PACKAGES, ZUGRIFFSRECHTE, THIS, GET/SET

Übersicht

- Package
- this
- Zugriffsrechte
- Getter/Setter

PACKAGE

Gliederung in Packages /1

Package in Java

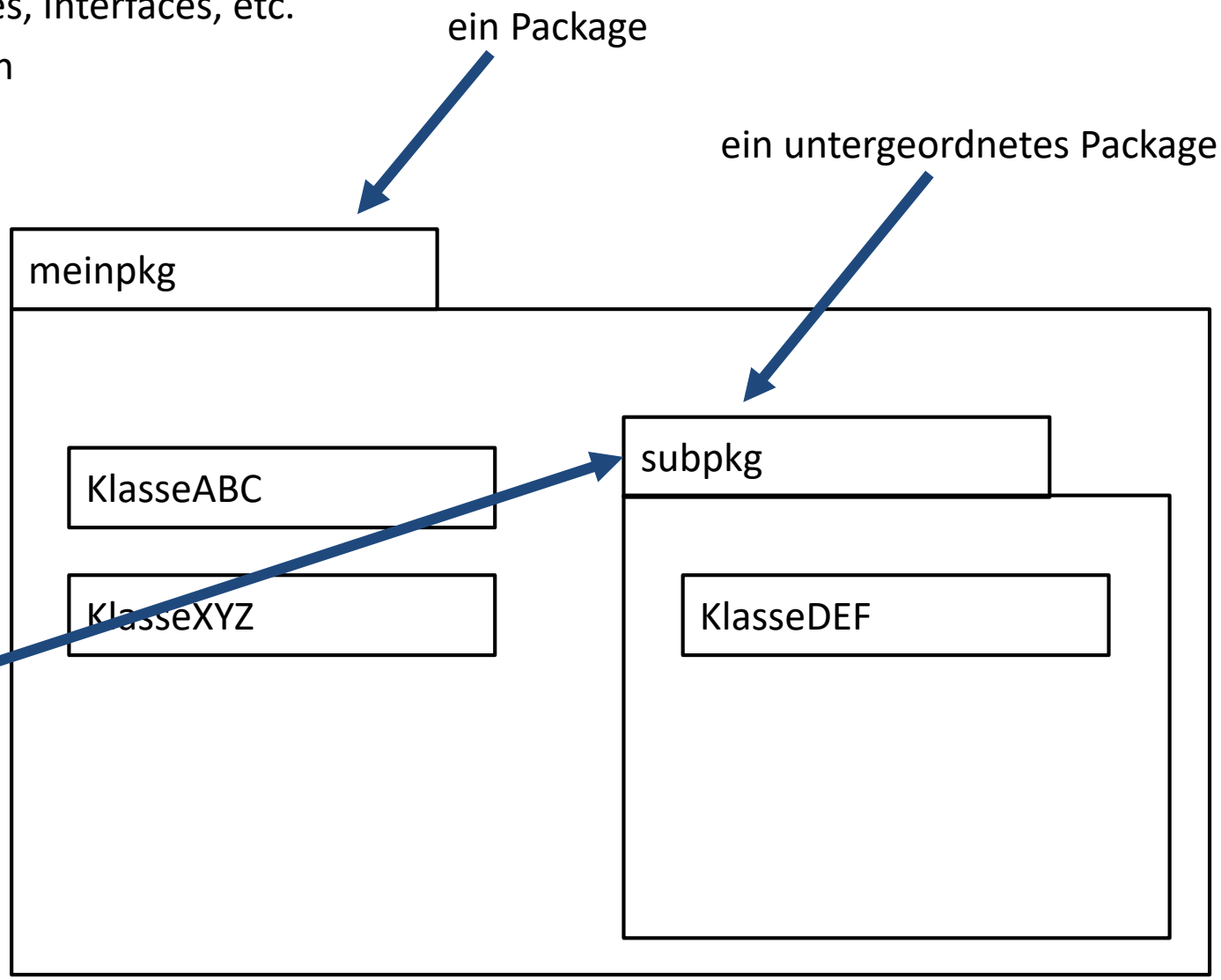
- eine Menge von Klassen, untergeordneten Packages, Interfaces, etc.
- organisatorische Hilfe -> Abgrenzung von Bereichen
- Schlüsselwort **`package`**, danach package-Name

Beispiel:

```
package meinpkg;

public class KlasseABC {

}
```

```
package meinpkg;

public class KlasseXYZ {

}
```

```
package meinpkg.subpkg;

public class KlasseDEF {

}
```



