



The University of
Nottingham

UNITED KINGDOM • CHINA • MALAYSIA

Lecture 04B

Introduction to Maven and Gradle

Horia A. Maior and Marjahan Begum



COMP2013

Developing Maintainable Software

Lecture 04B

Introduction to Maven and Gradle

Horia A. Maior and Marjahan Begum

Topics for this Week



- Lecture 04A:
 - OO Analysis and Design (OOA/D) with UML
- Lecture 04B:
 - Early Module Feedback (5 min)
 - Week 4 Labsheet
 - Build Tools (Maven and Gradle)
 - Coursework Preview
- Lab 04 (Tomorrow):
 - Virtual UML Speed Refactor/Extend Challenge for Group Project groups



Module Assessment Sheet 2023-24



Module Convenor(s)	Horia Maior, Marjahan Begum
Module Code	COMP2013
Module Credits	20

Term-time Assessment (TTA)

Assessment Name	TTA1 --- Coursework
Assessment Type	Coursework
Description and Deliverable(s)	Description: Maintaining and extending existing software Deliverables: Git activity; refactored and extended code base on GitHub; documentation (readme.md, Javadocs, class diagram reflecting changes); zip file of your project; video showing the game in action and explaining your maintenance work
Frequency, dates & workload	<input checked="" type="checkbox"/> Individual <input type="checkbox"/> Weekly <input type="checkbox"/> Fortnightly <input type="checkbox"/> Custom Release date: 31/10/2023 Submission date: 12/12/2023 Weight (%): 75 Workload (h): 75
Late Policy	<input checked="" type="checkbox"/> UoN Default late policy. <input type="checkbox"/> Custom: Enter details of custom late policy (if applicable)
Feedback Mechanism & Date	Feedback date: <input checked="" type="checkbox"/> Within 15 working days of submission <input type="checkbox"/> The expected date for feedback is Please enter date here. Feedback mechanism: We aim to provide written individual feedback in Moodle within 15 working days of deadline, but might need more time due to the class size.
Assessment Criteria	We plan to split marks as follows: 10% for git use (e.g. push, branch, merge, providing .gitignore) 30% for refactoring 30% for additions 10% for documentation (readme.md, Javadoc, class diagram) 20% for the demonstration video, git, documentation, showing the software running, and explaining your refactoring activities and your additions
ECs	Note that ECs are granted based on personal circumstances and are not guaranteed. <input type="checkbox"/> Extensions are not possible due to practical limitations <input type="checkbox"/> Assessment component can be disregarded in exceptional circumstances: Please specify details <input checked="" type="checkbox"/> Up to a maximum of 1 week per 10 credits (actual extension given will depend on specific circumstances) <input type="checkbox"/> Cut-off date for extensions applies: Please specify date <input type="checkbox"/> Other: Please provide details

Exam (E)

Examination Type	E1 --- In person ExamSys
Weight (%)	25
Duration (h)	1.5
Other Information	Other information

Module Information

Timetable of Activities

Teaching Week	W/C	Topic	Lead	Format		
				Lectures Mondays, 4pm-6pm	Labs Fridays 3-5pm	
1	02/10/2023	Introduction to DMS	Horia/Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	B
2	09/10/2023	More Advanced Java Topic	Horia	BS South B52 + Recording	Lab A07,A32,Atrium	B
3	16/10/2023	Maintainable GUI Development (1/2)	Horia	BS South B52 + Recording	Lab A07,A32,Atrium	B
4	23/10/2023	Design Principles and Patterns	Horia/Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	B
5	30/10/2023	Maintainable GUI Development (2/2)	Horia/Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	B
6	06/11/2023	Coding and Repository Tools for DMS	Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	B
7	13/11/2023	UML for the Maintainer	Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	B
8	20/11/2023	Refactoring Skills	Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	B
9	27/11/2023	Open Source	Horia	BS South B52 + Recording	Lab A07,A32,Atrium	B
10	04/12/2023	Guest Lecture	Horia/Marjahan	BS South B52	Lab A07,A32,Atrium	B
11	11/12/2023	Revision and Exam Prep	Marjahan/Hoira	BS South B52 + Recording	Lab A07,A32,Atrium	B

Last updated on 29/09/2023.

Module Assessment Sheet

[Module Assessment Sheet] is now available!

Early Module Feedback



Lecture Recordings

Links to recordings will be available, just beside/below each lecture/lab title.

