Übersicht

Pakete I

- Programmeinheiten
  - Dienen der übersichtlichen Strukturierung
  - Stellen logische Bestandteile eines Programms im Quellkode dar

- Programmeinheiten in Java sind:
  - Klassen
  - Schnittstellen (Interfaces)
  - Threads
  - ***Pakete***

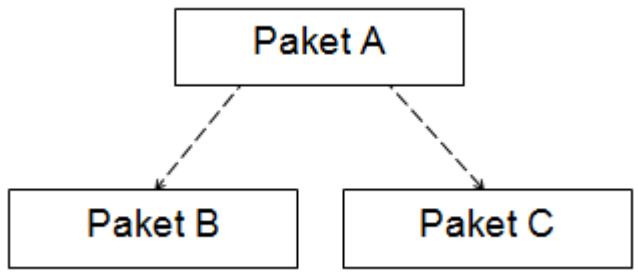
Quelle: Oliver Lazar

Pakete II

- Pakete stellen die gröbsten Strukturierungseinheiten der objektorientierten Technik dar.

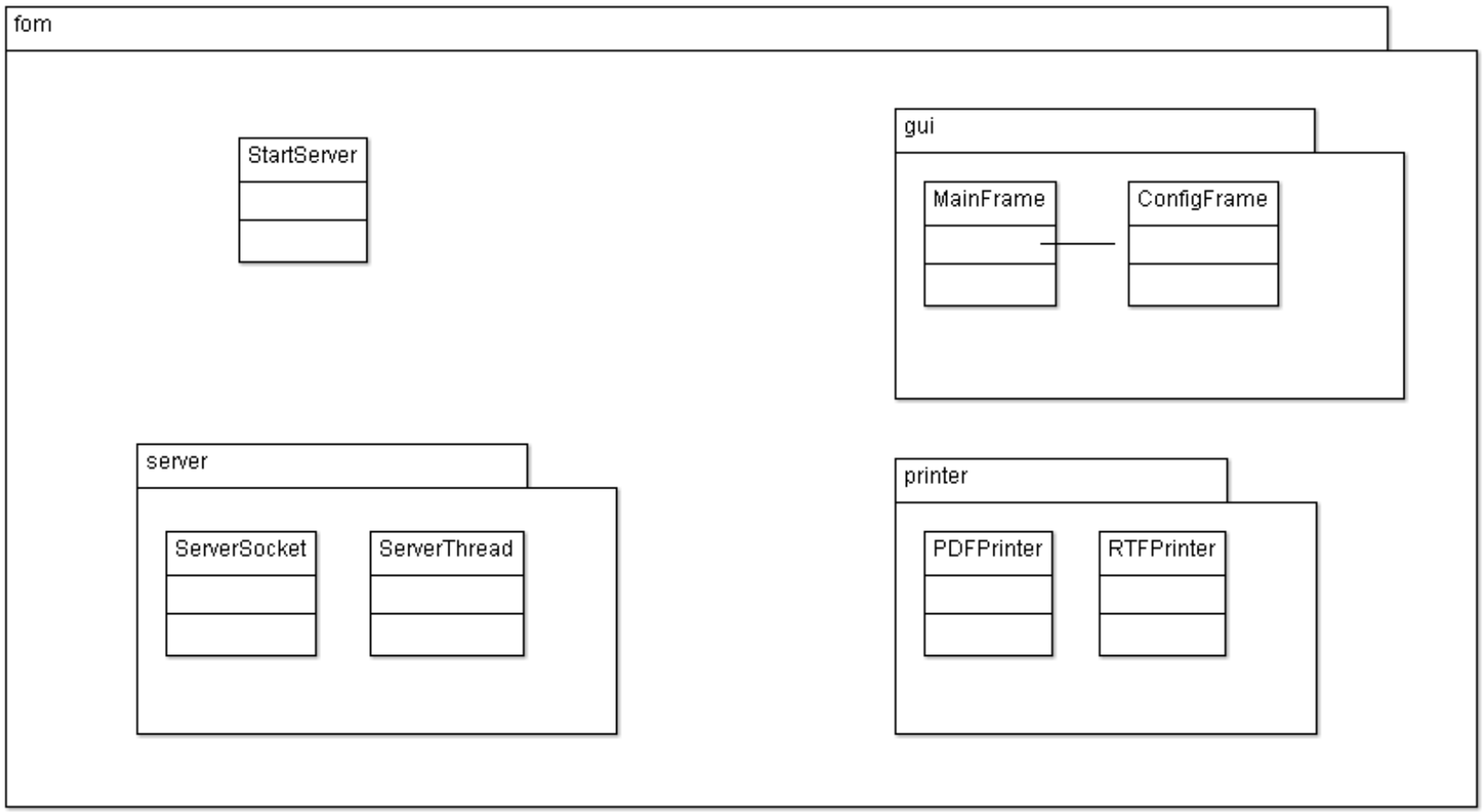
  Pakete werden im Rahmen des Entwurfs der Software konzipiert.

