

Secure OOP with Java

Lecture Unit - 06

Claudia Maderthaner <claudia.maderthaner@fh-hagenberg.at>

Packages

Packages

- Group classes
- Logical collection of types
- Package names must be unique

Namespace


























- Organizing projects
- Separate from other code

Package Naming Convention

- Inverted domain names
- Then append project name
- Then append further names as needed (and semantically appropriate)

Package Declaration

```
package com.example.java;
```

- ▼  src
 - ▼  org.maderthaner.timebucket.domain
 - >  Account.java
 - >  Entity.java
 - >  Project.java
 - >  Task.java
 - >  User.java
 - ▼  org.maderthaner.timebucket.repository
 - >  AccountRepository.java
 - >  GenericRepository.java
 - >  ProjectRepository.java
 - >  TaskRepository.java
 - >  org.maderthaner.timebucket.repository.exception
 - ▼  org.maderthaner.timebucket.repository.hibernate
 - >  AccountRepositoryImpl.java
 - >  GenericRepositoryImpl.java
 - >  ProjectRepositoryImpl.java
 - >  TaskRepositoryImpl.java
 - >  org.maderthaner.timebucket.service
 - ▼  org.maderthaner.timebucket.service.impl
 - >  AccountServiceImpl.java
 - >  ProjectServiceImpl.java
 - >  TaskServiceImpl.java
 - >  org.maderthaner.timebucket.value
 - >  org.maderthaner.timebucket.web





**THERE ARE NO
NESTED PACKAGES**

imgflip.com

Default Package

- Types without a package declaration belong to the **default package** (sometimes also called unnamed package).
- It is only for convenience when developing small or temporary applications or when just beginning development.

💧 Types in the default package cannot be imported in types in named packages.

Class usage

A class can use

- all classes from its own package
- all public classes from other packages

Imports

Access public classes in another package

- Fully Qualified Named (package name + class name)
- `import` statement
 - defines a shorthand to refer to the classes in another package
 - Fully Qualified Name no longer required

💡 Classes from the own packages as well as from the package `java.lang` do not need to be imported.

Import all classes of a package

```
import java.time.*;  
  
LocalDateTime now = LocalDateTime.now();
```

Import specific classes of a package

```
import java.time.LocalDate;  
  
LocalDate today = LocalDate.now();
```

```
package java.lang;

import java.io.ObjectStreamField;
import java.io.UnsupportedEncodingException;
import java.lang.annotation.Native;
import java.lang.invoke.MethodHandles;
import java.lang.constant.Constable;
import java.lang.constant.ConstantDesc;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
import java.nio.charset.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Formatter;
import java.util.List;
import java.util.Locale;
import java.util.Objects;
import java.util.Optional;
import java.util.Spliterator;
import java.util.function.Function;
import java.util.regex.Pattern;
import java.util.regex.PatternSyntaxException;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
import java.util.stream.Stream;
import java.util.stream.StreamSupport;

import jdk.internal.vm.annotation.ForceInline;
import jdk.internal.vm.annotation.IntrinsicCandidate;
import jdk.internal.vm.annotation.Stable;
import sun.nio.cs.ArrayDecoder;
import sun.nio.cs.ArrayEncoder;

import sun.nio.cs.ISO_8859_1;
import sun.nio.cs.US_ASCII;
import sun.nio.cs.UTF_8;

/** The {@code String} class represents character strings. All ...*/

public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence,
               Constable, ConstantDesc {
```

Name Conflicts

```
import java.util.*;  
import java.sql.*;
```

```
Date today; // compile error
```

```
import java.util.*;  
import java.sql.*;  
import java.util.Date;
```

```
Date today;
```

```
java.util.Date deadline = new java.util.Date();  
java.sql.Date today = new java.sql.Date(deadline.getTime());
```

Static Imports

```
import static java.lang.System.out;
import static java.lang.Math.*;

public class StaticImportsExample {
    public static void main(String[] args) {
        int number1 = 1;
        int number2 = 2;
        out.println(max(number1, number2) + " is bigger than " + min(number1, number2));
    }
}
```

Modifiers

Access Modifiers

public

member can be accessed from everywhere

protected

member can only be accessed in the package where the class was declared or by any subclass of its class inside or outside its own package

