

# Java SE 9: Exploiting Modularity and Other New Features

Kód kurzu: D99947

Kurz představuje Java module system (Jigsaw) a další novinky v Java SE 9. Mezi nimi jsou JShell, pomocné metody pro práci s kolekce a poli, vylepšení Stream API, nebo třeba detailnější možnosti nastavení anotace `@Deprecated`. Největší novinkou Java SE 9 je bezesporu modularizace JRE, z které benefituje jak samotné běhové prostředí (možnost sestavit si odlehčenou verzi JRE na míru pro svoji desktopovou aplikaci), tak i programátoři knihoven, kteří mohou jasně definovat, které třídy jsou veřejné a které naopak implementační detail, co se může v průběhu času měnit. Je to novinka tak zásadní, že její zvládnutí bude v budoucnu nutnost. Pojdte se s ní seznámit na tento kurz!

## Pro koho je kurz určen

- Kurz je vhodný pro Java programátory, kteří používají starší verzi Javy [5-8] a chtějí se obeznámit s novinkami Java SE 9, například psát modularizované knihovny.

## Co Vás naučíme

- Design applications to take advantage of the module system and its more reliable configuration, improved security and performance, and more easily scalable applications.
- Migrate existing applications to a modular applications in a step-by-step manner, choosing which parts of the application to migrate first.
- Deal with common problems encountered in migrating an application, including, cyclic dependencies and split packages.
- Use services to make modularized applications more robust and easily extensible.
- Create multi-release JAR files that can be run on different Java releases.
- Use convenience methods to reduce code that seems verbose, inefficient or boilerplate, and increase readability.
- Use JShell to quickly run small code experiments and test new APIs.

## Požadované vstupní znalosti

- Znalost Java SE programování na běžné úrovni, generics, lambda výrazy, Stream API.

## Osnova kurzu

### Why Modules?

- Module System
- Levels of a Typical Application
- How Does Java SE 8 Address Maintainability and Reliability?
- Classes, Subclasses, Interfaces
- Class Level Unit of Reuse (Java SE 8)
- Packages
- JARs
- JAR Files and Distribution Issues

### Working with the Module System

- Projects Before Modularization
- module-info.java: Introduction
- Creating a Truly Modular Project
- Compiling Modular JAR Files
- Accessibility Between Classes
- Readability Between Modules
- What Is Readable from the competition Module?
- The Effects of Exporting

### Modular JDK

- Modular Development in JDK 9
- The JDK

#### GOPAS Praha

Kodaňská 1441/46  
101 00 Praha 10  
Tel.: +420 234 064 900-3  
[info@gopas.cz](mailto:info@gopas.cz)

#### GOPAS Brno

Nové sady 996/25  
602 00 Brno  
Tel.: +420 542 422 111  
[info@gopas.cz](mailto:info@gopas.cz)

#### GOPAS Bratislava

Dr. Vladimíra Clementisa 10  
Bratislava, 821 02  
Tel.: +421 248 282 701-2  
[info@gopas.sk](mailto:info@gopas.sk)



Copyright © 2020 GOPAS, a.s.,  
All rights reserved

# Java SE 9: Exploiting Modularity and Other New Features

- The Modular JDK
- Modules in JDK 9
- Java SE Modules
- The Module Graph of Java SE
- The Base Module
- Finding the Right Platform Module

## Creating Custom Runtime Images

- What Is a Custom Runtime Image?
- Link Time
- Using jlink to Create a Custom Runtime Image
- Modules Resolved in a Custom Runtime Image
- Advantages of a Custom Runtime Image
- JIMAGE Format
- Footprint of a Custom Runtime Image
- jlink Resolves Transitive Dependencies

## Migration

- Typical Application
- The League Application
- Run the Application
- The Unnamed Module
- Top-Down Migration
- Dependencies
- Check Dependencies
- Typical Application Modularized

## Services

- Module Dependencies
- Service Relationships
- Expressing Service Relationships
- Using the Service Type in competition
- Choosing a Provider Class
- Module Dependencies and Services
- Designing a Service Type
- TeamGameManager Application with Additional Services

