Conceptual Modeling and Programming in GIScience

Lecture 10: External Libraries and Exceptions

Yingjing Huang

yingjing.huang@univie.ac.at

2026S – June 22, 2026

What You Will Learn in This Lecture

- Why programmers use external libraries
- What JAR files and classpaths are
- How dependencies work
- How to judge and learn a new library
- How to read CSV data with OpenCSV
- How to show points on a map with JXMapViewer2
- Final exam scope

Why not build everything ourselves?

Car

Not every screw
from scratch

Meal

Not every ingredient
from farming

Software

Not every tool
from scratch

Libraries let us focus on our own problem

Build or borrow?

Build it yourself

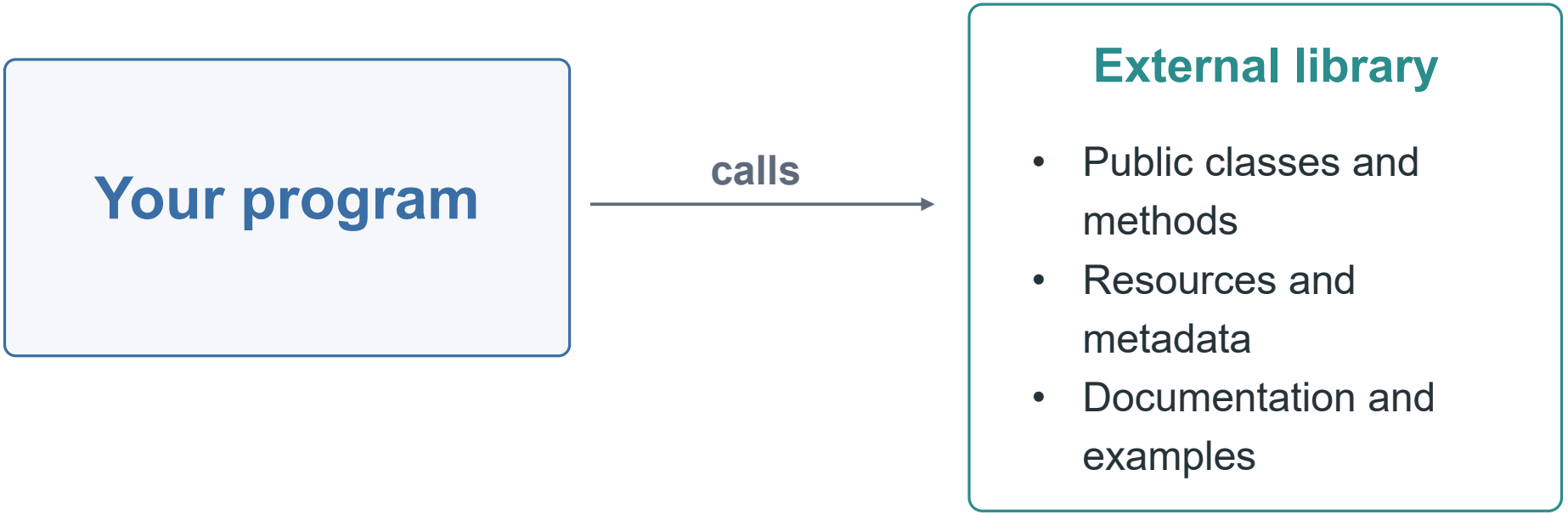
- Core logic of your program
- Small tasks that are easy to verify
- Parts where you need full control

Borrow a library

- Common technical tasks
- Details that are complex or easy to get wrong
- Parts already tested by many users
- Supporting tools rather than your main idea

Good programmers decide what is worth building

What is a library?



A library is reusable code written for other programs to call

What is a JAR file?

library.jar

JAR = Java Archive

Inside the package

- Compiled .class files
- Resources such as images or configuration
- Metadata about the package

A JAR is a file full of compiled Java classes that your program can use

How Java finds a library: classpath

Classpath = where Java looks for classes


```
import org.example.LibraryClass
```

If an import stays red, first check the classpath.

