:
  - Ein Attribut `radius`, welches den Radius des Kreises als Kommazahl speichert
  - Einen Konstruktor mit leerer Parameterliste, der den Radius mit 0 initialisiert
  - Einen zweiten Konstruktor, dem 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

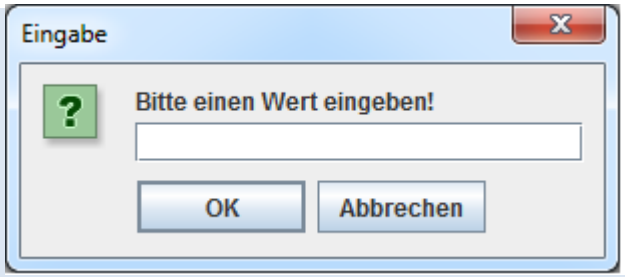

```
// Java Beispiel für Nutzereingabe über die Konsole

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.