



TRANSITION
TECHNOLOGIES

The background is a solid purple color. It features several abstract geometric shapes: a green line with a square node and an arrow pointing towards the title; a pink square with a diagonal line and a blue circle; a pink triangle with diagonal lines; a blue semi-circle; a blue circle with diagonal lines; a blue triangle with diagonal lines; a blue semi-circle with a pink segment; and a blue semi-circle. The title is in white text.

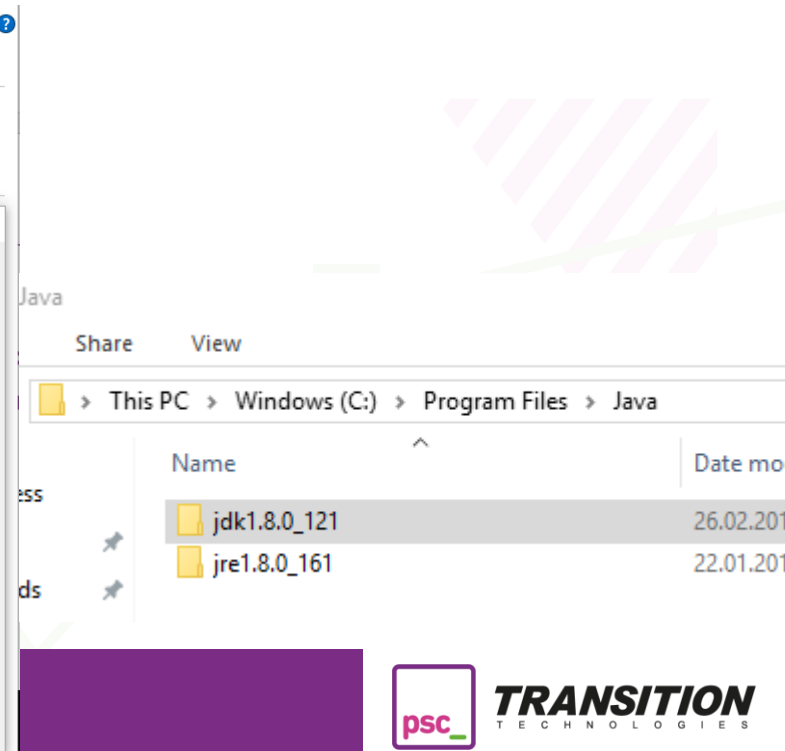
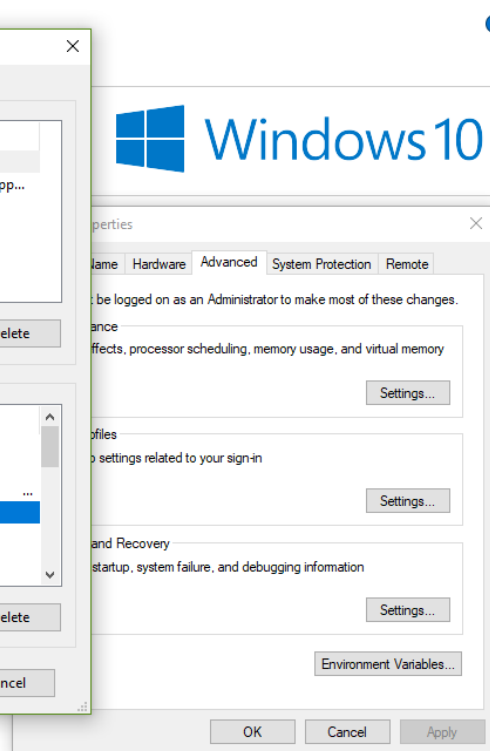
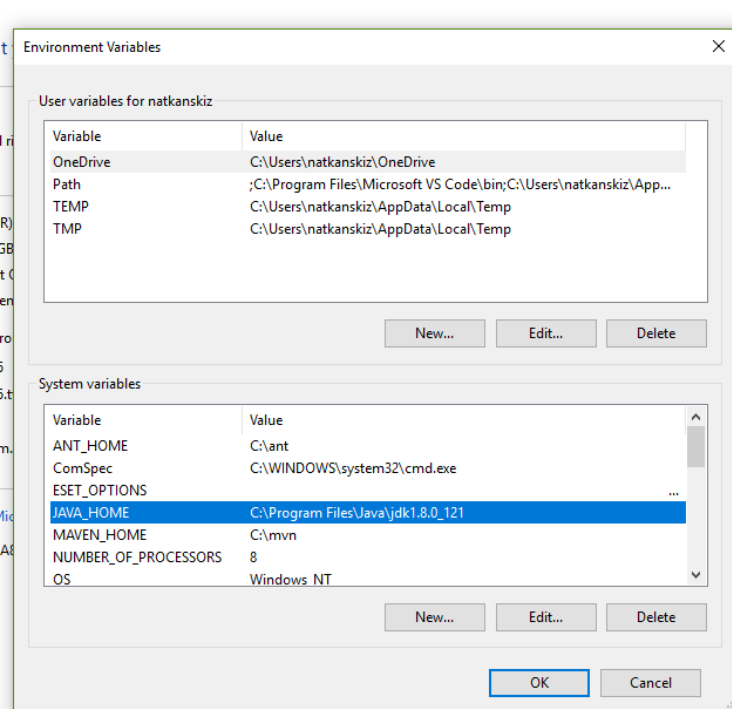
Narzędzia i technologie wspomagające programowanie