OpenCSV

depends on several supporting libraries

- commons-lang3
- commons-text
- commons-beanutils
- commons-collections
- commons-logging

These are dependencies

Dependencies form a small network of code

Why tools like Maven and Gradle exist

Manual JAR setup

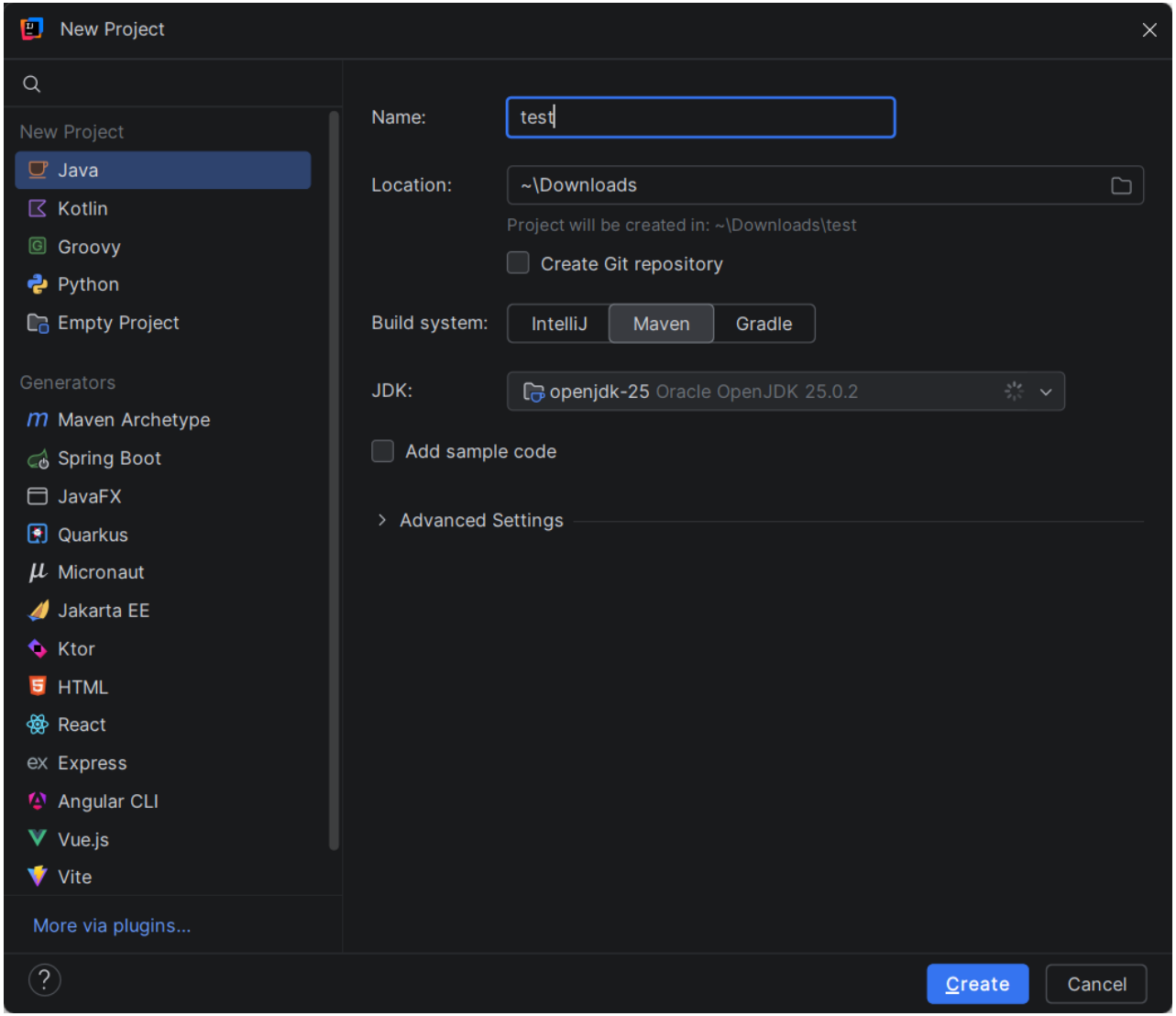
- download each JAR
- check dependencies
- add to classpath
- fix missing classes

