Java Einführung

Packages

Inhalt dieser Einheit

- Packages (= Klassenbibliotheken)
- Packages erstellen
- Packages importieren
- Packages verwenden
- Standard Packages

Code-Reuse

- **Einbinden** von bereits (selbst-/fremd) programmiertem Code
- **Wiederverwendung** von Klassen, mit oder ohne deren Implementierung zu kennen

Techniken:

- Vorhandenen Klassen (aus Packages) **verwenden**
- Vererbung (vorhandene Klassen **erweitern** und **spezialisieren**)

Packages

- Verschiedene, funktional zusammengehörende Klassen werden in Packages mit einem klaren Interface (`public` Klassen und Methoden) integriert.

- API (Application Program Interface): Gesamtheit der von zusammengehörenden Paketen definierte Schnittstellen

- Packages werden mit der
  - Java-Plattform ausgeliefert (Standard-Packages),
  - können selbst oder
  - von Dritten programmiert werden.

Vorteile von Packages

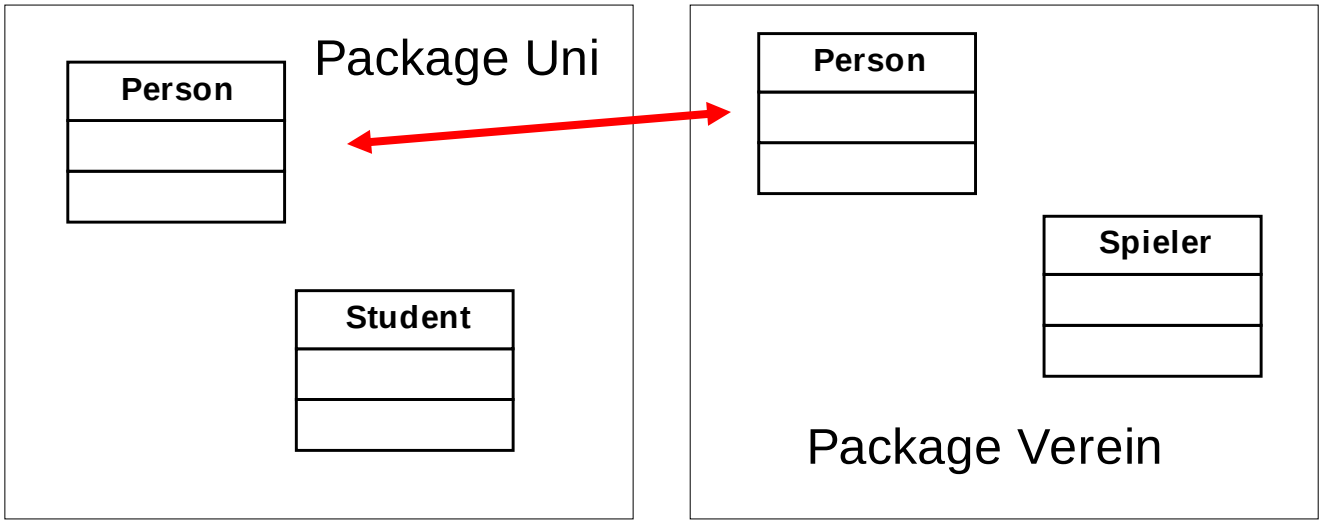
Die Verwendung von Packages bringt folgende Vorteile:

- Systematische Ordnung bei vielen Klassen ("Name spaces")
- Leichter überschaubaren Programmcode
- Definition eines klaren `public`-Interfaces der Packages (API) meist mit Automatisch generierter Dokumentation (JavaDoc)

Name spaces

- Ermöglichen die Verwendung der gleichen Namen in verschiedenen Packages.

- Bsp: `Uni.Person` und `Verein.Person` sind verschiedene Klassen



Packages erstellen

Alle Klassen die zu einem Package gehören, müssen:

- In ein Verzeichnis (Ordner) mit dem Namen des Pakets gespeichert werden.
- Mit ***package*** *packageName* beginnen.

Bsp: Datei `mypackage/MyClass.java`

```
package mypackage;

public class MyClass {
...
}
```

Packages verwenden

- Packages müssen entweder ausdrücklich (explizit) oder implizit importiert werden.

  Beispiele:

  - Import mit expliziter Importanweisung

```
import mypackage.*;
MyClass m = new MyClass();
```

  - Impliziter Import

```
mypackage.MyClass m =
        new mypackage.MyClass();
```

Packages verwenden II

- Einbinden des gesamten Pakets
  ```
  import java.util.*;
  ```
- Einbinden einzelner Klassen eines Pakets
  ```
  import java.util.Date;
  ```

Interface des eigenen Pakets

- Die Klassen und Methoden, die von außerhalb des Pakets verwendet werden sollen, müssen als **`public`** deklariert werden, da sie sonst nur innerhalb des Paketes sichtbar sind.

- Um Packages weltweit eindeutig zu benennen soll der Pfad als verkehrter Domain-Name des Klassenerzeugers gewählt werden
  z.B. `at.ac.wuwien.meinPaket`
  das dazugehörige Verzeichnis ist
  `at/ac/wuwien/meinPaket/`

Java-Standard-Packages

- `java.lang /*`wird implizit immer importiert (keine `import-`Anweisung nötig), enthält z.B. Klasse `String, Math,... */`
- `java.io //`Ein/Ausgabe
- `java.util //`nützliche Klassen (Datum, Random etc)
- `java.net //`für die Kommunikation über Netzwerke
- `java.awt /*`Klassen für die Benutzerschnittstelle und Graphikprogrammierung `*/`

- Information bietet die Spezifikation der Java-API.

Java-Standard-Packages II

- Das Package **`java.lang`** wird immer implizit importiert.
- Verwendungsbeispiel:
  `System.out.println(String)`
- `System` ist eine Klasse im Package `java.lang`
- `out` ist eine statische Variable in der Klasse `System` vom Typ `PrintStream.`
- In der Klasse `PrintStream` ist die Methode `println(String)` definiert.

- Nachvollziehbar in den JavaDocs
  http://java.sun.com/reference/api/index.html

Lernkontrolle

- Sie kennen die Möglichkeiten der Verwendung von Packages,
- die Problematik von Namenskonflikten.
- Zerlegen Sie Ihr Programm zur Umwandlung römischer Zahlen in Package und Programmcode und bringen Sie das Programm zum Laufen.