Maven

Apache Maven is a software project management and comprehension tool.

Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

For these lessons

Maven is a tool for dependency management and building

Dependency management

Open in IDEA the example L1 from

https://github.com/vitaly-chibrikov/harbour_java_2017_05

How to add a new dependency:

- Create a new project managed by Maven
- Open pom.xml
- Add the following

```
<dependencies>
    <dependency>
        <groupId>com.google.guava</groupId>
        <artifactId>guava</artifactId>
        <version>21.0</version>
    </dependency>
</dependencies>
```

Now you can use classes from Guava library in your application

Maven as a builder

Maven can be used as build system (like makefile, ant, gradle…)

Maven can create a jar file for you.

A JAR (Java ARchive) is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file for distribution.

Maven contains a special script language interpreter. And can execute commands from pom.xml

So, pom.xml contains an xml code.

And by adding tags to the pom.xml file you are writing a programm.

Reserved names:

• <name>	name of the project
• <description>	description of the project
• <developers>	list of developers
• <parent>	parent pom if any
• <groupId>	identity of the group of the project
• <artifactId>	identity of the project
• <version>	version

Properties:

- project.
- settings.
- env.
- java.
- user.
- file.

Phases and Goals

A goal represents a specific task which contributes to the building and managing of a project.

Each phase by default has a list of goals to run in.

So, phase is a name for a group of goals.

A Build Lifecycle is Made Up of Phases:

- validate - validate the project is correct and all necessary information is available
- compile - compile the source code of the project
- test - test the compiled source code using a suitable unit testing framework.
- package - take the compiled code and package it in its distributable format, such as a JAR.
- verify - run any checks on results of integration tests to ensure quality criteria are met
- install - install the package into the local repository, for use as a dependency in other projects locally
- deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

Examples

Phases:

```
mvn compile
mvn clean compile
mvn test
mvn package
```

Goals:

```
mvn assembly:assembly
```