Maven or Gradle

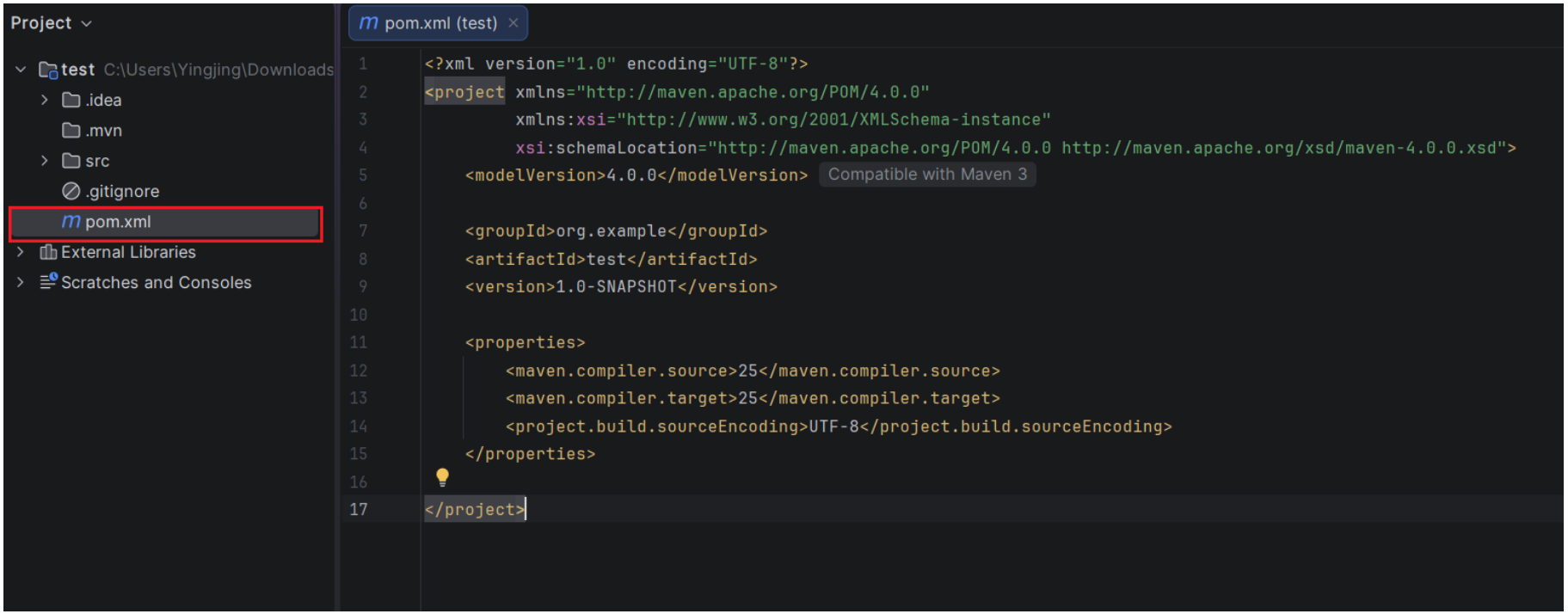
- write library coordinates
- tool downloads dependencies
- tool manages versions
- tool sets classpath