## Multi-release JAR files

- Packaging an Application for Different JDKs
- Packaging an Application for Different JDK Versions
- The Solution: A Multi-Release JAR file
- What Is a Multi-Release JAR File?
- Structure of a JAR File
- Structure of a Multi-Release JAR File
- Search Process in an MRJAR
- Creating a Multi-Release JAR File

## Private Interface Methods

- Private Methods in Interfaces
- Java SE 7 Interfaces
- Implementing Java SE 7 Interface Methods
- Implementing Methods in Interfaces

### GOPAS Praha

Kodaňská 1441/46  
101 00 Praha 10  
Tel.: +420 234 064 900-3  
[info@gopas.cz](mailto:info@gopas.cz)

### GOPAS Brno

Nové sady 996/25  
602 00 Brno  
Tel.: +420 542 422 111  
[info@gopas.cz](mailto:info@gopas.cz)

### GOPAS Bratislava

Dr. Vladimíra Clementisa 10  
Bratislava, 821 02  
Tel.: +421 248 282 701-2  
[info@gopas.sk](mailto:info@gopas.sk)



Copyright © 2020 GOPAS, a.s.,  
All rights reserved

# Java SE 9: Exploiting Modularity and Other New Features

- What About the Problems of Multiple Inheritance?
- Inheritance Rules of default Methods
- Interfaces Don't Replace Abstract Classes

## Enhancements to the Stream API

- One More Benefit of Default Methods
- Changing a Java Interface
- Why Enhance the Stream API?
- An Ordered List
- takeWhile Provides a Solution
- The Effects and Benefits of takeWhile
- An Unordered List
- filter vs takeWhile

## JShell

- Has This Happened to You?
- A Million Test Classes and Main Methods
- JShell Provides a Solution
- Comparing Normal Execution with REPL
- Getting Started with JShell and REPL
- Scratch Variables
- Declaring Traditional Variables
- Code Snippets

## Convenience Methods for Collections

- What Are Convenience Methods?
- Many Convenience Methods in Java SE 9
- Key Collections Interfaces
- Overloading the of Convenience Method
- Why Overload the of Method?
- Growing a Collection
- ofEntries() Method for Maps
- Immutability

## Convenience Methods for Arrays

- Arrays
- Modeling DNA Strands
- Working with DNA Strands
- Working with DNA Strands by Using a for Loop
- Convenience Methods in the Arrays Class
- Equating DNA Strands
- DNA Subsequences
- Equating Subsequences of DNA

## Enhanced Deprecations for APIs

- What Is Deprecation?
- What Is Enhanced Deprecation?
- How Do You Deprecate an API?
- Using @deprecated
- Enhancements to the @Deprecated Annotation in JDK 9
- Using the @Deprecated Annotation
- Notifications and Warnings

### GOPAS Praha

Kodaňská 1441/46  
101 00 Praha 10  
Tel.: +420 234 064 900-3  
[info@gopas.cz](mailto:info@gopas.cz)

### GOPAS Brno

Nové sady 996/25  
602 00 Brno  
Tel.: +420 542 422 111  
[info@gopas.cz](mailto:info@gopas.cz)

### GOPAS Bratislava

Dr. Vladimíra Clementisa 10  
Bratislava, 821 02  
Tel.: +421 248 282 701-2  
[info@gopas.sk](mailto:info@gopas.sk)



Copyright © 2020 GOPAS, a.s.,  
All rights reserved

# Java SE 9: Exploiting Modularity and Other New Features

- Compiler Deprecation Warnings

## **GOPAS Praha**

Kodaňská 1441/46  
101 00 Praha 10  
Tel.: +420 234 064 900-3  
[info@gopas.cz](mailto:info@gopas.cz)

## **GOPAS Brno**

Nové sady 996/25  
602 00 Brno  
Tel.: +420 542 422 111  
[info@gopas.cz](mailto:info@gopas.cz)

## **GOPAS Bratislava**

Dr. Vladimíra Clementisa 10  
Bratislava, 821 02  
Tel.: +421 248 282 701-2  
[info@gopas.sk](mailto:info@gopas.sk)



Copyright © 2020 GOPAS, a.s.,  
All rights reserved