- Der Einsatz von Paketen bietet die folgenden Vorteile:
  - Pakete bilden einen eigenen Bereich (Zugriffsschutz/ Information Hiding)
  - Pakete bilden einen eigenen Namensraum (Vermeidung von Namenskonflikten)
  - Pakete sind eine größere Stukturierungseinheit als Klassen, so können zusammengehörige Klassen (z.B. GUI-Klassen oder DB-Klassen) in jeweils eigene Pakete gepackt werden



Pakete III

- Pakete könne bereits beim Klassendiagramm im UML – Entwurf verwendet werden:



Quelle: Oliver Lazar

Pakete erstellen I

- Ein Paket wird definiert, indem alle Dateien des Pakets mit der Deklaration des Paketnamens versehen werden.

```
// Datei: Artikel.java

package lagerverwaltung;  // Deklaration des Paketnamens

public class Artikel      // Definition der Komponente Artikel des
{                         // Pakets lagerverwaltung
   private String name;
   private float preis;

   public Artikel (String name, float preis)
   {
      this.name = name;
      this.preis = preis;
   }
   // Es folgen die Methoden der Klasse
}
```

Pakete erstellen II

- Paketnamen werden konventionsgemäß immer klein geschrieben.
- Verwendung des Schlüsselwortes *package*: muss immer die erste Zeile in der jeweiligen Quellcodedatei stehen(außer Kommentare)
- Pakete können andere Pakete enthalten u.s.w.
  - So könnte z.B. das Paket lagerverwaltung ein Unterpaket vom Paket betriebsverwaltung sein (Notation in Subpaketen erfolgt per Verkettung mit dem Punkt → siehe Quellcode)
  - So können beliebig viele Hierarchieebenen angelegt werden
  - Per Konvention orientieren sich die Paketnamen in Java an den Internet-Domänen ihrer Entwickler, z.B. de.fraunhofer.ims.projekt…

```
package betriebsverwaltung.lagerverwaltung;
…
```

Pakete benutzen I

- Sind die Klassen A und B einem Paket namens paket zugeordnet, so sind diese Klassen Komponenten des Pakets paket.
- Möchte man aus einer Klasse C, die nicht Bestandteil von paket ist, auf A zugreifen, erfolgt das mit der Punktnotation: paket.A
- Beispiel mit drei Paketen (figurpaket, kreispaket, eckpaket) und drei Klassen

```
package figurpaket;

public class Figur{
    kreispaket.Kreis kreisRef = new kreispaket.Kreis();

    eckpaket.Eck eckRef = new eckpaket.Eck();
    …
}
```

Pakete Import

- Es kann schnell lästig werden, ständig per Punktnotation auf andere Pakete und deren Klassen zugreifen zu müssen.
- Einfacher geht es mit der import-Vereinbarung, damit können mit public deklarierte Klassen aus anderen Paketen sichtbar gemacht werden.
- Die import-Vereinbarung muss hinter der package-Deklarartion aber vor dem Rest des Quellcodes stehen.
- Es können beliebig viele import-Vereinbarungen aufeinanderfolgen.

```
package figurpaket;

import kreispaket.*;
import static eckpaket.*;

public class Figur{
    Kreis kreisRef = new Kreis();

    Eck eckRef = new Eck();
    …
}
```