Build tools reduce dependency management mistakes

When creating a Java project, choose **Maven** as the build system



The pom.xml file is the project recipe



Maven reads this file to understand your project and libraries

Maven coordinates

```
<dependencies>
    <dependency>
        <groupId>com.opencsv</groupId>
        <artifactId>opencsv</artifactId>
        <version>5.9</version>
    </dependency>

    <dependency>
        <groupId>org.jxmapviewer</groupId>
        <artifactId>jxmapviewer2</artifactId>
        <version>2.8</version>
    </dependency>
</dependencies>
```

groupId
who provides it

artifactId
which library

version
which release

The API is the part of the library that your code can call

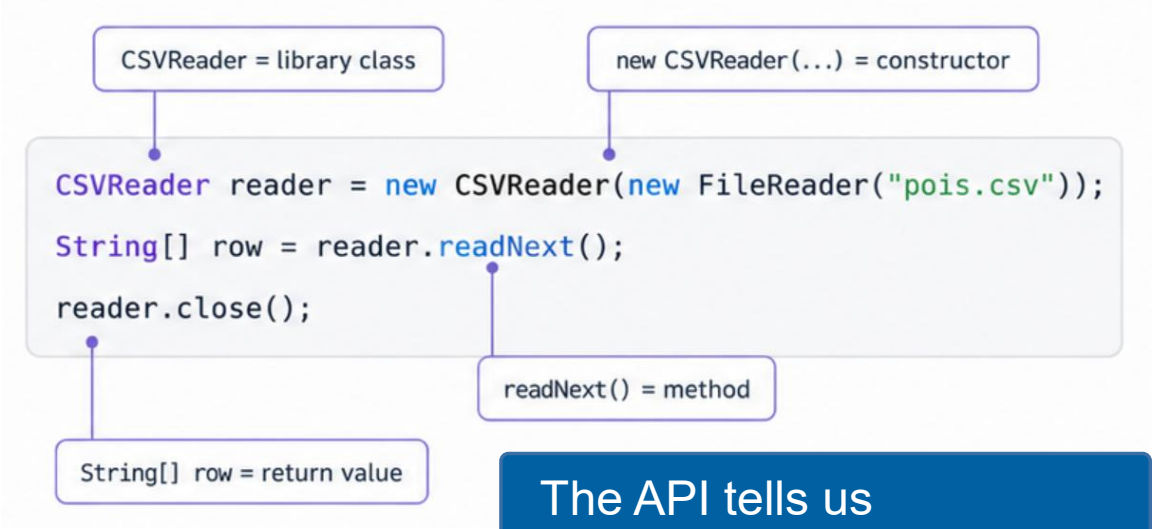
API = Application Programming Interface

OpenCSV library

Hidden inside

- CSV parsing rules
- quoted fields
- commas inside fields
- line breaks

Public API



The API tells us

what we can call

what we pass in

and what we get back

Before using a library, ask:

1. **Who maintains it?**
2. **Is it still active?**
3. **Do other people use it?**

Check maintenance, documentation, and usage before you trust a library

How to learn a new library

1. **README**
2. **Getting started guide**
3. **Official example**
4. **API reference or Javadoc**
5. **Source code if needed**
6. **Small test in your own project**

Java ecosystem

OpenCSV
read CSV data

JXMapViewer2
show maps in Swing

XChart
charts

JTS
geometry operations

GeoTools
full GIS framework

Example 1: Reading CSV data

vienna_pois.csv

```
1  name,type,lat,lon
2  Stephansdom,landmark,48.2086,16.3731
3  Schloss Schönbrunn,palace,48.1845,16.3122
4  Belvedere,museum,48.1915,16.3805
5  Riesenrad im Prater,landmark,48.2166,16.3958
```

Name	Type	Lat	Lon
Vienna State Opera	landmark	48.203	16.369
Stadtpark	park	48.204	16.380
MuseumQuartier	museum	48.203	16.359

CSV = Comma-Separated Values

- plain text
- one record per line
- common in GIS and open data

From CSV rows to POI objects

CSV row

["Stadtpark", "park", "48.204", "16.380"]

parse and construct

POI object

- name: Stadtpark
- type: park
- lat: 48.204
- lon: 16.380

Data enters as a simple external format

We convert it into objects so the rest of the program is cleaner

OpenCSV in code: rows become POI objects

```
try (CSVReader reader = new CSVReader(
        new FileReader(path, StandardCharsets.UTF_8))) {

    String[] row;
    int lineNumber = 0;
    while ((row = reader.readNext()) != null) {
        lineNumber++;
        if (lineNumber == 1) {
            continue; // header row
        }

        double lat = Double.parseDouble(row[LAT].trim());
        double lon = Double.parseDouble(row[LON].trim());
        model.add(new POI(row[NAME], row[TYPE], lat,
lon));
}
```