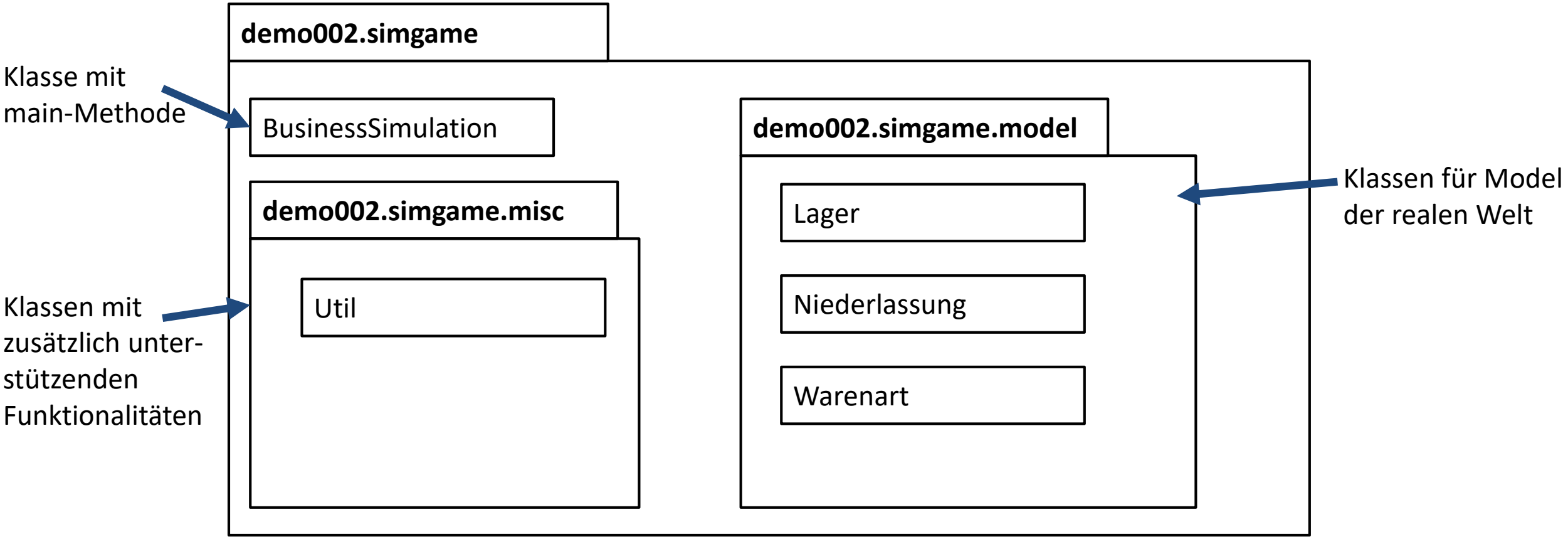
Gliederung in Packages /2

Packages in Wirtschaftssimulation

- eine Menge von Klassen, untergeordneten Packages und Interfaces
- organisatorische Hilfe
- Schlüsselwort **`package`**, danach package-Name