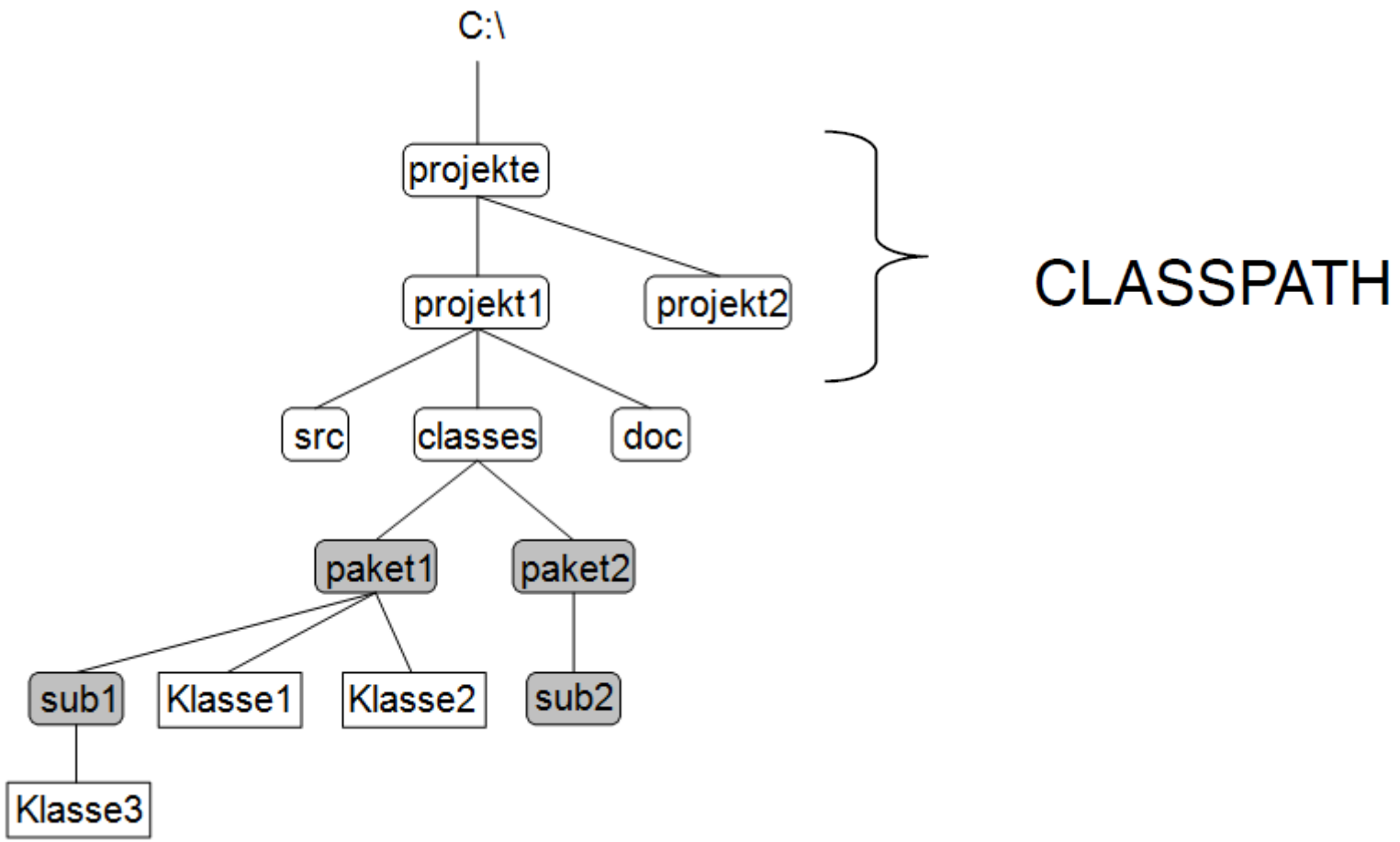
Übersicht

Inhalte

- ✓ Pakete
- ✓ Paketnamen
- Zugriffsmodifikatoren

Paketnamen I

- Der CLASSPATH ist eine Umgebungsvariable, die dem Compiler und dem Interpreter sagt, wo diese nach Quellcode und Bytecode-Dateien suchen sollen.



