Objektovo Orientované programovanie

9. Prednáška
LS 2024/2025
Juraj Petrík

Projekt

- Do 27.4.2025 20:00
- Za final max 20b
- Prezentovanie na cviceniach
- Priebezna praca
- Implementacia NEstaci, je potrebne vediet co a preco robite

- Unit testy getterov a setterov – pre reporty

YT Playlist

- Pridane nahravky z pravej strany, snad lepsia viditelnost

Pozvané prednášky

- 13.5.2025 9:00: Java/Kotlin/IntelliJ IDEA/JetBrains Ecosystem (Rober Novotny @ JetBrains)
- 15.5.2025 9:00: Observablity in software development from DEVops perspective (Adam Hamsik & Martin Hauskrech @ Labyrinth Labs)

- Mozno donesu aj nejaky merch ☺

Java reflecion API (java.lang.reflect)

- Skúmanie a modifikovanie programu počas behu (”seba samého”)
- Inšpekcia tried, rozhraní, metód počas behu (runtime)
- Vytváranie nových inštancií tried
- Dynamické volanie metód
- Get and set atributov metod

- Pozor, **ne**robí z Javy dynamický jazyk (nevieme pridávať atribúty, metódy, triedy atď.) a nie vždy podporuje „OOP“

Jadro java.lang.reflect

- Class
- Field
- Method
- Constructor
- Modifier

get() vs getDeclared()

https://docs.oracle.com/javase/tutorial/reflect/class/classMembers.html

Class Methods for Locating Fields

Class API	List of members?	Inherited members?	Private members?
getDeclaredField()	no	no	yes
getField()	no	yes	no
getDeclaredFields()	yes	no	yes
getFields()	yes	yes	no

Class Methods for Locating Methods

Class API	List of members?	Inherited members?	Private members?
getDeclaredMethod()	no	no	yes
getMethod()	no	yes	no
getDeclaredMethods()	yes	no	yes
getMethods()	yes	yes	no

Class Methods for Locating Constructors

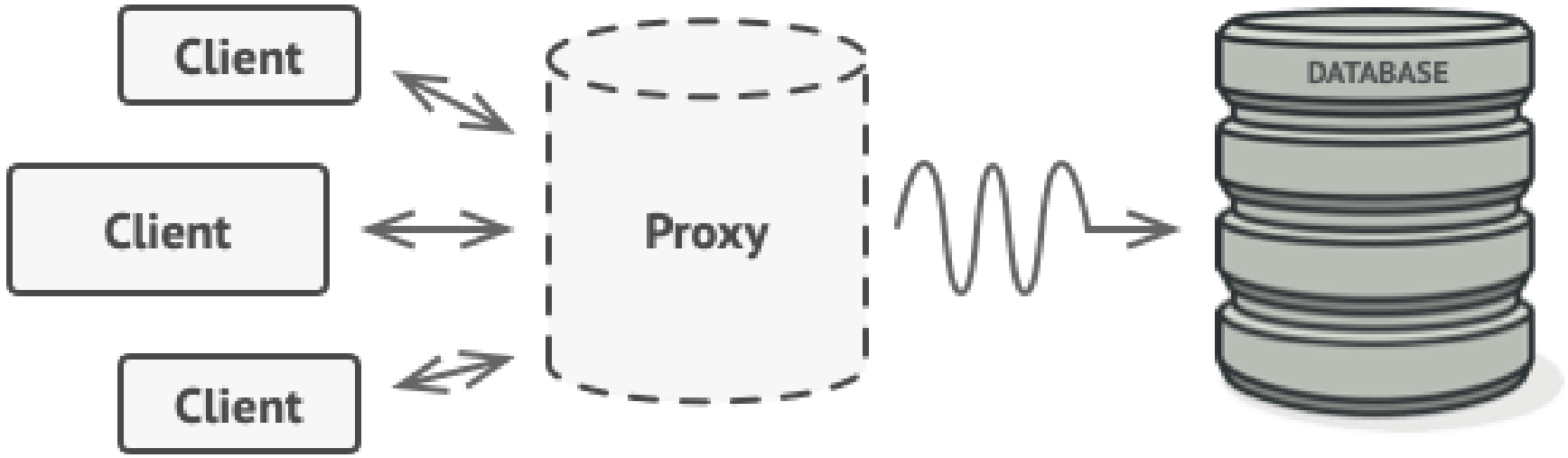
Class API	List of members?	Inherited members?	Private members?
getDeclaredConstructor()	no	N/A[1]	yes
getConstructor()	no	N/A[1]	no
getDeclaredConstructors()	yes	N/A[1]	yes
getConstructors()	yes	N/A[1]	no

[1] Constructors are not inherited.