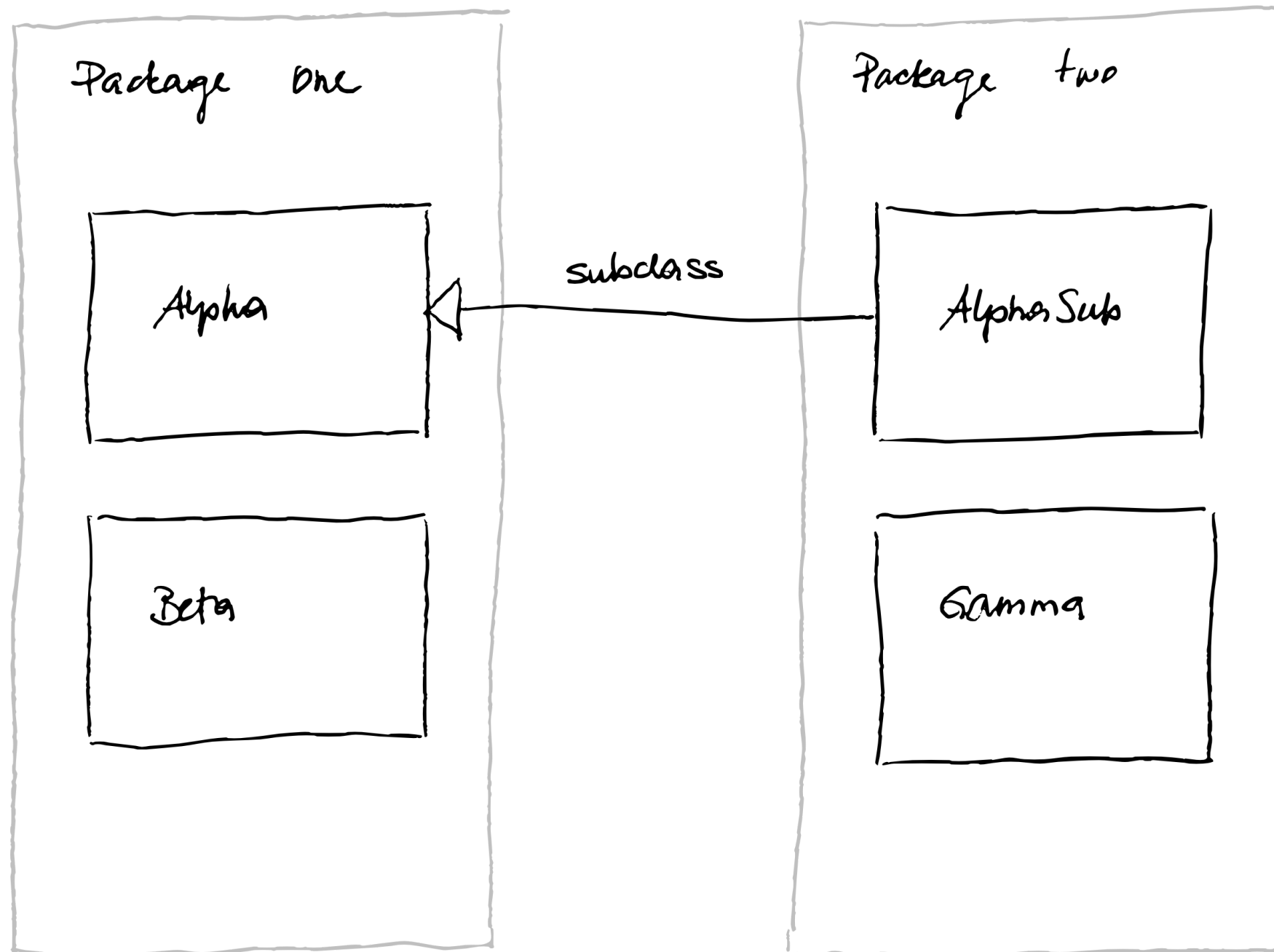
none (default/package private)

member can only be accessed from within its own package

private

member can only be accessed in the class where it is declared

Modifier	Class	Package	Subclass	World
public	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	No
none	Yes	Yes	No	No
private	Yes	No	No	No