Übersicht

Inhalte

- ✓ Pakete
- ✓ Paketnamen
- ✓ Zugriffsmodifikatoren

Zugriffsmodifikatoren I

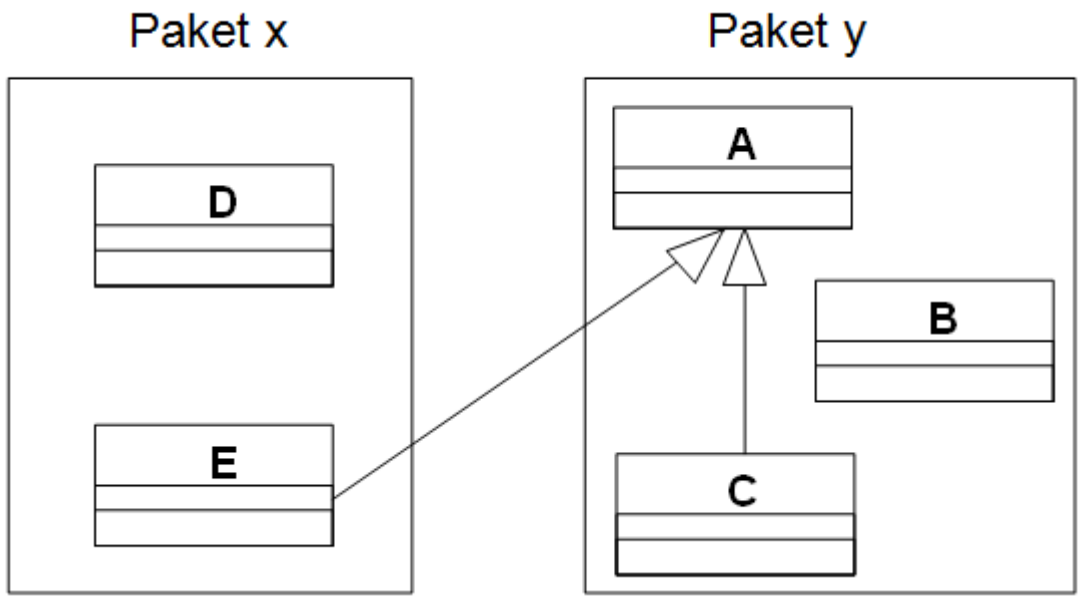
- Zur Regelung des Zugriffschutzes in Java gibt es die Zugriffsmodifikatoren (Schlüsselwörter) *public*, *protected* und *private*
- Zum Zugriff auf für Klassen und Schnittstellen in einem Paket gibt es nur
  - *default* (friendly): ist nur für Klassen/Schnittstellen desselben Paketes sichtbar ist selbst in Unterpaketen nicht sichtbar (Beachten Sie, dass default (bzw. friendly) kein Schlüsselwort von Java ist)
  - *public* ist auch für Klassen/Schnittstellen aus anderen Paketen sichtbar

```
package lagerverwaltung;

public class Artikel{
    …
}

// Zugriffsschutz default
class Lieferant{
    …
}
```

```
package abrechnung;

import lagerverwaltung.Artikel;

//Fehler, da nicht public
import lagerverwaltung.Lieferant;

public class Materialabrechnung{
    …
}
```

