



Statische Methoden, Vererbung, Benutzereingabe

Annabelle Klarl

Zentralübung zur Vorlesung

„Einführung in die Informatik: Programmierung und Softwareentwicklung“

<http://www.pst.ifi.lmu.de/Lehre/wise-14-15/infoeinf>



Action required now



1. Smartphone: installiere die App "socrative student" **oder**
Laptop: öffne im Browser b.socrative.com/login/student
2. Betrete den Raum **InfoEinf.**
3. Mache die **Zwischenevaluation** sofort!



Organisatorisches: Hausaufgabenabgabe

- Textaufgaben als PDF-Datei
 - in Word kann man Dateien auch als PDF abspeichern!
- Programmieraufgaben als .java-Datei
- Zippen Sie Ihre Abgabe nur 1x!
(keine geschachtelten .zip-Archive)

- Programmieraufgaben sollten in Eclipse gelöst werden.
 - Achten Sie auf korrekte Formatierung!
 - Geben Sie möglichst syntaktisch korrekte Programme ab!



Instanz- vs. Klassenattribute/methoden (I)

Wiederholung: Klassen und Objekte (vgl. ZÜ6)

- Attribute (=Instanzattribute) legen die charakteristischen Eigenschaften eines Objekts fest.
 - Z.B. **Klasse Mensch** mit Attributen `name`, `alter`, `groesse` (**Schablone**)
 - Z.B. **Objekt Annabelle** vom Klassentyp Mensch mit Attributwerten `name="Annabelle"`, `alter=28`, `groesse=165` (**konkrete Ausprägung**)
- Methoden (=Instanzmethoden) legen das charakteristische Verhalten eines Objekts fest.
 - Z.B. **Klasse Mensch** mit den Methoden `getGroesse()`, `wachsen(int cm)`
 - Z.B. **Verwendung eines Objekts** mit `annabelle.wachsen(10)`



Instanz- vs. Klassenattribute/methoden (II)

Neu: Statische Attribute und Methoden

- Statische Attribute (=Klassenattribute) legen die charakteristischen Eigenschaften einer Klasse unabhängig vom Zustand konkreter Objekte fest.
 - Z.B. Klasse **Mensch** mit den **statischen Attributen**
`populationsgroesse, biologFamilie`
- Statische Methoden (=Klassenmethoden) legen das charakteristische Verhalten einer Klasse unabhängig vom Zustand konkreter Objekte fest.
 - Z.B. Klasse **Mensch** mit den **statischen Methoden**
`geburtRegistrieren(), getBiologFamilie()`



Beispiel 1: Klasse Mensch (I)

Wir möchten uns in einem Attribut `ernaehrung` merken, was die bevorzugte Ernährungsform eines Menschen ist.



Welche Art von Attribut muss für die Ernährungsform angelegt werden?

- a) ein Instanzattribut speichert die Ernährungsgewohnheit eines bestimmten Menschen (z.B. Alice ist Vegetarier, Bob ist Veganer, Charles isst alles)
- b) ein Klassenattribut speichert die biologische Ernährungsform der Familie der Menschen (= Allesfresser)
- c) ein Instanz- oder/und ein Klassenattribut



Allgemeiner Aufbau einer Klasse in Java

```

public class C {
    private type1 attr1;
    ...
    private static type2 attr2 = ...;
    ...

    public C(params) {body}
    ...

    public type1/void methodName1(params1) {body1}
    ...

    public static type2/void methodName2(params2) {body2}
    ...
}

```

} Instanzattribute
 } Klassenattribute
 } Konstruktoren
 } Instanz-
 } methoden
 } Klassen
 } -methoden



Beispiel 1: Klasse Mensch (II)

```
public class Mensch {  
    private String name;  
    private int alter;  
    private int groesse;  
  
    private static int populationsgroesse = 0;  
    public static final String biologFamilie = "Menschenaffen";  
  
    public static void geburtRegistrieren() {  
        populationsgroesse++;  
    }  
  
    public static String getBiologFamilie() {  
        return biologFamilie;  
    }  
    ... // siehe ZÜ6  
}
```

Klassenattribute

Instanzattribute

Klassenmethoden



Beispiel 1: Klasse Mensch (III)

```
public class MenschTest {  
    public static void main(String[] args) {  
        Mensch baby =  
            new Mensch("Helena", 0, 53);  
        // Registrierung der Geburt  
        baby.wachsen(15);  
    }  
}
```



Was ist der korrekte
Aufruf, um die
Geburt von Helena
bekanntzugeben?