Other Modifiers

Modifier	Scope	Note
static	class, field, method	
abstract	class, method	
final	class, field, method, parameter, local variable	
transient	field	Serialization
volatile	field	Concurrency
synchronized	methods, blocks	Concurrency
native	methods	

Java Classpath

- List of locations that are searched for Java class files
- Used by both `javac` and `java`
- A classpath may contain directories and/or JAR files

Setting the Classpath

Environment Variable Windows

```
C:\> set CLASSPATH=C:\Users\jdoe\workspace\classes;C:\jdoe\workspace\lib\util.jar;.
```

Environment Variable Linux/macOS

```
% export CLASSPATH=/home/jdoe/workspace/classes:/home/jdoe/workspace/lib/util.jar:.
```

Command Line Option

```
% javac -classpath /home/jdoe/workspace/classes:/home/jdoe/workspace/lib/util.jar:. Application.java  
% javac -cp /home/jdoe/workspace/classes:/home/jdoe/workspace/lib/util.jar:. Application.java  
% javac -class-path /home/jdoe/workspace/classes:/home/jdoe/workspace/lib/util.jar:. Application.java
```

Packaging

Java Archives (JAR)

- Provide single file for an application or library
- JAR files may contain
 - class files
 - other resources like images or documents
- JAR files are compressed (ZIP compression)

Creating JAR files

```
jar cvf app.jar *.class
```

The Manifest

- Each JAR file contains a **manifest** file
- It is called `MANIFEST.MF` and is located in the `META-INF` subdirectory of the JAR file
- The manifest describes special features of the archive



The last line in the manifest must end with a newline character.

Executable JAR Files

Create executable JAR

```
jar cvfe app.jar app.HelloWorld app/*.class
```

MANIFEST.MF

```
Manifest-Version: 1.0  
Main-Class: hello.HelloWorld  
Class-Path: .
```

Start program

```
java -jar app.jar
```

Components

POJO

"Plain Old Java Object"

- Java object not bound by any restriction like
 - extending specific classes

```
public class MyServlet extends javax.servlet.http.HttpServlet { ... }
```

- implementing a specific interface

```
public class MyEntity implements javax.ejb.EntityBean { ... }
```

- contain pre-specified annotations

```
@javax.persistence.Entity  
public class MyEntity { ... }
```

Java Beans

- Reusable in a variety of different environments
- Requirements
 - a default constructor (no-argument constructor)
 - field access with so-called "getter" and "setter" methods
 - (serializable)

Getter/Setter Conventions

Getter

- Starts with the word "get" (except for boolean fields - they use "is")
- Followed by the field name with the first letter capitalized

