Based on:
- Seidl et al.'s "UML@Classroom", Chapter 9.1
- Miles and Hamilton's "Learning UML 2.0", Chapter 13
- https://www.uml-diagrams.org/package-diagrams-overview.html

Package Diagrams

Software Design (40007) – 2024/2025

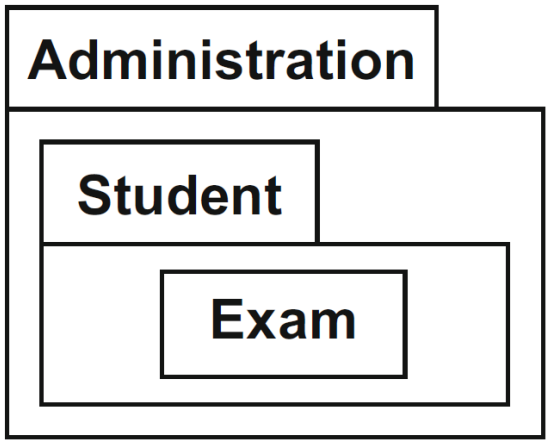
Justus Bogner
Ivano Malavolta

VU VRIJE UNIVERSITEIT AMSTERDAM

Software and Sustainability research group (S2)
Department of Computer Science, Faculty of Sciences

Why package diagrams?

- How to organize your classes?
- Many programming languages have *namespaces*, i.e., containers for higher-level structuring
- Java has *packages*: correspond to file system directories
- UML provides **package diagrams:** display packages that group model elements, e.g., classes, states, other packages, etc.
- Can be used for grouping various models or model elements

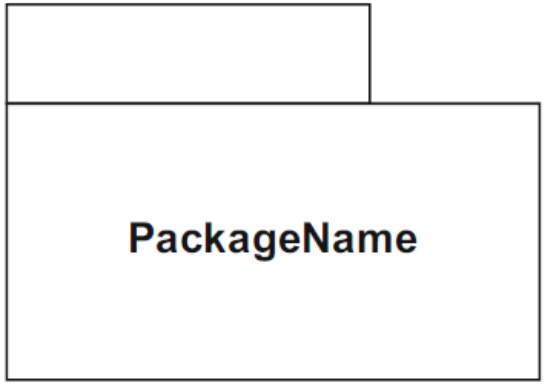


How we use package diagrams

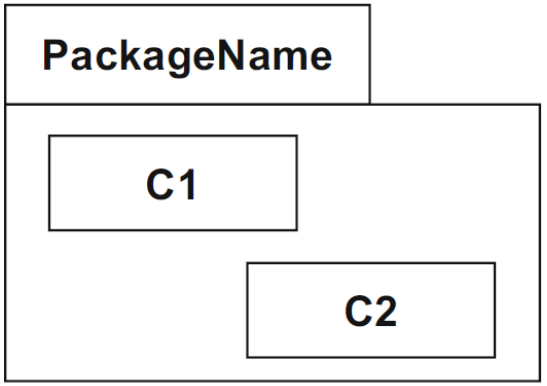
- For modeling the package structure of our Java programs, similar to *modules* → units of implementation
- *Descriptive* usage: design and document the higher-level structure for human consumers, e.g., software engineers
- Serves as a first entry point to understand the major building blocks of the project → quick overview
- Short textual description per package
- Including all classes might decrease readability
- Depending on # of classes: only include the most important ones or no classes at all

Package diagram notations

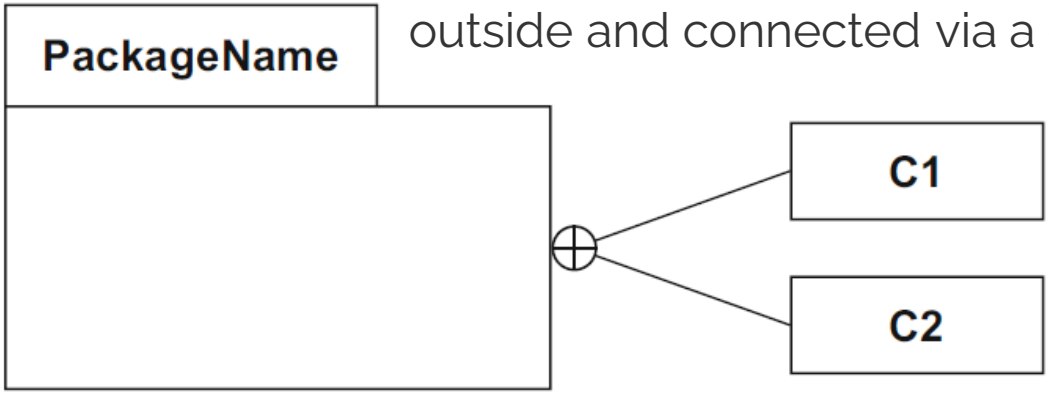
1. Displaying packages without their content:



2. With content, `name` moves to the smaller rectangle on top:



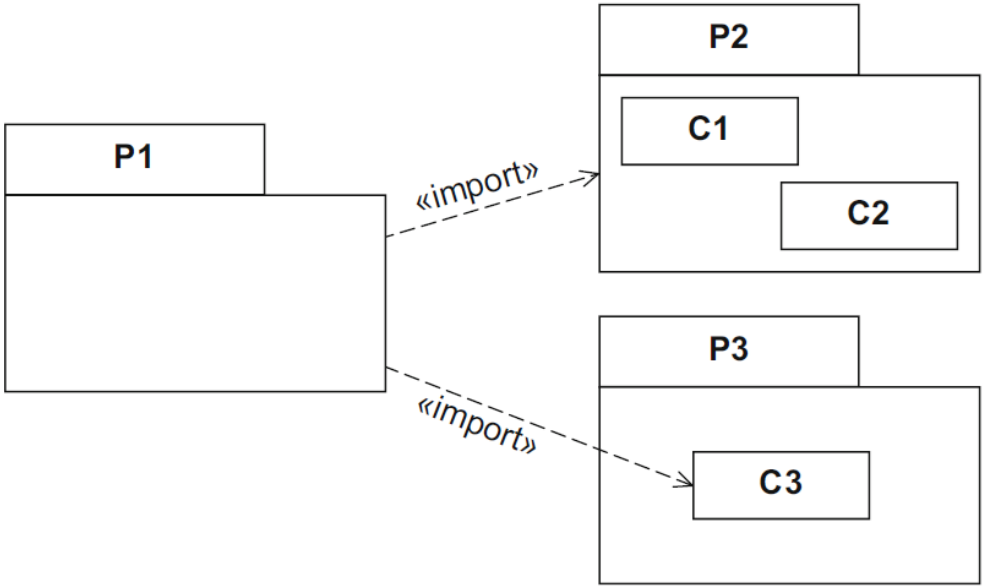
3. With a lot of content, elements can be moved outside and connected via a circle with cross:



Question: in how many packages can a class be?

Importing packages and classes

- Use `import` connectors to indicate that functionality is used in other packages



- `import` is possible for packages or individual classes
- Other connectors: `access`, `merge`, `use`
- Suggestion: focus on `import`, ignore the rest

How to name packages?

- We use package diagrams to model Java packages
- Follow common Java naming conventions!
- Only **lowercase letters and digits**
- No spaces, no special characters like hyphens
- Having no word separators encourages short names
- Oracle guide: https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html
- Java style guide from Google: https://google.github.io/styleguide/javaguide.html#s5.2.1-package-names

How to decide which packages to create?

- Two main goals for packaging:
  - Improve reasoning about and navigation through the project
  - Isolate change
- Two general approaches for packaging:
  - Package by layer
  - Package by feature

Package by layer

- Organize packages according to the types of classes and the **technical responsibility** they have, e.g., UI, business logic, data persistence, etc.
- Advantage: easy to understand, easy to navigate for small projects
- Disadvantage: not good at isolating change
- Examples of other packages:
  - `validation`
  - `parsing`
  - `controllers`
  - `businesslogic`