- a) `this.geburtRegistrieren();`
- b) `baby.geburtRegistrieren();`
- c) `Mensch.geburtRegistrieren();`
- d) `geburtRegistrieren(baby);`



Klassen und Vererbung

- Ein Klasse definiert die charakteristischen Merkmale von Objekten einer bestimmten Art.
 - Z.B. **Klasse Mensch** mit Attributen `name`, `geschlecht`, `alter`, `groesse`
- Vererbung dient dazu, neue Unterklassen zu definieren, die **alle** charakteristischen Merkmale einer Oberklasse haben und ...
 - ... diese Oberklasse **spezialisieren**
Z.B. **Klasse Frau** mit den geerbten Attributen `name`, `geschlecht`, `alter`, `groesse`, wobei `geschlecht="weiblich"` festgelegt wird
 - ... diese Oberklasse **erweitern**
Z.B. **Klasse Angestellter** mit den geerbten Attributen `name`, `geschlecht`, `alter`, `groesse`, und den zusätzlichen Attributen `position`, `gehalt`



Wissenswertes zur Vererbung (I)

Sichtbarkeiten beachten!

- Eine Subklasse erbt automatisch alle Attribute der Oberklasse.
 - **Geerbte** Attribute dürfen in der Subklasse nicht nochmal deklariert werden.
 - **Zusätzliche** Attribute werden in der Subklasse deklariert.
 - Der Konstruktor der Subklasse muss **alle** Attribute initialisieren (eventuell unter Verwendung des `super`-Konstruktors).
- Eine Subklasse erbt automatisch alle Methoden der Oberklasse.
 - **Geerbte** Methoden sind automatisch in der Subklasse verfügbar.
 - **Geerbte** Methoden **können** nochmal definiert werden und überschreiben damit die Funktionsweise der Methode in der Oberklasse.
 - **Zusätzliche** Methoden werden in der Subklasse definiert.



Beispiel 2: Schiff

- Jedes **Schiff** hat eine Tragfähigkeit **Tonnage** (in metrischen Tonnen) und eine **aktuelle Beladung**.
- Ein Schiff kann beladen werden und es kann ausgegeben werden, mit **welchem Antrieb** das Schiff fährt.

```
public class Schiff {  
    private int tonnage;  
    private int beladung;  
  
    public Schiff(int tonnage) {  
        this.tonnage = tonnage;  
        this.beladung = 0;  
    }  
  
    public void beladen(int ladung) {  
        this.beladung += ladung;  
    }  
  
    public String antriebAusgeben() {  
        return "Schiff ohne Antrieb";  
    }  
}
```



Beispiel 2: Subklasse Motorschiff

- Jedes **Motorschiff** hat zusätzlich eine **Motorleistung** (Kilowatt).

```
public class Motorschiff extends Schiff {  
    private int motorleistung;  
  
    public Motorschiff(int tonnage, int motorleistung) {  
        super(tonnage);  
        this.motorleistung = motorleistung;  
    }  
  
    public String antriebAusgeben() {  
        return "Motorschiff mit einer Motorleistung von "  
            + this.motorleistung;  
    }  
}
```

Achtung: Sichtbarkeit von tonnage ist private!

Die Methode antriebAusgeben wird überschrieben.

Die Methode beladen ist automatisch verfügbar.



Beispiel 2: Subklasse Segelschiff

- Jedes **Segelschiff** hat zusätzlich eine **Segelfläche** (m²).

```
public class Segelschiff extends Schiff {  
    private int segelflaeche;  
  
    public Segelschiff(int tonnage, int segelflaeche) {  
        super(tonnage);  
        this.segelflaeche = segelflaeche;  
    }  
  
    public String antriebAusgeben() {  
        return "Segelschiff mit einer Segelfläche von "  
            + this.segelflaeche;  
    }  
}
```

Die Methode `antriebAusgeben` wird überschrieben.