Setter

- Starts with the word "set"
- Followed by the field name with the first letter capitalized

```
public class Person {

    private String firstname;
    private String lastname;

    private boolean registered;

    public Person() {
        // no op
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastName() {
        return lastname;
    }

    public void setLastName(String lastname) {
        this.lastname = lastname;
    }

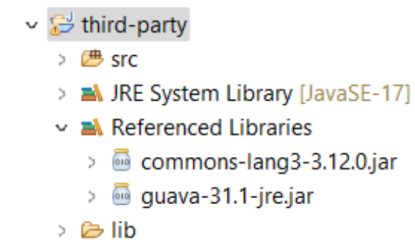
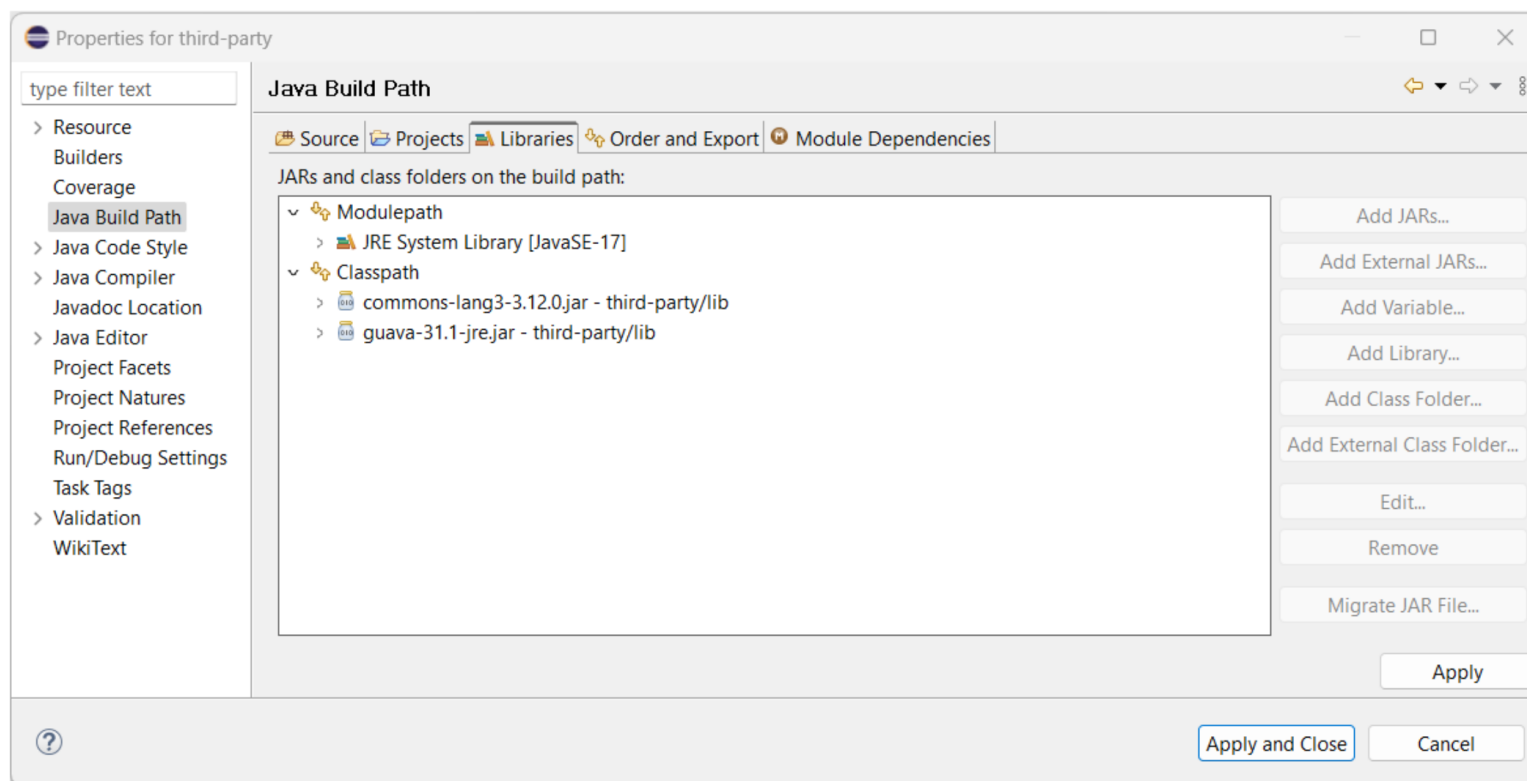
    public boolean isRegistered() {
        return registered;
    }

    public void setRegistered(boolean registered) {
        this.registered = registered;
    }
}
```


3rd-Party Libraries

3rd-Party Libraries

- Include and use code from sources outside the Java API and the project code
- Provided as JAR archives
- Included in the CLASSPATH
- Dependency Managers
 - Maven
 - Gradle



Java API

The Java API is a set of libraries that are found in the standard java distribution.

Module `java.base`

`java.lang`

provides classes that are fundamental to the design of the Java programming language

`java.math`

provides classes for performing arbitrary-precision integer (`BigInteger`) and decimal arithmetic (`BigDecimal`)

`java.util`

contains the collection framework, some i18n support and other arbitrary utils (e. g. scanning classes, base64 encoding and decoding)

`java.text`

provides classes for handling text, dates, numbers, and messages independent of natural languages

Module `java.base`

`java.io`

provides for system input and output through data streams, serialization and the file system