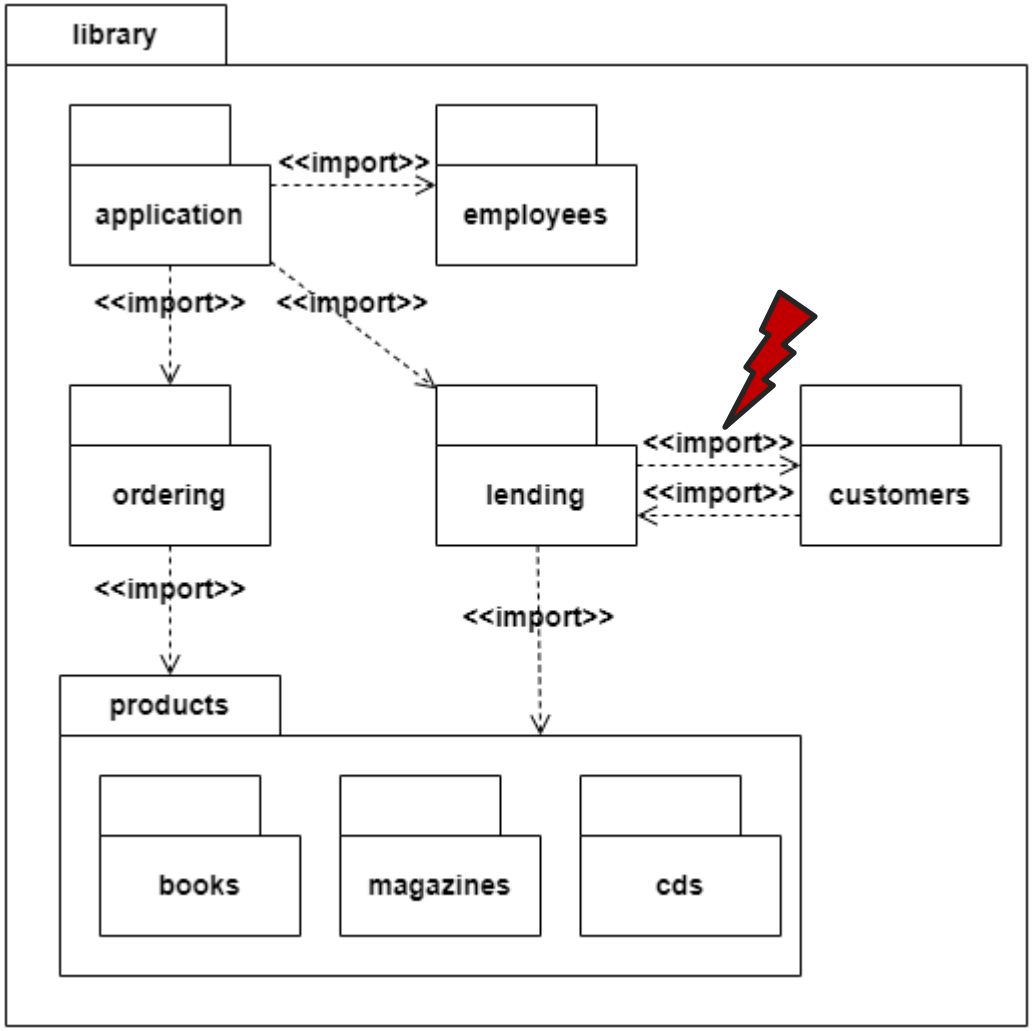
```
nl
 +- vu
     +- bookstore
         +- Application.java
         |
         +- domainentities
         |   +- Customer.java
         |   +- Order.java
         |   +- Product.java
         |
         +- persistence
         |   +- CustomerRepository.java
         |   +- OrderRepository.java
         |   +- ProductRepository.java
         |
         +- ui
            +- CustomerWidget.java
            +- OrderWidget.java
            +- ProductWidget.java
```

Package by feature

- Organize packages according to their **domain-related functionality** → *functional cohesion*
- Advantage: very good at isolating change
- Disadvantage: only easy to navigate if you know the domain well
- **Strongly advised for larger projects!**

```
nl
 +- vu
     +- bookstore
         +- Application.java
         |
         +- customers
         |  +- Customer.java
         |  +- CustomerRepository.java
         |  +- CustomerWidget.java
         |
         +- orders
         |  +- Order.java
         |  +- OrderRepository.java
         |  +- OrderWidget.java
         |
         +- products
            +- Product.java
            +- ProductRepository.java
            +- ProductWidget.java
```

Full example: library management system



Is this *package by layer* or *package by feature*?
→ by feature (except for `application` package)

Important: try to avoid cyclic dependencies!