Library work

- CSVReader knows CSV rules
- readNext() returns one String[] per row
- quoted commas stay inside one field

Our work

- skip the header
- parse lat/lon as numbers
- construct one POI object

The boundary is clear: OpenCSV parses the file; our code models the domain.

What OpenCSV saves us from

Without OpenCSV

- manual split
- quotes
- escaped characters
- line breaks
- inconsistent files

With OpenCSV

- call library
- get rows
- focus on POI logic

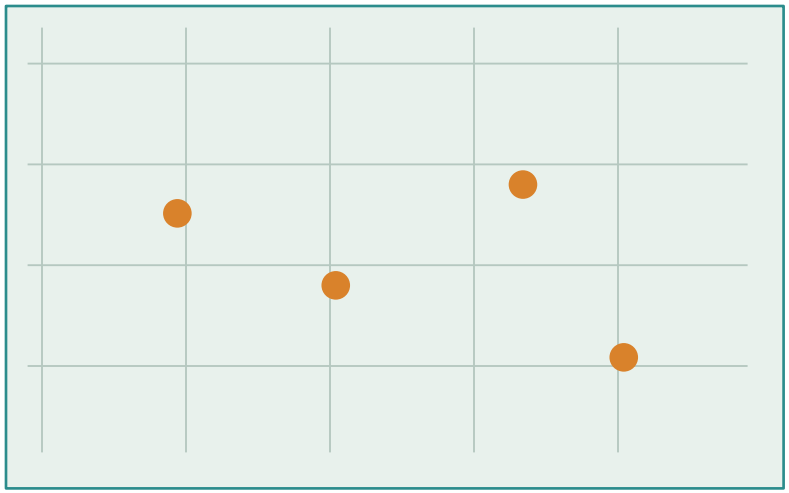
The library handles the general problem

We handle our specific problem

Example 2: Showing points on a real map

Blank Swing panel

Real map context



A map library turns coordinates into an interactive map display

OpenStreetMap

OpenStreetMap = OSM

Free and open

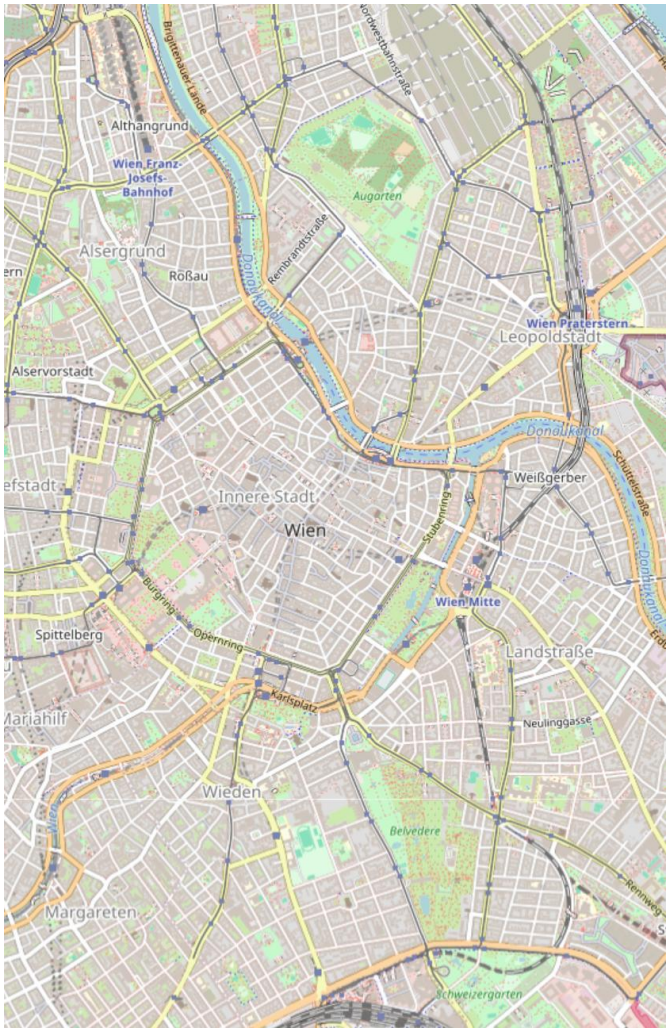
Geographic data edited by contributors around the world