`java.nio`

defines buffers, charsets and channels

`java.util.concurrent`

utility classes commonly useful for concurrent programming

`java.security`

Provides classes and interfaces for the security framework

Module `java.logging`

`java.util.logging`

provides classes for core logging facilities

Module `java.sql`

`java.sql`

provides the API for accessing and processing data stored in a datasource (usually a relational database)

Modules

Modules

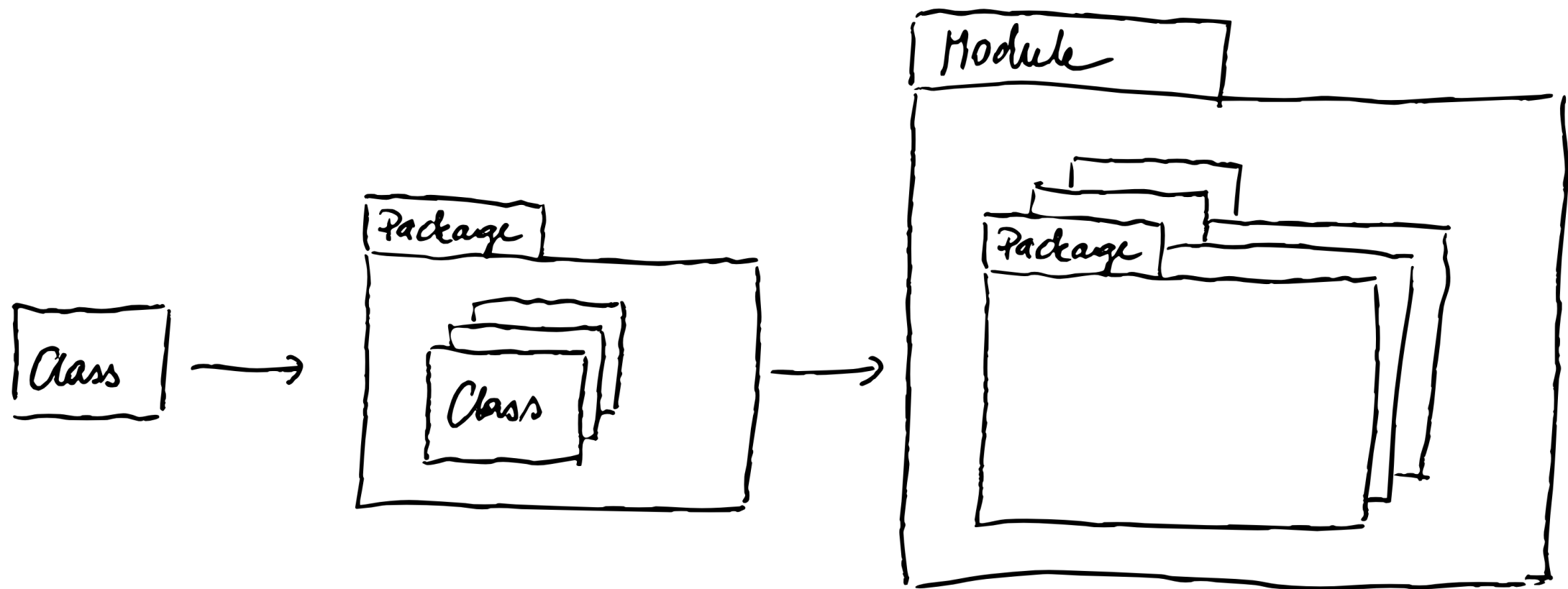
- Java Platform Module System (JPMS)
 - Since Java 9
- Strong encapsulation
- Stable abstraction
- Explicit dependencies

Packages of packages

Module

Exported Packages

Concealed Packages



Module Descriptor

- Name
- Dependencies
- Public packages
- Services offered
- Services consumed
- Reflection permissions

Naming Modules

- Module name and packages names need not be related
- There is no hierarchical relationship between modules
- Modules should follow the package naming conventions

ATTENTION: Modules names are only used in module declarations not other source files.

Module Descriptor Examples

```
module example.hello {  
  exports hello.domain;  
  exports hello.service;  
}
```

```
module example.app {  
  requires java.logging;  
  requires example.hello;  
}
```


Contact

Moodle Discussion Board

claudia.maderthaner@fh-hagenberg.at