Hinweis

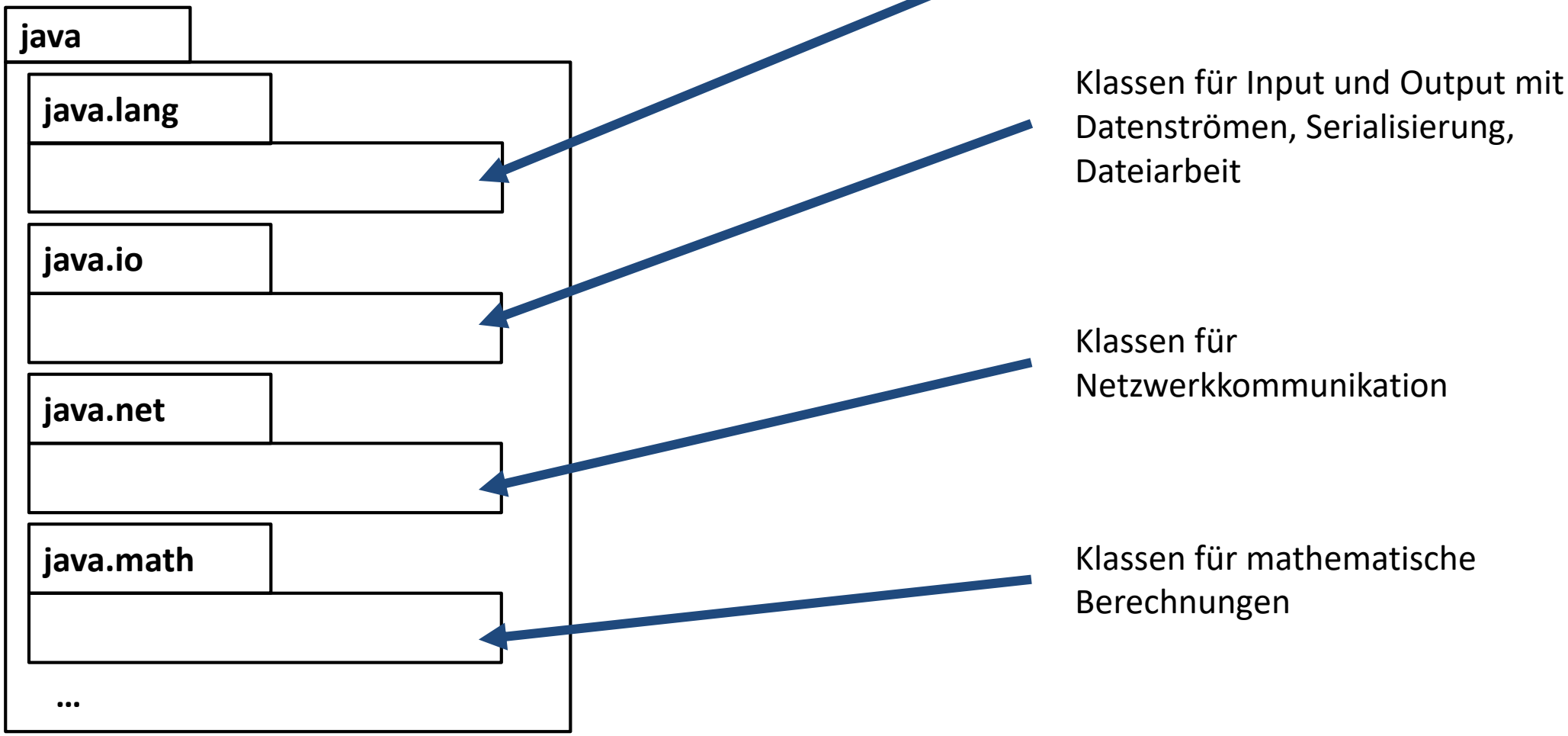
Diese Gliederung in die packages ist nur eine Möglichkeit. In anderen Anwendungen können andere Gliederungen nötig sein.



Gliederung in Packages /2

Packages in Java

- Gliederung nach Funktionalität und Einsatzgebiet
- Grundlage der Sprache Java
- mit JDK (Java Development Kit) mitgeliefert