Microsoft Teams Access

If you do not have access to COMP2013 - Developing Maintainable Software (COMP2013 UNUK) in Microsoft Teams. Please use the [link](#) to participate



Early Module Feedback 2023-24 is now open

Module Information



Timetable of Activities

Teaching Week	W/C	Topic	Lead	Format		
				Lectures Mondays, 4pm-6pm	Labs Fridays 3-5pm	Thursd
1	02/10/2023	Introduction to DMS	Horia/Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
2	09/10/2023	More Advanced Java Topic	Horia	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
3	16/10/2023	Maintainable GUI Development (1/2)	Horia	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
4	23/10/2023	Design Principles and Patterns	Horia/Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
5	30/10/2023	Maintainable GUI Development (2/2)	Horia/Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
6	06/11/2023	Coding and Repository Tools for DMS	Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
7	13/11/2023	UML for the Maintainer	Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
8	20/11/2023	Refactoring Skills	Marjahan	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
9	27/11/2023	Open Source	Horia	BS South B52 + Recording	Lab A07,A32,Atrium	BS South
10	04/12/2023	Guest Lecture	Horia/Marjahan	BS South B52	Lab A07,A32,Atrium	BS
11	11/12/2023	Revision and Exam Prep	Marjahan/Hoira	BS South B52 + Recording	Lab A07,A32,Atrium	BS South

Last updated on 29/09/2023.

Module Assessment Sheet

[\[Module Assessment Sheet\]](#) is now available!

Early Module Feedback



[COMP2013 Early Module Feedback 2023-24](#)

We want to hear from you!

Survey is anonymous.

Lecture Recordings

Links to recordings will be available, just beside/below each lecture/lab title.

Microsoft Teams Access

If you do not have access to COMP2013 - Developing Maintainable Software (COMP2013 UNUK) in Microsoft Teams. Please use the [link](#) to participate.



Week 4 Labsheet

Week 4 labsheet



- Practice your UML Design skills
- Work in your already existing SE Groups from COMP2002
- Exercise useful for your coursework
- Exercise useful also for your COMP2002 group projects. This will be great for your design blueprints in your reports.



LAB 4: "VIRTUAL UML SPEED REFACTOR/EXTEND CHALLENGE"



Aims:

- Practice your object-oriented analysis and design/maintenance skills
- Practice working in small design teams
- Consider how to design software with lower maintenance effort in the future

PREPARATION AND POST PROCESSING

This lab session is a **group exercise**, and you are asked to work with your **COMP2002 group project team members** on this! The way you do it, is up to you. You can go for face-2-face meetings in A32, any other location, or use Teams and do it online (using your group project Teams site). In the latter case you should use a virtual whiteboard, so that you can draw diagrams together. If you have questions, ask the lab helpers if you are in Lab A32 or ask us on the Teams Questions channel, if you are elsewhere.

To set the scene, imagine you are competing in a "**Virtual UML Speed Refactor/Extend Challenge**". Here are the rules: Overall you have **120 minutes to fulfil the given task**. It is advised that you split up your precious time into two chunks: Use 90 minutes for the design and 30 minutes for summarising the outcome in form of a Power Point presentation. To get through the entire task in time you might want to have an initial discussion with the whole teams and then split up into smaller teams (perhaps pairs, if that fits) to do some 15-minute sprints. After each sprint you then reconvene briefly as a group and give some feedback to each other's outputs.

At the end of the lab session each group project team is welcome to submit a Power Point presentation (**one per group**) with a summary of the outputs you produced during the lab session (use case specification and screenshots / photos of UML diagrams). Your submissions will not be marked, but we are very interested to see what innovative and adventurous designs you are coming up with within the short time given. The submission link on Moodle will only be open until the end of the lab session. Please understand, that due to time constraints, we will not be able to provide individual feedback for each of the submissions.



You have been asked by the Student Union to develop a new "Student-Internship Match" application for them. A legacy system exists but it was abandoned some years ago. The idea is to match students with the right skillset to internship offers provided by local companies.



Before students and companies can use the application, they need to register and once they received their username/password they can log in. After registration students and companies need to add a profile which will be stored in a database (together with their login details). The profile can be edited at any time. Companies can then start posting job offers which will be stored in the database. Student profiles consist of student details and skills while job offers consist of company details, a job description, and requirements. Skills and requirements are to be submitted in form of predefined keywords so that they are easy to be matched in later searches.

Once the data has been stored in the database students as well as companies can then query the database and a query engine will try to match job offers with student profiles and vice versa. At a later stage the student union is planning to replace the keyword search with an intelligent query engine. It is a requirement that the application is designed following object-oriented principles and that it is easy to maintain and extend. If you have any ideas for extensions in order to improving the usability of the application, you are encouraged to add these to your design.

The legacy system was operated by a secretary who would handle all the data input and search requests. The only surviving design artefact from this system is the class diagram shown in Figure 1.

