Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner PhD.

Učebnica:
https://oop.wagjo.com/

OPG
Teória 1

1. Digitálny asistenti
2. Objektovo orientované programovanie
3. Jazyk Java

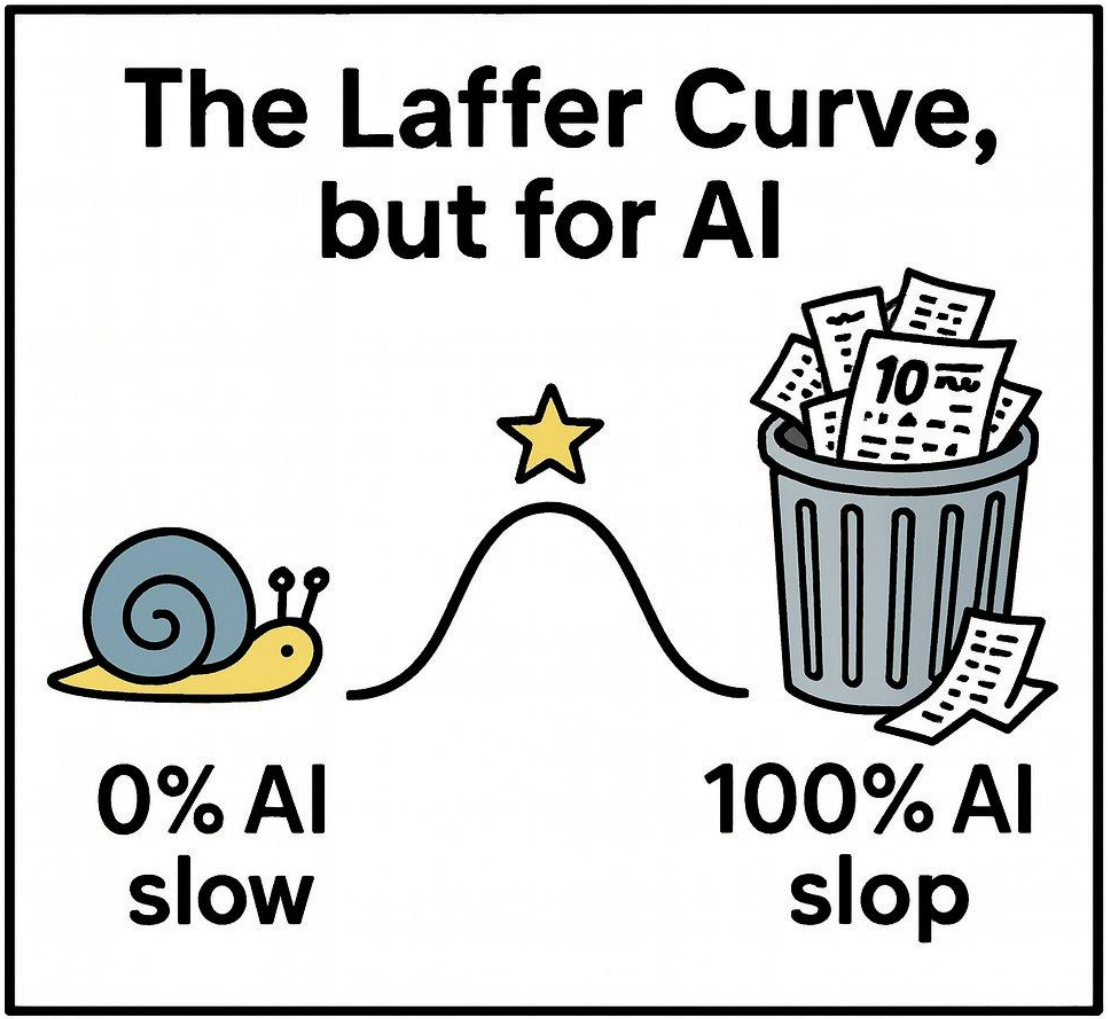
Digitálny Asistenti

AI Asistent: reaktívny, odpovedá na otázky, jednoduché úlohy

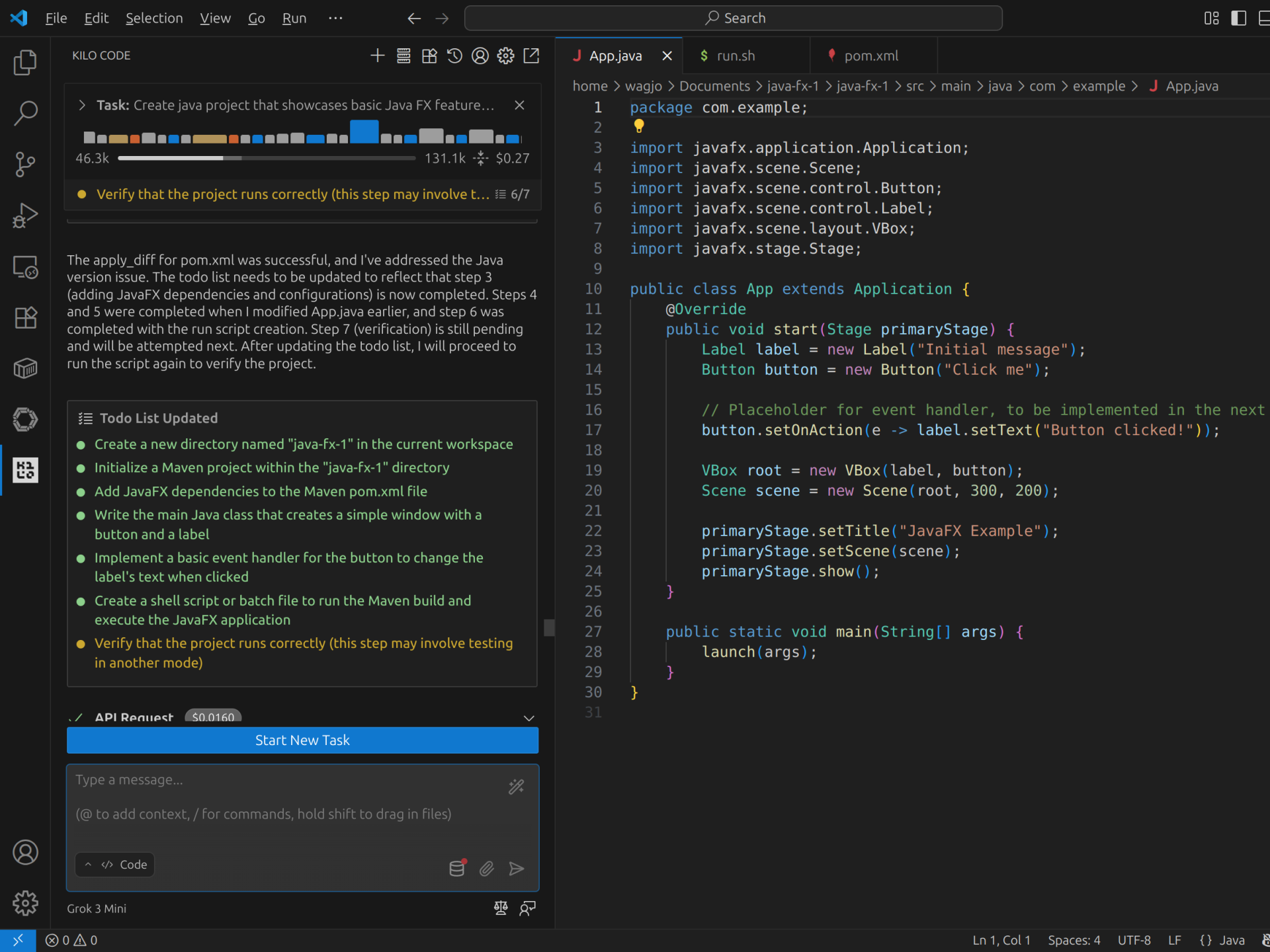
AI Agent: proaktívny, autonómny, zložité viackrokové úlohy

AI Prompting: vytváranie inštrukcií pre AI

AI Verifikácia: kontrola presnosti, spoľahlovosti a kvality výstupu. Je vždy riadená človekom



AI Slop - odpad, vznikne ak AI slepo dôverujem


File Edit Selection View Go Run ···
Search
KILO CODE
Task: Create java project that showcases basic Java FX feature...
46.3k
131.1k
$0.27
Verify that the project runs correctly (this step may involve t... 6/7
The apply_diff for pom.xml was successful, and I've addressed the Java version issue. The todo list needs to be updated to reflect that step 3 (adding JavaFX dependencies and configurations) is now completed. Steps 4 and 5 were completed when I modified App.java earlier, and step 6 was completed with the run script creation. Step 7 (verification) is still pending and will be attempted next. After updating the todo list, I will proceed to run the script again to verify the project.
Todo List Updated
Create a new directory named "java-fx-1" in the current workspace
Initialize a Maven project within the "java-fx-1" directory
Add JavaFX dependencies to the Maven pom.xml file
Write the main Java class that creates a simple window with a button and a label
Implement a basic event handler for the button to change the label's text when clicked
Create a shell script or batch file to run the Maven build and execute the JavaFX application
Verify that the project runs correctly (this step may involve testing in another mode)
API Request $0.0160
Start New Task
Type a message...