Methods	getMethods()	getDeclaredMethods
public	✔	✔
protected	✘	✔
private	✘	✔
static public	✔	✔
static protected	✘	✔
static private	✘	✔
default public	✔	✔
default protected	✘	✔
default private	✘	✔
inherited public	✔	✘
inherited protected	✘	✘
inherited private	✘	✘
inherited static private	✔	✘
inherited static protected	✘	✘
inherited static private	✘	✘
default inherited public	✔	✘
default inherited protected	✘	✘
default inherited private	✘	✘

getMethods() vs getDeclaredMethods()

Proxy pattern



- https://refactoring.guru/design-patterns/proxy

Real world usage

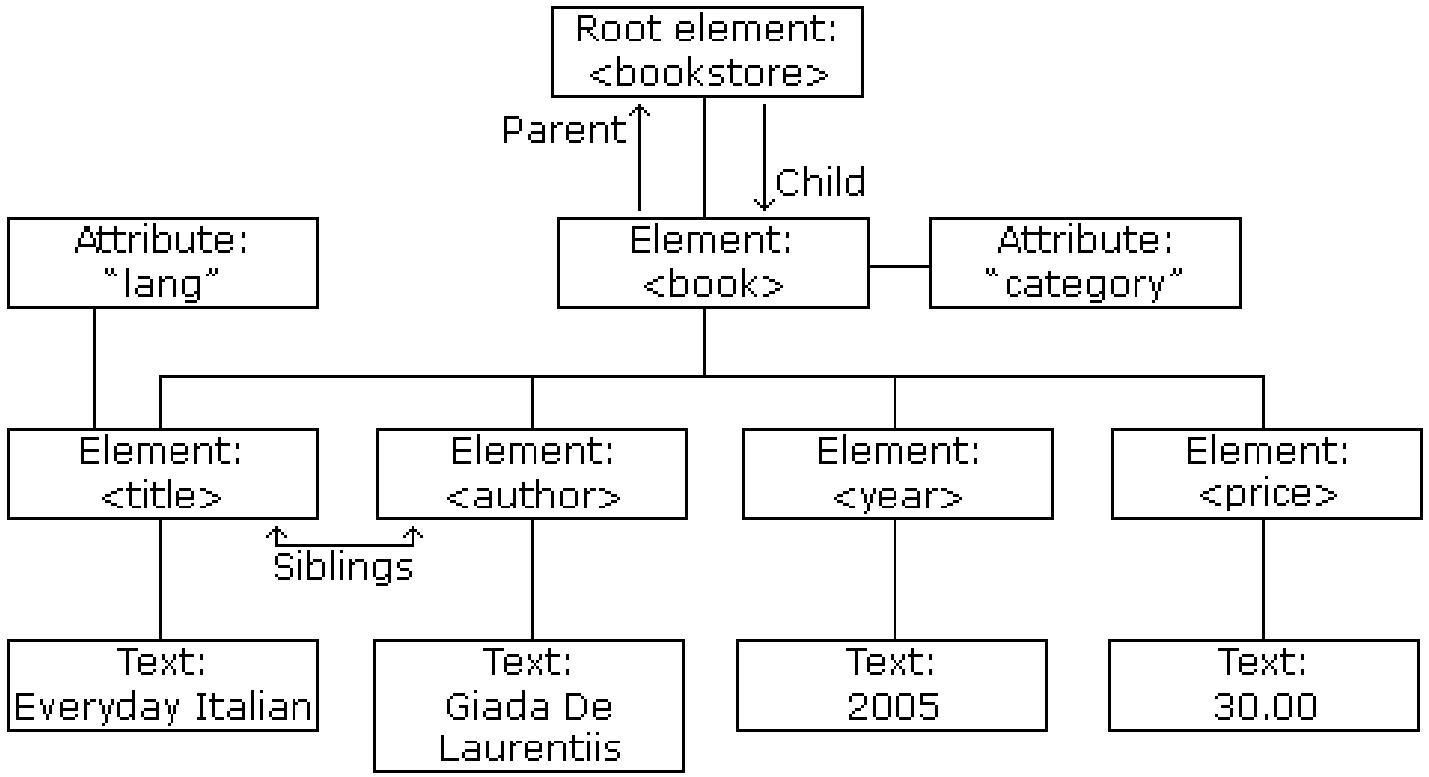
- Spring framework – e.g. proxies (AOP), configs (XML)
- Hibernate – e.g. fields
- JUnit, Mockito – e.g. Runners
- Jackson/GSON – e.g. fields
- JavaFX, Swing