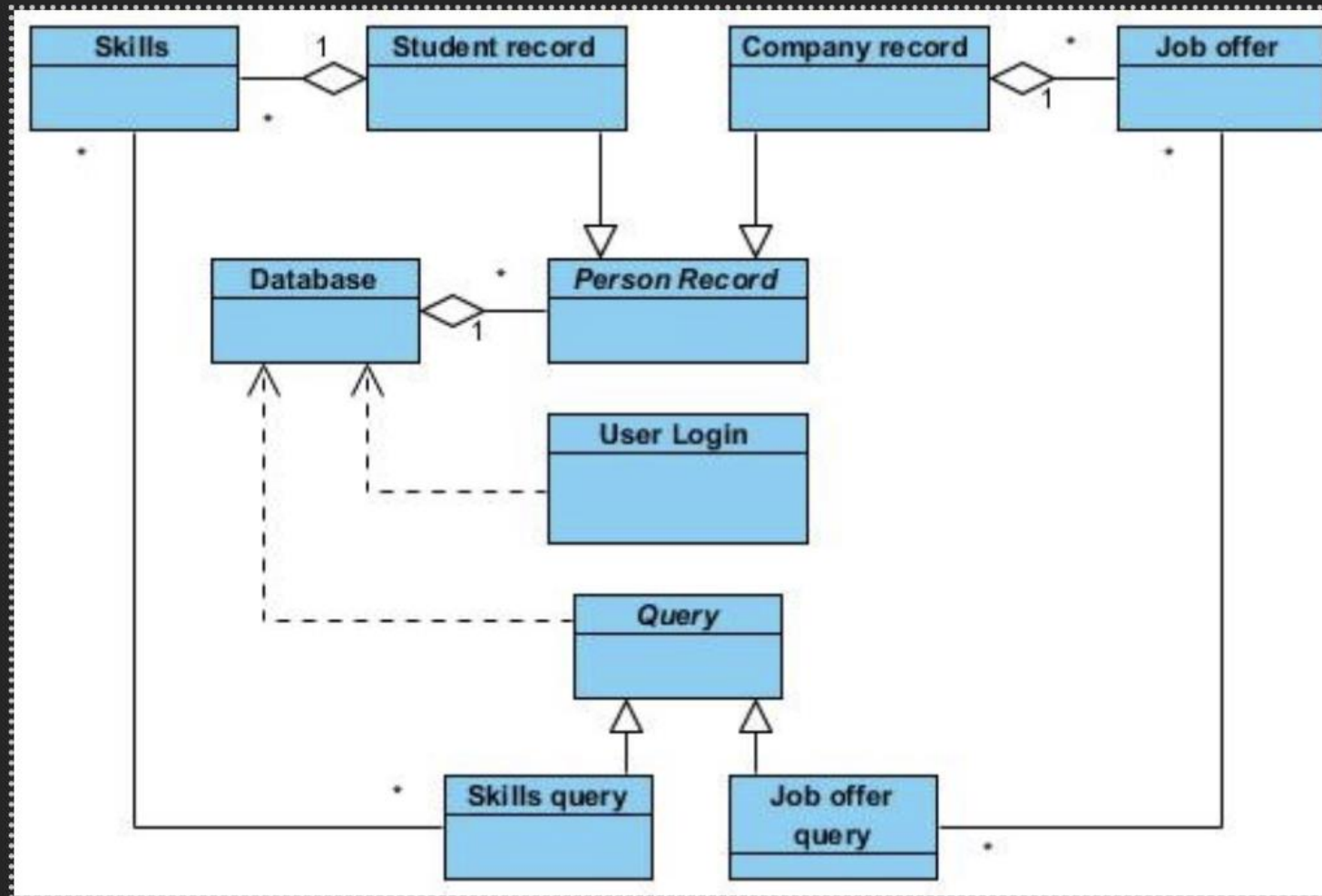


Figure 1: Legacy system class diagram



YOUR TASK

Find your **COMP2002 group project team members** and work on the object-oriented analysis and design for the refactoring and extension of the "Student-Internship Match" application.