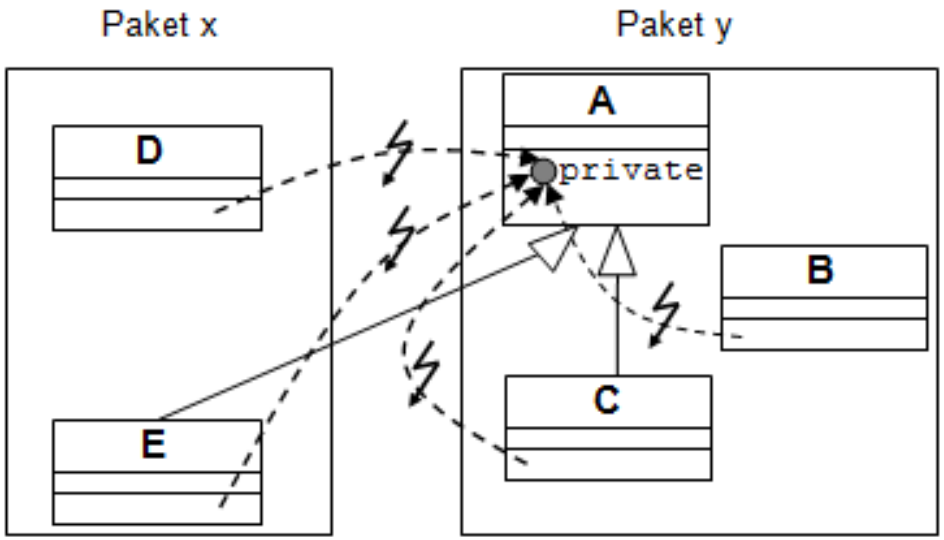
Zugriffsmodifikatoren II

- Zugriffschutz für Methoden und Datenfelder
  - *default*
  - *public*
  - *protected*
  - *private*



Zugriffsmodifikatoren III

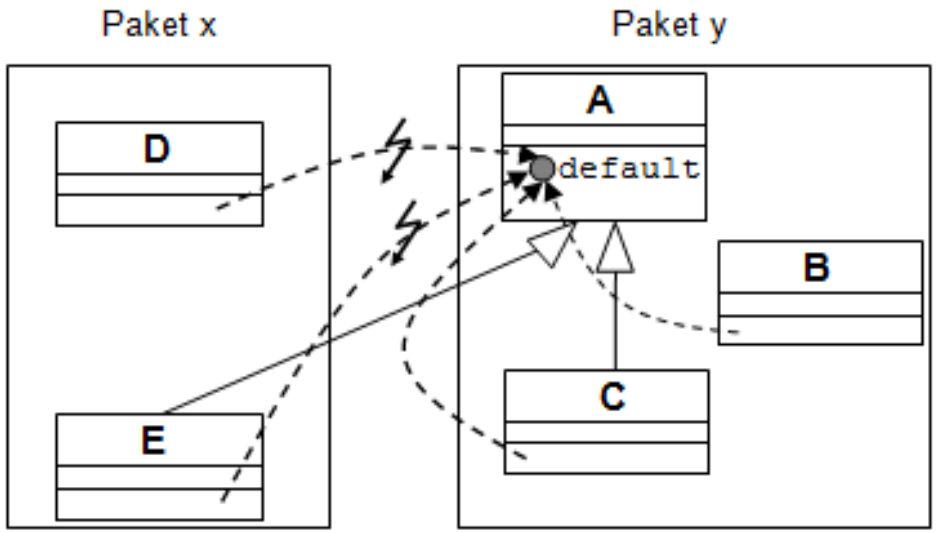
- Zugriffsmodifikator *private*
  - *Zugriff nur innerhalb der Klassendefinition*



Keine der anderen Klassen B, C, D oder E haben Zugriff auf `private` Methoden und Datenfelder von A

Zugriffsmodifikatoren IV

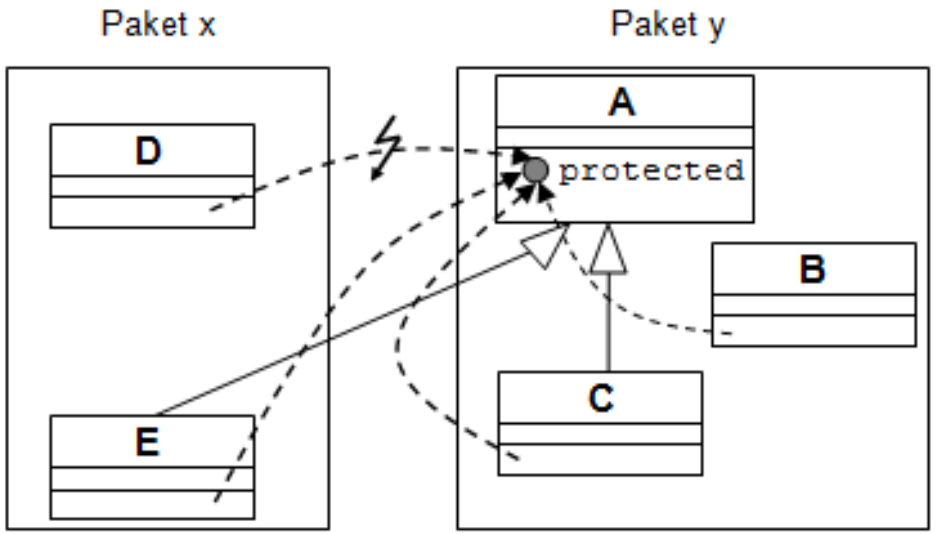
- Zugriffsmodifikator *default*
  - *Zugriff von allen Klassen aus dem gleichen Paket möglich*