Edycja 2018

Introductory Exercise

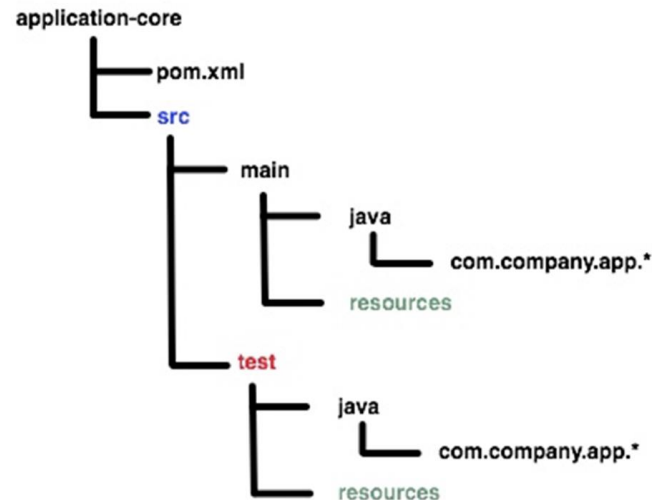
- Install newest Java (JRE + JDK!)
- Set environment variable: **JAVA_HOME** in your operating system
- Using command line or terminal, type: *java -version*

```
C:\Users\natkanskiz>java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
```



Start your development with Maven!

- **Advanced build tool to support the developer at the whole process of a software project**
 - Typical developer tasks: compile, tests, pack into JAR, run
- **Automates creation of the initial folder structure for the Java application**
- **Management of dependencies (no need to manual download of external libraries)**
- **Repository: dependencies can be loaded from the local file system, from the Internet or public repositories**
- **Management of releases**
- **Simple usage: supply project templates (archetypes)**
- **Convention over configuration: avoid as much configuration as possible, by choosing real world default values**
- **Extensible (plugins)**
- **Empower pom.xml**



Scaffolding a project with Maven

- Maven supports project scaffolding, based on project templates called archetype
 - Maven comes with number of „ready-to-go” archetypes
 - Extremely speeds up the development preparation
 - Type: *mvn archetype:generate*
 - A pom should minimum have the following information:
 - **modelVersion**
 - **groupId**
 - **artifactId**
 - **version**
- ```
<project>
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.mycompany.app</groupId>
 <artifactId>my-app</artifactId>
 <version>1</version>
</project>
```

```
mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart
-DgroupId=com.companynam.bank
-DartifactId=consumerBanking
-DarchetypeArtifactId=maven-archetype-quickstart
-DinteractiveMode=false
```

## Maven lifecycles, phases, goals

- 3 built-in build **lifecycles**:
  - **default** - project deployment
  - **clean** - project cleaning
  - **site** - creation of project's documentation
- Each lifecycles is defined by a different list of **phases**, wherein a phase represents a stage in the lifecycle.
- For example, the default lifecycle comprises of the following phases:
  - **validate** - check project correctness
  - **compile**
  - **test** - test the compiled source code using a unit testing
  - **package** - take the compiled code and package into distributable format (JAR/WAR)
  - **verify** - run any checks on results of integration tests to ensure quality criteria are met
  - **install** - install the package into the local repository
  - **deploy** - copies the final package to the remote repository for sharing with other developers and projects.
- **Phase** is made up of plugin **goals**