You are asked to produce the following for the new version of the application:

- Use-case diagram for the described application
- Complete use case specification for a nontrivial use case (e.g. "search for jobs")
- Activity diagram of the same use case
- Sequence diagram of the same use case
- Class diagram
 - Classes including key attributes and operations
 - Relationships
 - Multiplicity indicators
- Nontrivial state machine diagram for one of the classes



SOME TIPS

In case you want to use Visual Paradigm for creating your diagrams, here are the links:

- Visual Paradigm Community Edition:
 - <https://www.visual-paradigm.com/download/community.jsp> <https://www.visual-paradigm.com/download/community.jsp>
- Visual Paradigm User Guide:
 - <https://www.visual-paradigm.com/support/documents/vpuserguide/>
- Visual Paradigm Online:
 - <https://online.visual-paradigm.com/diagrams/>
Scala Build Tool)

To improve maintainability in the future, think about these points when you design your system:

- Keep it as simple as possible (KISS principle)
- Keep similar functionality together, and different functionality apart (encapsulation)
 - This is also known as "high cohesion, and loose coupling"
- How easy would it be to change a module or feature in the future?
- How good are your diagrams at explaining the system to a new programmer?



Coursework preview

COMP2013 Coursework Task Description



This coursework is contributing 75% to your overall 20 CR COMP2013 assessment mark and it will be marked out of 100.

Recommendations: We recommend dedicating approximately 40-60 total hours on the coursework. We expect 30-40 hours of work for those who are aiming for a pass (40+) and 60 or more hours of work for those that are aiming for a first (70+). Please keep in mind that the skill level varies quite a bit in a class with over 350 students, and that the exact number of hours depends on your individual skill level. To help you, we have less required lectures in the last two weeks of term and some of the lab sessions are dedicated to your coursework.

Deadline:

- Milestone 1 – Friday the 24th of November 2023
- Milestone 2 – Tuesday 12th of December 2023

Friday 12/12/2022 @ 3pm (to be confirmed)



Assessment: The marks will be split as follows:

- 10% for git use (e.g. push, branch, merge, providing .gitignore)
- 30% for refactoring
- 30% for additions
- 10% for documentation (UML + Software Documentation (readme file + source code))
- 20% for the demonstration videos (Milestone 1 and 2), explaining your: git + documentation, maintenance activities and additions

Questions: Questions can be asked in person during the lab sessions and on Teams. Please read the questions that have already been asked on Teams before you ask or post yours, to avoid duplication. We will try to answer your questions as quickly as possible.



Requirement Specification - Overview

- Basic Software Maintenance on the provided code
- Better organisation of the code
- Excellent Design Documentation
- Extend the delivered code base by adding additional features (some ideas will be provided, but additional features are always a bonus)
- Refactoring the code by adding design patterns (e.g. MVC), etc.
- Version Control with Git
- Testing Strategy
- Object Oriented Design and Implementation
- Improve legacy codebase
- Present your work well





Marking Scheme Overview

- Git (10%)
- REFACTORING (30%)
- ADDITIONS (30%)
- DOCUMENTATION (10%)
- DEMONSTRATION AND PRESENTATION OF THE WORK (20%)



Marking Scheme Overview

- Milestone 1
 - Understand the code base
 - Set up your code in your IDE
 - Set up Git
 - Initial Class Diagram (Refactoring)
- Milestone 2
 - Everything else



ANY
QUESTIONS
?