Weder D noch E haben Zugriff auf `default` Methoden und Datenfelder von A

Zugriffsmodifikatoren V

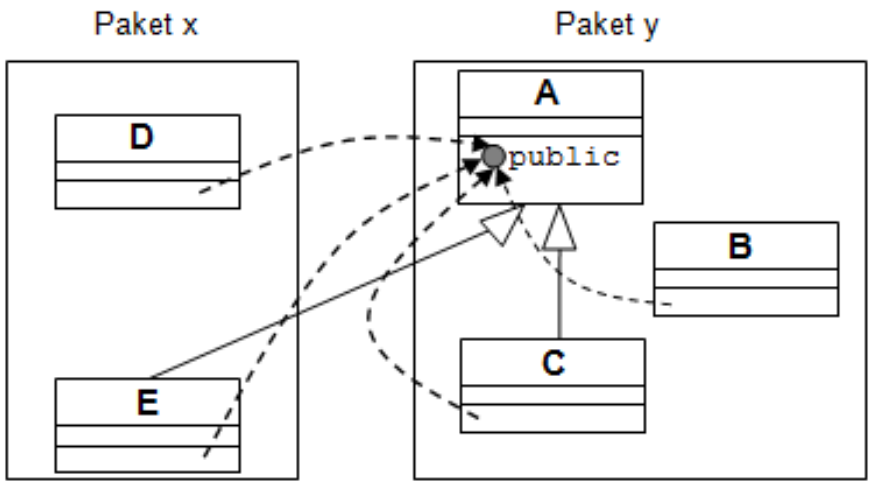
- Zugriffsmodifikator *protected*
  - *Erweiterung von default, zusätzlich können Subklassen aus anderen Paketen zugreifen*



Da E eine Subklasse von A ist, kann auf `protected` Methoden und Datenfelder von A aus E zugegriffen werden. D hat hingegen keinen Zugriff auf `protected` Methoden und Datenfelder von A.

Zugriffsmodifikatoren VI

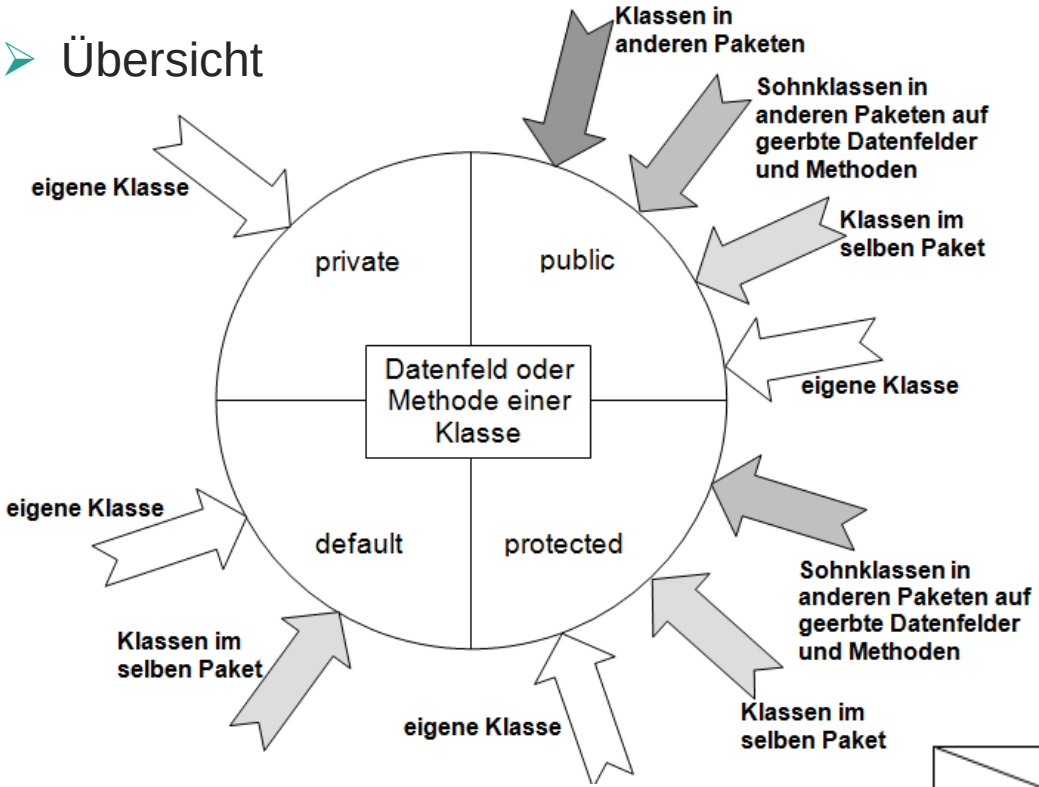
- Zugriffsmodifikator *public*
  - *Es existiert keine Zugriffsschutz, alle Klassen können zugreifen*