Map tiles

Small images requested and assembled by the map component

Important distinction

OSM is the data source. JXMapViewer2 is the Java component



A reusable map widget for Java Swing applications

Map display

show map tiles inside a Swing application

Interaction

zoom and pan without writing all mechanics yourself

Overlays

Draw markers or other layers on top of the map

Do not confuse JXMapViewer2 with JMapViewer

GeoPosition, Waypoint, and POI

The model stores data

The view creates map objects

JXMapViewer2 in code: POIs become map markers

```
private void showPois(List<POI> pois) {
    Set<PoiWaypoint> waypoints = new HashSet<>();
    for (POI poi : pois) {
        waypoints.add(new PoiWaypoint(poi));
    }

    WaypointPainter<PoiWaypoint> painter = new WaypointPainter<>();
    painter.setWaypoints(waypoints);
    painter.setRenderer(new PoiMarkerRenderer());
    setOverlayPainter(painter);
    repaint();
}

private static final class PoiWaypoint extends DefaultWaypoint {
    private PoiWaypoint(POI poi) {
        super(new GeoPosition(poi.getLat(), poi.getLon()));
    }
}
```

Conversion

- POI keeps plain doubles
- GeoPosition is the map-library type
- DefaultWaypoint makes it drawable

Overlay

- WaypointPainter draws markers
- setOverlayPainter attaches them
- repaint refreshes the view

What JXMapViewer2 does for us

- map tile requests
- zoom behaviour
- pan behaviour
- coordinate transformation
- marker redraw
- map background

We focus on

Vienna POIs and

The application logic

The library handles the map mechanics

Extended!

July 31

Individual SIR assignment

- Submit through Moodle
- Check that all required files are included

June 29 · Next Week

Part A

- 25 single-answer multiple-choice questions
- 50 points

Part B

- 2 code-reading questions
- 20 points

Part C

- 2 short code-writing questions
- 30 points

Exam scope: Weeks 1 to 5

Week	In scope	Not required
1	primitive types; String; integer division; basic control flow; …	IntelliJ; debugging; breakpoint; binary system; documentation; …
2	class; object; field; method; encapsulation; private field; controlled access; constructor; this; new; parameter; pass-by-value; …	ASCII, hexadecimal; IPO model; BoundingBox; …
3	Inheritance; is-a relationship; ==; static; ArrayList; for-each loop; toString(); …	package; protected; enum; delegation; OGC; WKT; …
4	polymorphism; dynamic dispatch; runtime type; compile-time type; superclass; subclass; super; super(…); instanceof; abstract class; @override;…	Random; seed; this(…); downcasting; diamond problem; …
5	interfaces; method signature; simultaneous update; swap; …	default methods; UML; cellular automata theory; Game of Life rules; …

Exam scope: Weeks 7 to 10

Week	In scope	Not required
7	GUI; Swing; event-driven program; ActionListener; button click; control flow; …	layout manager; AffineTransform; MouseListenr; …
8	MVC; file I/O; line-by-line reading; exception handling; try-with-resources; …	ImageIO; hex colors; radiometric resolution; PGM; …
9	none	Ripley's K; spatial statistics
10	external library; reusable code; tested code; maintained code; …	JAR; classpath; dependency; Maven; OpenCSV API; OpenStreetMap; …