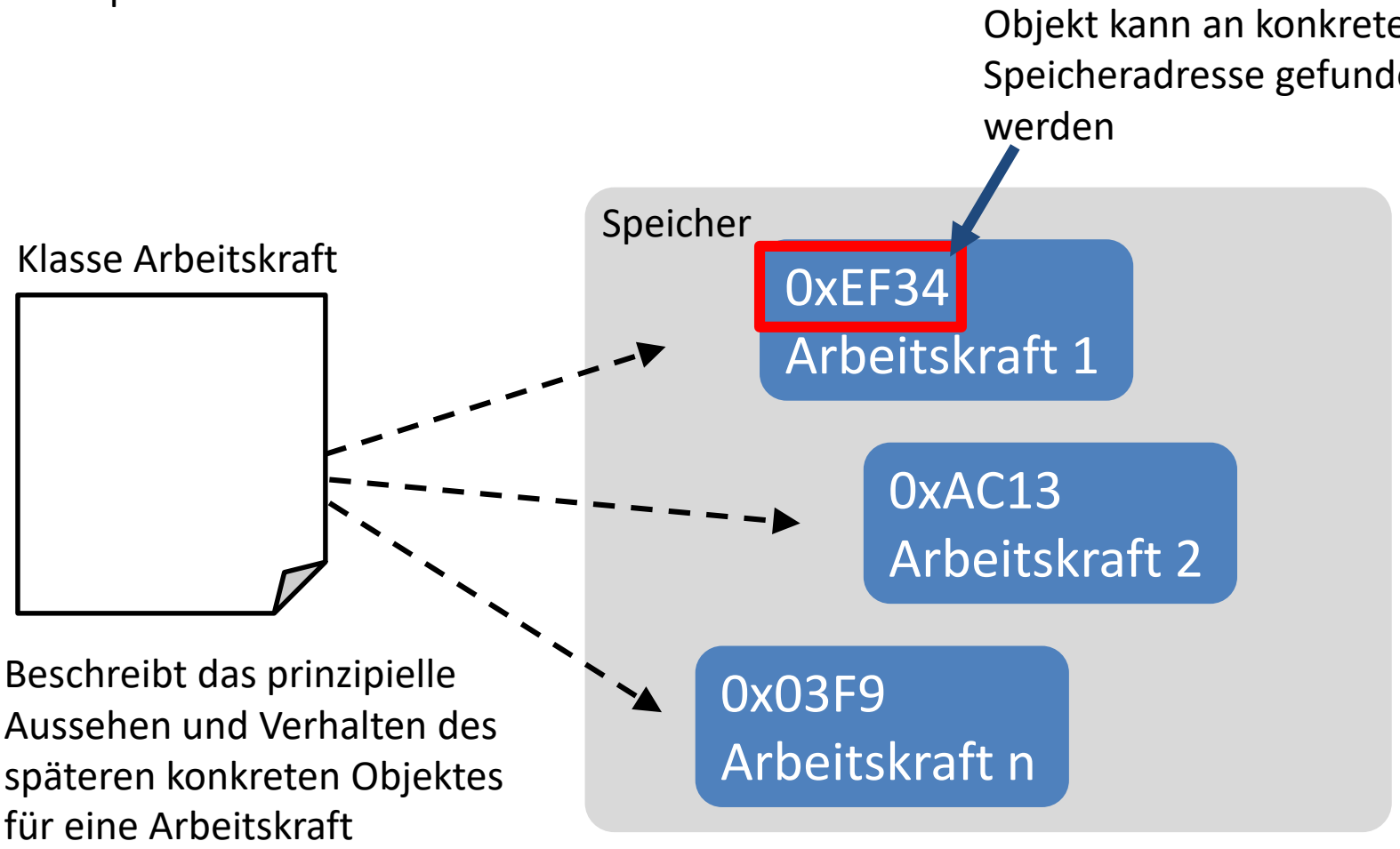
SCHLÜSSELWORT THIS

Schlüsselwort >>this<< /1

Wiederholung: Instanziieren einer Klasse

- mit „Bauplan“ ein konkretes Objekt im Speicher bauen
- für Attribute werden Speicherbereiche reserviert

Beispiel:



Schlüsselwort >>this<< /2

Referenz auf sich selbst

- bezeichnen des eigenen Attributes
- Unterscheidung von lokaler Variable (z. B. im Konstruktor)

Beispiel:

```
...
private String personalnr;
private String name;
private int gehalt;

public Arbeitskraft(String nr, String name, int gehalt) {
    this.personalnr=nr;

    this.name=name;

    this.gehalt=gehalt;
}
...
```

Referenz auf das Attribut der Klasse

lokale Variable Gehalt, nur im Konstruktor gültig

Schlüsselwort >>this<< /3

Referenz auf sich selbst

- Übergabe einer Referenz
- Nutzen der Referenz in anderem Objekt

Referenz auf sich selbst. Mitteilung der Adresse, an welcher das Objekt im Speicher „lebt“, an die Methode setAbteilung()

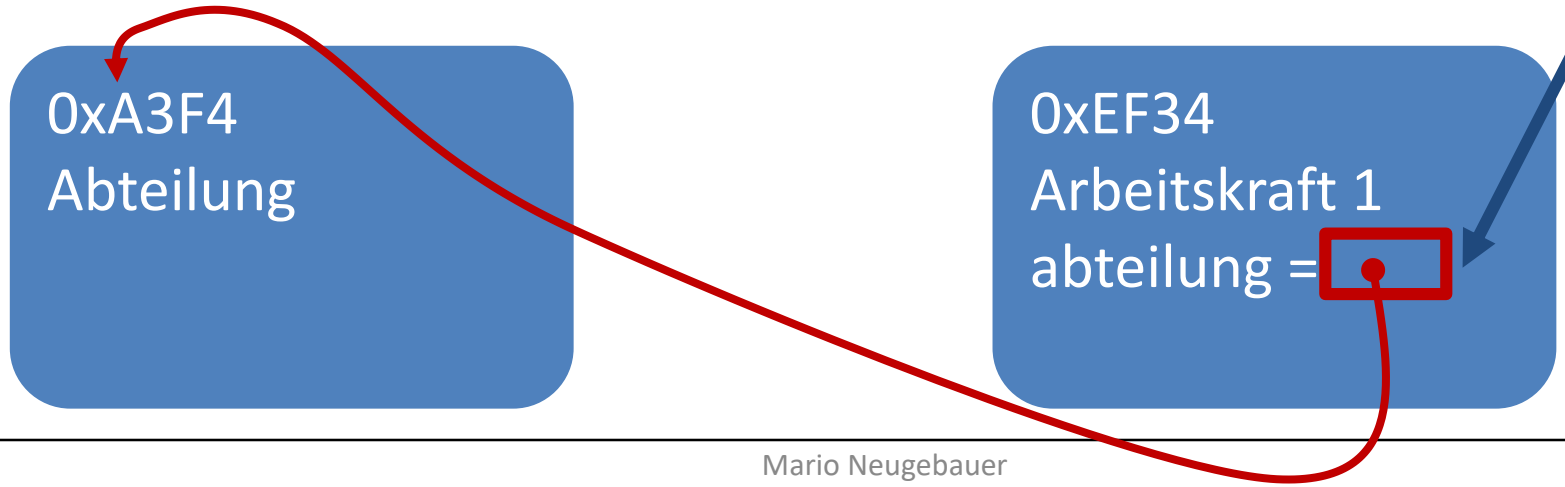
Übernahme einer Referenz auf ein Objekt der Klasse Abteilung, das in einer bestimmten Speicheradresse „lebt“