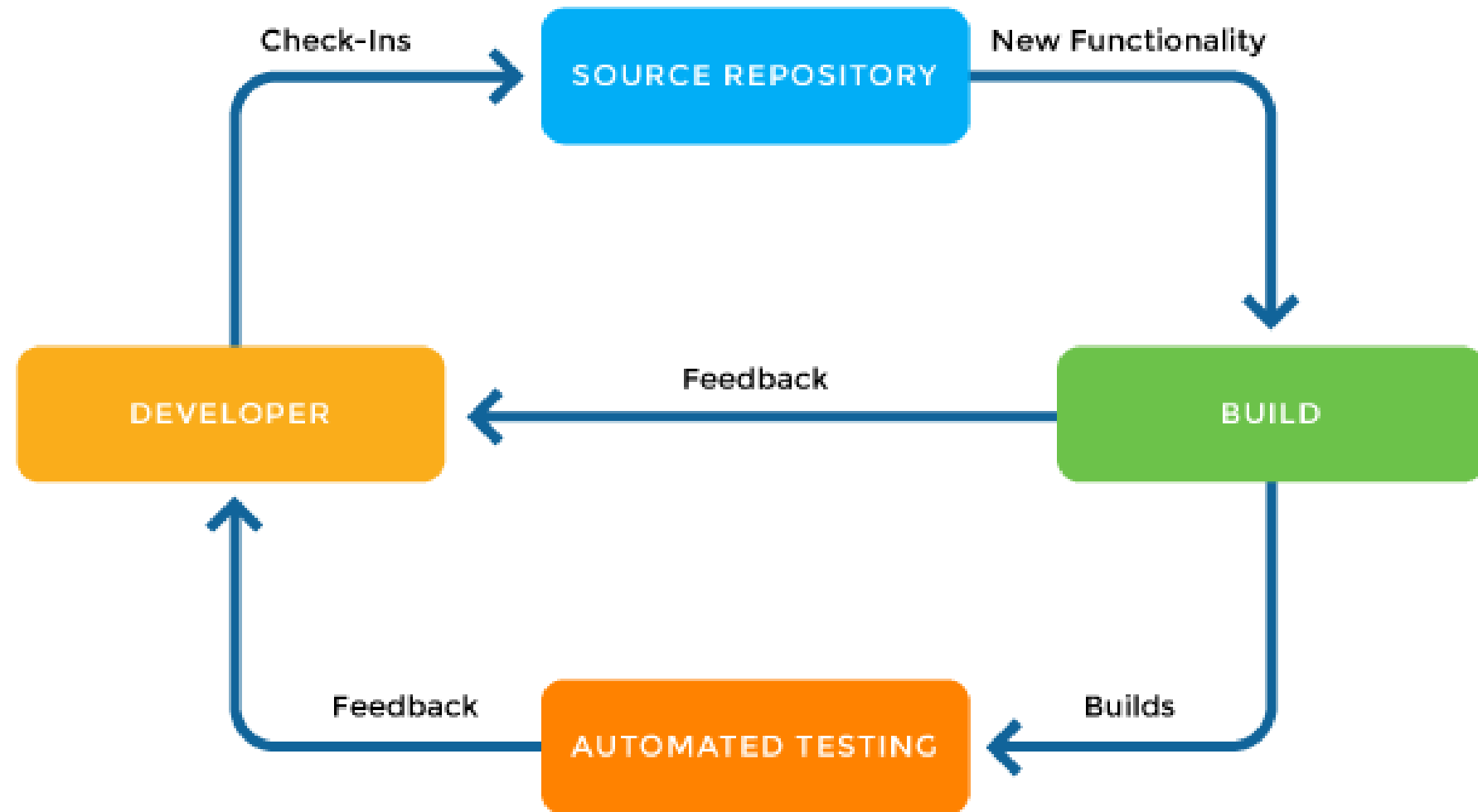




Build Tools

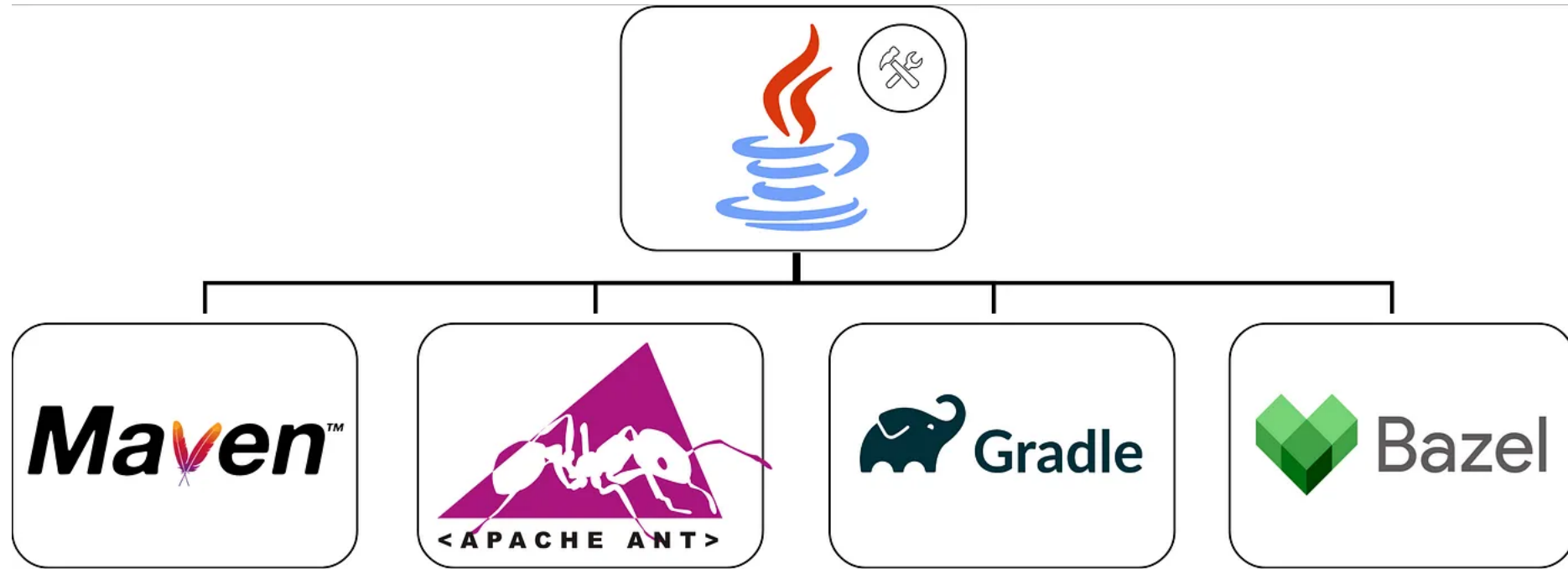


Question:
What are
Build Tools in
Programming
Languages?