XML

- eXtensible Markup Language
- Konfiguračné súbory (napr. Maven, Spring), prenos dát
- Pomerne ukecané
- Stromová štruktúra:

```
<?xml version="1.0" encoding="UTF-8"?>
  <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
```

XML structure



XML Schema

- Predpis ako má (konkrétne) XML vyzerať:

```
<xs:element name="note">

  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  </xs:element>
```

javax.xml.*

- DOM (Document Object Model) - Tree-based **in-memory** representation
- SAX (Simple API for XML) - **Event-based** streaming parser
- StAX (Streaming API for XML) - **Pull-parser** approach

Maven

- Project management
- (Build)
- (Manage dependencies)

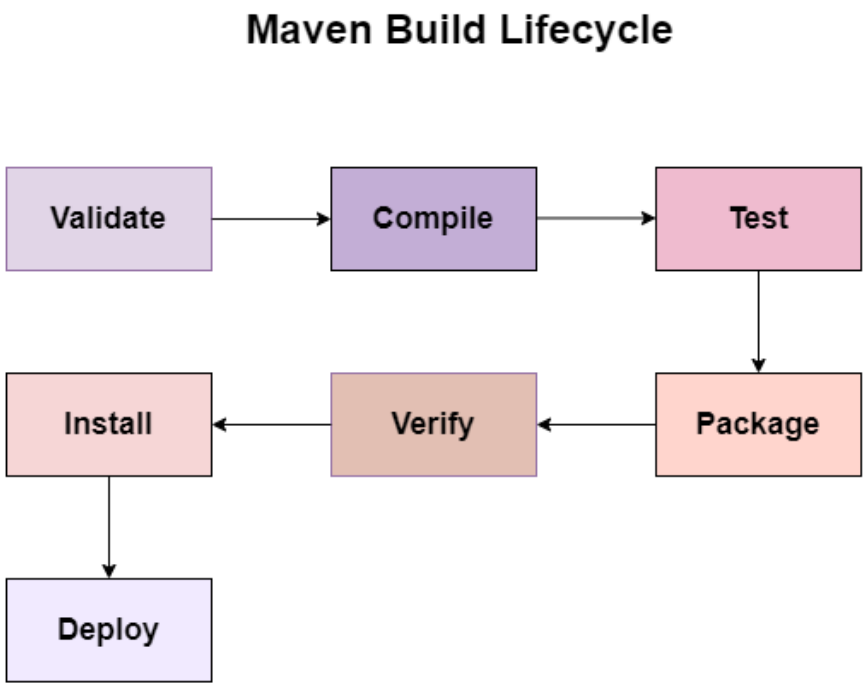
- Standardizovana struktura projektov
- Dependency hell!
- Integracia CI/CD
- Necheme to isté púšťať ručne všetko stále dokola

Maven

- POM (Project Object Model) – pom.xml
- Lifecycle and Phases (clean, compile, test, package, install, deploy)
- Dependency Management
- Repositories (local, central, remote)

Lifecycle

- mvn clean: deletes the target directory.
- mvn compile: compiles source code.
- mvn test: runs unit tests.
- mvn package: creates JAR/WAR file.
- mvn install: installs artifact in local repository.
- mvn dependency:tree: shows dependency hierarchy.



pom.xml

- <project>
- <modelVersion>4.0.0</modelVersion>
- <groupId>com.example</groupId>
- <artifactId>my-app</artifactId>
- <version>1.0.0</version>
-
- <dependencies>
- <dependency>
- <groupId>junit</groupId>
- <artifactId>junit</artifactId>
- <version>4.12</version>
- <scope>test</scope>
- </dependency>
- </dependencies>
- </project>

Quiz time