Beispiel:

```
public void hinzufuegen(Arbeitskraft arbeitskraft) {
     if(listeArbeitskraefte!=null) {
          listeArbeitskraefte.add(arbeitskraft);
          arbeitskraft.setAbteilung(this);
     }
}
```

```
public void setAbteilung(Abteilung abteilung) {
     this.abteilung = abteilung;
}
```

Ergebnis:



Referenz auf das konkrete Objekt Abteilung, welches der Arbeitskraft zugeordnet wurde

ZUGRIFFSRECHTE

Zugriffsrechte /1

Regeln für Zugriffsschutz

- Attribute und Methoden sollen nicht überall sichtbar sein
- Zugriffsrechte sind detailliert definierbar

Möglichkeiten

- private
- protected
- (ohne Schlüsselwort)
- public

Zugriff	Class	Package	Subclass	World
public	erlaubt	erlaubt	erlaubt	erlaubt
protected	erlaubt	erlaubt	erlaubt	verboten
(ohne)	erlaubt	erlaubt	verboten	verboten
private	erlaubt	verboten	verboten	verboten

Bereiche für Zugriff auf Attribute/Methoden

- innerhalb der gleichen Klasse
- innerhalb des gleichen Packages
- in einer abgeleiteten Klasse (Subclass, Unterklasse) – siehe Vererbung
- in allen anderen Klassen

Zugriffsrechte /2

Überlegungen zur Verwendung

- andere Programmierer könnten Deine Klasse nutzen
- Fehler durch falsche Verwendung sollen ausgeschlossen werden
- restriktivstes Recht nutzen, außer es gibt einen guten Grund für mehr Zugriff
- public für Attribute vermeiden, außer Konstanten

➔ weitere Überlegungen bzgl. Getter-/Setter-Methoden

GETTER-/SETTER-METHODEN

Getter-/Setter-Methoden

Getter-/Setter-Methoden

- Attribute immer mit Zugriffsrecht **private**
- Zusätzliche Methoden für Zugriff auf dieses Attribut
- in IDE generierbar

Mögliche Vorteile (Auszug, praktisch kontroverse Diskussion)

- Kapseln von Verhalten für Setzen der Werte; zusätzliche Funktionalität (z. B. Wertüberprüfung) möglich
- innere Repräsentation verbergen und nur alternative Repräsentation nach außen zeigen
- bei Veränderungen mit der Zeit: Implementierung innen kann sich ändern; sichtbares Verhalten außen bleibt gleich
- Zugriff für Debugging einfacher, Änderung des Attributes passiert immer nur über Setter, mögliche Fehlerursache leichter finden
- Subklassen können Verhalten bei get/set ändern
- unterschiedliche Zugriffsrechte für get und set möglich, z. B. public get, aber set ganz verbieten

```
public class GetterSetterDemoClass {

    private int einIntAttribut;

    public GetterSetterDemoClass(int value) {
        einIntAttribut = value;
    }

    public int getEinIntAttribut() {
        return einIntAttribut;
    }

    public void setEinIntAttribut(int einIntAttribut) {
        this.einIntAttribut = einIntAttribut;
    }
}
```

Regeln für Getter (anpassbar an spezielle Anforderungen)

- public; Rückgabetyp (meist des Attributes)
- **get + attributname** (in CamelCase)

Regeln für Setter (anpassbar an spezielle Anforderungen)

- public; kein Rückgabetyp
- **set + attributname** (in CamelCase)

Zusammenfassung

- Packages strukturieren Klassen
- this dient der Referenz auf sich selbst
- Zugriffsrechte schränken Zugriff ein
- Getter/Setter für Attribute