Die Methode `beladen` ist automatisch verfügbar.



Subtyping



Welche Initialisierung ist **nicht** erlaubt?

```
public class Main {  
    public static void main(String[] args) {  
        Schiff schiff1 = new Schiff(1000); // a)  
        Schiff schiff2 = new Segelschiff(1000,150); // b)  
  
        Segelschiff sschiff1 = new Schiff(1000); // c)  
        Segelschiff sschiff2 = new Segelschiff(1000,150); // d)  
        Segelschiff sschiff3 = new Motorschiff(1000,35000); // e)  
    }  
}
```



Dynamische Bindung



Was ist die Ausgabe des Programmstücks?

```
public class Main {  
    public static void main(String[] args) {  
        Schiff schiff1 = new Segelschiff(1000,150);  
        System.out.println(schiff1.antriebAusgeben());  
    }  
}
```

- a) "Schiff ohne Antrieb"
- b) "Motorschiff mit einer Motorleistung von 150"
- c) "Segelschiff mit einer Segelfläche von 150"



Grafische Benutzereingabe mit `JOptionPane`

Die bisherigen Anwendungen in den Übungen können verschiedene Dinge berechnen – aber sie sind noch nicht interaktiv. Wie kann eine **grafische Benutzereingabe** auf einfache Weise realisiert werden?

Bisher:

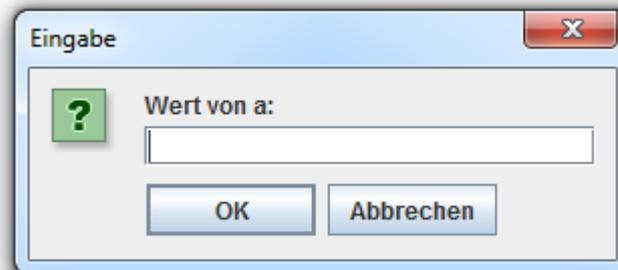
```
public static void main(String[] args) {  
    int a = 2;  
    int b = 5;  
  
    int produkt = a * b;  
    System.out.println("Produkt von a und b: " + produkt);  
}
```

Grafische Benutzereingabe mit `JOptionPane`

```
import javax.swing.JOptionPane;

public static void main(String[] args) {
    String a = JOptionPane.showInputDialog("Wert von a: ");
    String b = JOptionPane.showInputDialog("Wert von b: ");
}
```

Wir brauchen für die Berechnung eines Produkts Integerwerte



Solange das Dialogfenster angezeigt wird, stoppt das Programm und wartet, bis der Nutzer einen Wert eingegeben hat.



Grafische Benutzereingabe: Parsen eines Strings

```
import javax.swing.JOptionPane;

public static void main(String[] args) {

    String sa = JOptionPane.showInputDialog("Wert von a: ");
    int a = Integer.parseInt(sa);

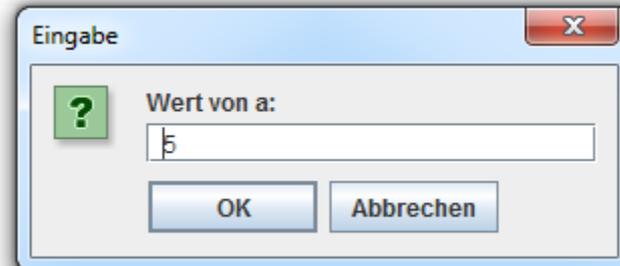
    String sb = JOptionPane.showInputDialog("Wert von b: ");
    int b = Integer.parseInt(sb);

    int produkt = a * b;
    System.out.println("Produkt von a und b: " + produkt);
}
```



Grafische Benutzereingabe: Parsen eines Strings

Was passiert, wenn der Nutzer in der Eile ein Leerzeichen vor der Ganzzahl eingibt? Oder eine Gleitkommazahl, oder einen Buchstaben?



```
Problems @ Javadoc Declaration Console
<terminated> Multiplier [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (29.11.2011 22:09:54)
Exception in thread "main" java.lang.NumberFormatException: For input string: " 5"
    at java.lang.NumberFormatException.forInputString(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at java.lang.Integer.parseInt(Unknown Source)
    at Multiplier.main(Multiplier.java:8)
```