Benefits of Build tools

- Project Compilation
- Reduces errors from manually running steps,
- Increases the consistency of the process.
- Dependencies
- Allow for better maintenance (keep a build log)
- Automated testing
- Deployment
- Supports documentation
- Repository management
- ...

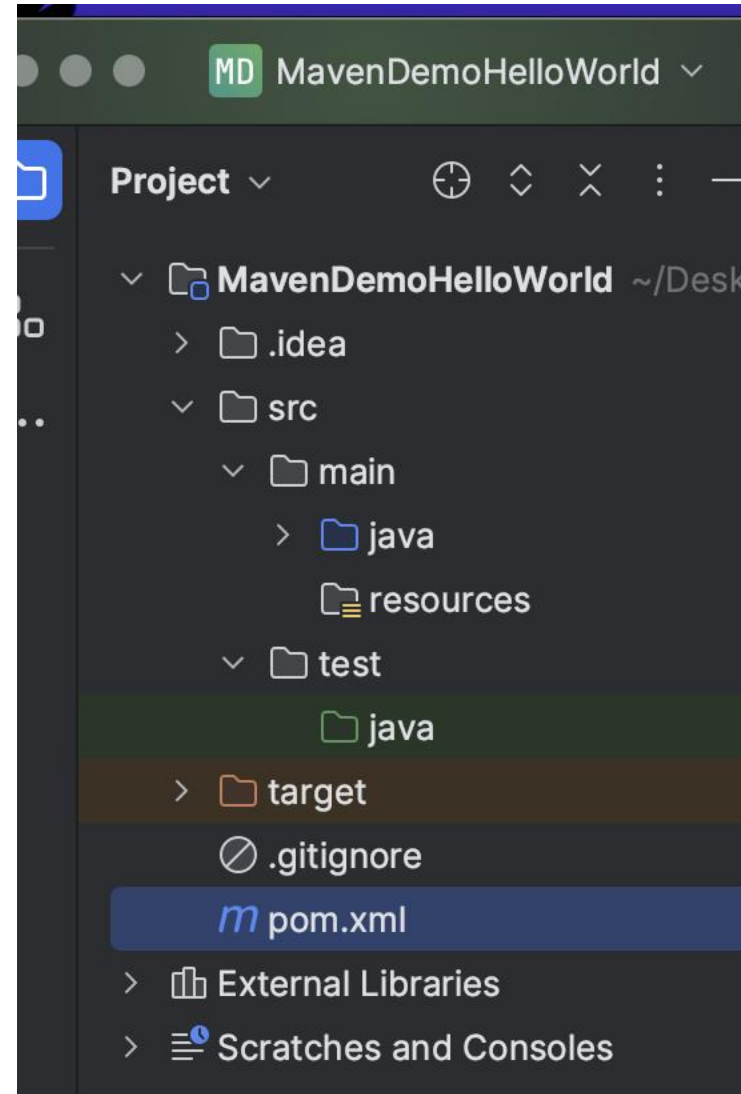


What is Maven?

- A build tool

What is Maven?

- A build tool
- Maven builds are structural

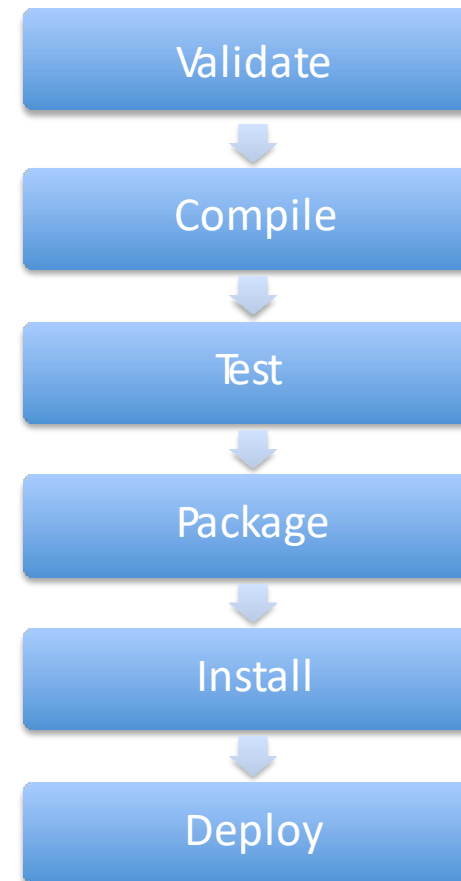


What is Maven?