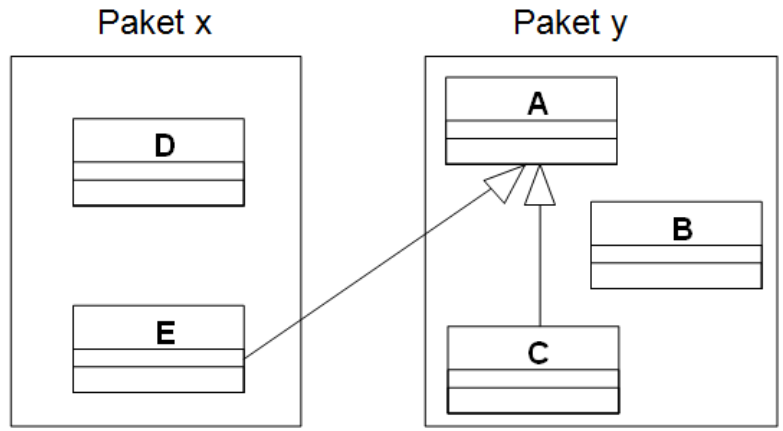


Alles und jeder hat Zugriff auf `public` Methoden und Datenfelder von A.

Zugriffsmodifikatoren VII

- Übersicht





Zugriff auf Datenfelder und Methoden der Klasse A

hat Zugriff auf	`private` Datenfelder und Methoden	default Datenfelder und Methoden	`protected` Datenfelder und Methoden	`public` Datenfelder und Methoden
Klasse `A` selbst	Ja	Ja	Ja	Ja
Klasse `B` gleiches Paket	Nein	Ja	Ja	Ja
Subklasse `C` gleiches Paket	Nein	Ja	Ja	Ja
Subklasse `E` anderes Paket	Nein	Nein	Ja/Nein	Ja
Klasse `D` anderes Paket	Nein	Nein	Nein	Ja

Zugriffsmodifikatoren VIII

- Zugriffsschutz *für Konstruktoren*
  - *Wird überhaupt kein Konstruktor zur Verfügung gestellt, so existiert der vom Compiler zur Verfügung gestellte voreingestellte Default-Konstruktor. Dieser Konstruktor hat den Zugriffsschutz der Klasse. Ist die Klasse public, so ist auch der voreingestellte Default-Konstruktor public. Ist die Klasse default, so ist auch der voreingestellte Default Konstruktor default.*

- Zugriffsmodifikatoren beim Überschreiben von Methoden

Zugriffsmodifikatoren in der Superklasse	Zugriffsmodifikatoren in der Subklasse
private	Kein Überschreiben möglich, aber neue Definition im Sohn.
default	default protected public
protected	protected public
public	Public

Aufgaben

- **Aufgabe 09.01** (12.3)
  - Bearbeiten Sie die Aufgabe 09.01

- **Aufgabe 09.02** (12.4)
  - Bearbeiten Sie die Aufgabe 09.02