## Maven Exercise

- Download maven package
- Set environment variable: `MAVEN_HOME`
- Using command line or terminal, check your maven version: `mvn -v` or `mvn --version`
- Generate project with archetype „quickstart” (default)
- Import project into IDE
- Review pom.xml
- Build package using Maven (using command line)
- Add project lombok dependency
- Create simple Java class with lombok annotation
- Build again using Maven

- **Reduce boilerplate code**
- **Generation of constructors, getters/setters, equal and hashCode, toString methods via annotations:**
  - @EqualsAndHashCode
  - @ToString
  - @AllArgsConstructor
  - @Getter
  - @Setter
- **Automatic beans creation**
  - @Data
- **Builder design pattern**
  - @Builder
- **More:** <https://projectlombok.org>

```
<dependency>
 <groupId>org.projectlombok</groupId>
 <artifactId>lombok</artifactId>
 <version>1.16.20</version>
 <scope>provided</scope>
</dependency>
```

## Another Maven's key features that are worth to be known

- **Wrappers**
- **Multi module projects (aggregator)**
- **Profile**
- **Own plugins**
- **Adding goals to life cycle phases**
- **Local/Remote/Central repositories**
- **Read more: <http://maven.apache.org/index.html>**



# Gradle

- uses script file (**build.gradle**)
- handles two things:
  - **project** is made up of different tasks.
  - **task** means a piece of work which a build performs (compile classes, create JAR, generate Javadoc, or publish to repository).
- uses Groovy language for writing scripts (provides a Domain Specific Language (DSL), for describing builds)
- basic predefined tasks:
  - *init*
  - *tasks*
  - *test*
  - *build*
  - *clean*
- learn more: <https://guides.gradle.org/> (tutorials)

- Download Gradle package
- Set environment variable **GRADLE\_HOME**
- Check *gradle -version*
- Create new project: *gradle init --type java-application*
- Import project into your IDE
- Review *build.gradle* file
- Write custom task
- Display all available tasks
- Migrate your Maven project into Gradle
- Run it!

## Continuous Integration

- Merging all developer working copies with a shared mainline several times a day
- Repository stability checks
- Regular releases
- Reduce of costs
- Early error/bug detection



**Travis CI**

Test and deploy with confidence



**CircleCI**

Automatically build, test, and deploy your project in minutes



**AppVeyor**

Cloud service for building, testing and deploying Windows apps



**Percy**

Continuous visual testing and reviews for web apps



**Buddy**

One-click delivery automation for Web Developers



**Semaphore**

Test and deploy at the push of a button



**Cloud 66 Skycap**

Skycap is a container native CI/CD tool



# Jenkins

## Continuous Integration Exercise

- Use Github account and login: <https://travis-ci.com>
- Use synchronize button to download your repositories from GitHub
- Activate repository for which you would like to start *continuous integration*
- Add *.travis.yml* file to your repository
- Run Travis CI for your project

Travis CI

Blog
Status
Help

znpsc

znpsc
Repositories 1
Token:

znpsc
Sync account

We're only showing your public repositories. You can find your private projects on [travis-ci.com](#).

Organizations

You are not currently a member of any organization.

Is an organization missing?  
Review and add your authorized organizations.

1
Click the repository switch on

2
Add .travis.yml file to your repository

3
Trigger your first build with a git push

znpsc/repo1

[illegible]

The screenshot shows the Travis CI interface for a repository named 'znpsc/repo1'. The build is for the 'master' branch. The log shows a successful build with the following steps: 'Commit 88f7a0d', 'Compare e1212b9..88f7a0d', 'Branch master', and 'znpsc authored'. The build status is 'Success'.

[illegible]

```

Worker information
Node system information

Export DESKTOP_PATH=~/Desktop
git clone --recursive --branch master https://github.com/rapid7/rapid7.git
This bin is running on container based infrastructure, which does not allow use of 'sudo', instead use 'sudo' privileges.
If you require sudo, add 'sudo:' required' to your .travis.yml

```

## Free ebooks

---

- Programming Notes for Professionals books
  - <http://goalkicker.com>
- Cheat sheets
  - <https://zeroturnaround.com/rebellabs/reports/>