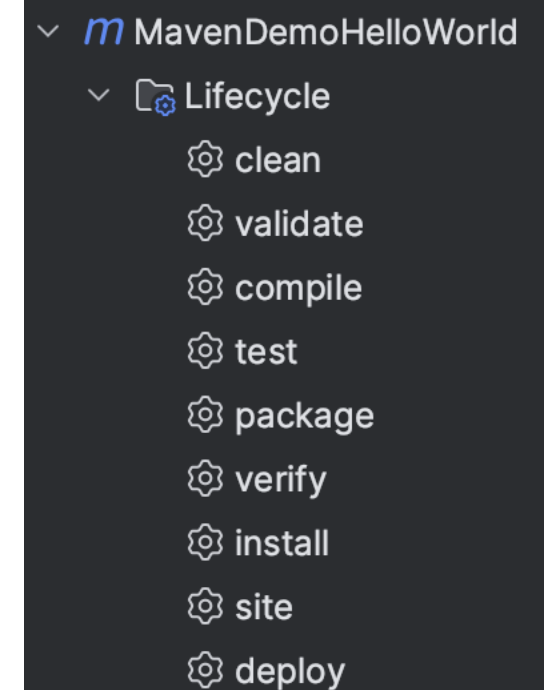
- A build tool
- Maven builds are structural
- Manages dependencies
 - You describe your dependencies, Maven downloads and includes all you need in your build path
- Manages repositories
 - Your projects share a common set of artifacts/jar files
 - No need to check in JAR files to version control

Maven Project Management

- Maven builds follows a lifecycle, using phases
 - Each lifecycle phase includes the ones above it
- (Package starts with compile, then test, then package)
- Plugins can customize or hook-in to the phases
 - Key Maven lifecycles
- **Default** – your normal lifecycle
- **Clean** – issued during the mvn cleanup command
- **Site** – issued during the mvn site command




The Default Lifecycle



Maven Demo: Getting Started

What is a POM file?

- Project Object Model (POM)
- Contains configuration information about your project
- Key items
 - Project Identification
 - Dependencies
 - Plug-Ins
 - Other Settings

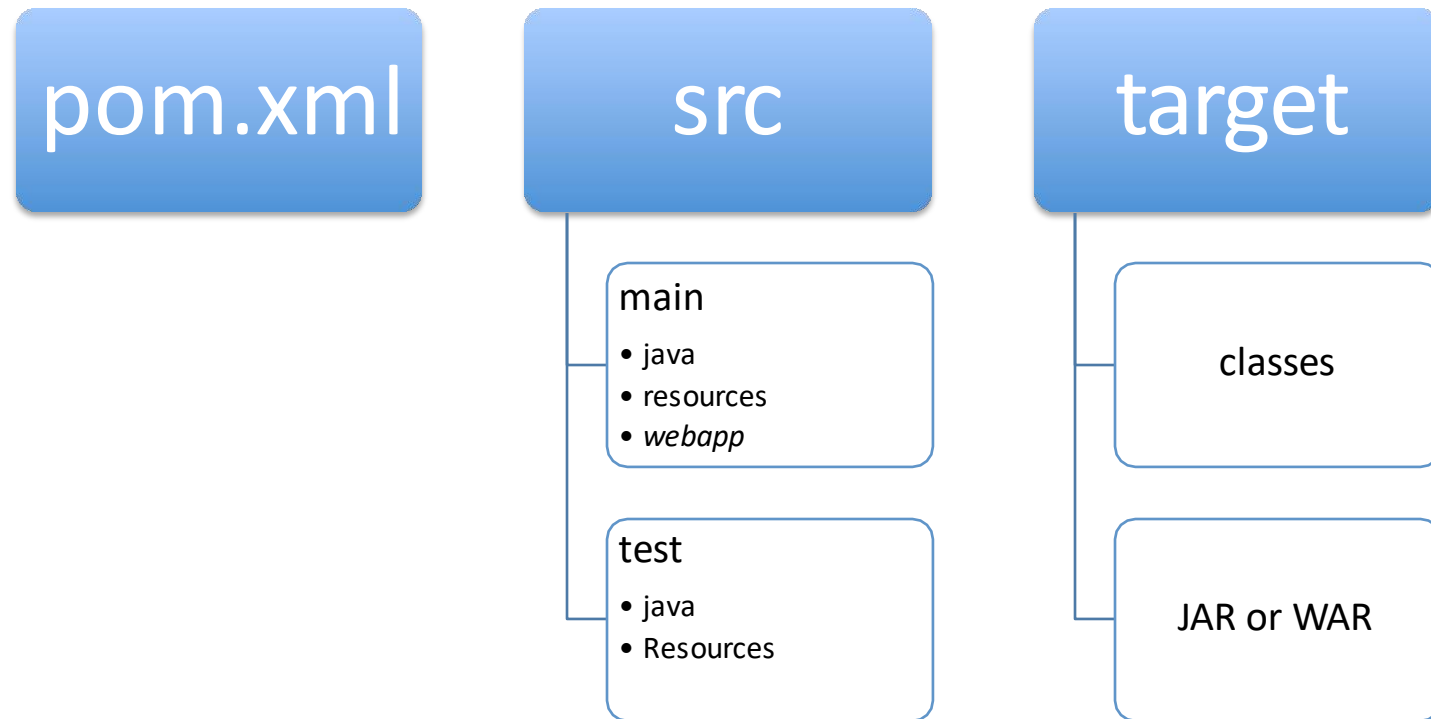
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>com.COMP2013</groupId>
8      <artifactId>MavenDemoHelloWorld</artifactId>
9      <version>1.0-SNAPSHOT</version>
10
11     <properties>
12         <maven.compiler.source>20</maven.compiler.source>
13         <maven.compiler.target>20</maven.compiler.target>
14         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     </properties>
16     <dependencies>
17         <dependency>
18             <groupId>junit</groupId>
19             <artifactId>junit</artifactId>
20             <version>4.13.2</version>
21             <scope>test</scope>
22         </dependency>
23     </dependencies>
24     
25 </project>
```



What happens when you Set Up a Maven Project?

- Mavens build a project skeleton
 - Maven used its' conventions to place files in the right places
- Let's look at the directory structure...

Directory Structure, Basic Maven Project



Using Maven Commands

- The maven command (mvn)
 - mvn clean – removes files in target
 - mvn compile – compiles a project
 - mvn test – Runs all tests in the src/test directory
 - mvn package – builds the final target artifact (JAR, WAR)
 - mvn install – Installs the target artifact in your local repository
 - mvn deploy – Deploys the artifact (if configured)

Helpful Maven Reports

- **mvn site** -- Generates a web site report in target/site/index.html
- **mvn pmd:pmd** – runs a PMD (programming mistake detector) report, a source code analyzer for finding common flaws, output in target/site/pmd.html
- **mvn cobertura:cobertura** – runs a cobertura code test coverage report, the percentage of branches/lines accessed by unit tests, output in target/site/cobertura/index.html
- **mvn jdepend:generate** – runs a report on coupling between packages, output in target/site/jdepend-report.html
- **mvn checkstyle:checkstyle** – runs a checkstyle report
- **mvn javancss:javancss-report** – Runs a JavaNCSS (non commenting source statements) report to show method complexity, roughly equivalent to counting “;” and ‘{’ characters in Java source file
- Note: all of these and more can be configured in the <site> section of the maven build... YMMV.

Managing Dependencies

- Key fact: Maven projects have ONE Artifact (JAR, WAR)
 - Can build projects that depend on other projects
 - Can build parent projects that build child projects
 - Can depend on other open source libraries and frameworks
- Manage all of this via the <dependencies> tags

- Adds dependency to jUnit
 - Downloads 3.8.1 of Junit and stores in your maven repository
 - Only uses in path for testing

```
<dependencies>  
  <dependency>  
    <groupId>junit</groupId>  
    <artifactId>junit</artifactId>  
    <version>3.8.1</version>  
    <scope>test</scope>  
  </dependency>  
</dependencies>
```

Dependency Tips...

- Look for public dependencies on www.mvnrepository.com
- Split out re-usable functionality into JAR projects
 - Create top-level projects with a “pom” target which coordinate the build of subordinate projects
 - Manage your own shared projects using a company repository (Archiva, others)

What we didn't cover...

- Transitive dependencies
 - If one dependency needs version A of a project, but another needs version B... You have to exclude the download of the wrong version
- We didn't cover multiple projects
- We didn't talk much about plugins
- Sites?
- But this will get you started

Resources

- Maven web site: <http://maven.apache.org>

Gradle: Next week



Acknowledgements

Thanks to:

Robert Laramée, Ken Rimple of Chariot Solutions and Julie Greensmith for slides