(@ to add context, / for commands, hold shift to drag in files)
Code
Grok 3 Mini
App.java
run.sh
pom.xml
home > wagjo > Documents > java-fx-1 > java-fx-1 > src > main > java > com > example > App.java
1 package com.example;
2
3 import javafx.application.Application;
4 import javafx.scene.Scene;
5 import javafx.scene.control.Button;
6 import javafx.scene.control.Label;
7 import javafx.scene.layout.VBox;
8 import javafx.stage.Stage;
9
10 public class App extends Application {
11 @Override
12 public void start(Stage primaryStage) {
13 Label label = new Label("Initial message");
14 Button button = new Button("Click me");
15
16 // Placeholder for event handler, to be implemented in the next
17 button.setOnAction(e -> label.setText("Button clicked!"));
18
19 VBox root = new VBox(label, button);
20 Scene scene = new Scene(root, 300, 200);
21
22 primaryStage.setTitle("JavaFX Example");
23 primaryStage.setScene(scene);
24 primaryStage.show();
25 }
26
27 public static void main(String[] args) {
28 launch(args);
29 }
30 }
31
0 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Java

Objektovo orientované programovanie

Paradigmy v programovaní:

- *Štruktúrované:* sekvencie príkazov, podmienky, cykly
- *Procedurálne:* podprogramy, ktoré sa vedia navzájom volať
- *Objektovo orientované:* objekty, zapuzdrenie, polymorfizmus
- *Funkcionálne:* anonymné funkcie, high-order funkcie

Objektovo orientované programovanie

OOP:

- modeluje podľa nášho vnímania sveta
- zvláda väčšiu komplexnosť
- umožňuje znovupoužitie
- intuitívna organizácia kódu

Jazyk Java

Vlastnosti:

- objektovo orientovaný
- univerzálny
- platformovo nezávislý
- staticko typovaný: typy premenných sú kontrolované pri kompilácii

Jazyk Java

Použitie:

- veľké podnikové systémy
- webový back-end
- Android aplikácie *(jazyk Kotlin)*
- cloudové a distrubuované systémy

Jazyk Java

Platformová nezávislosť:

- kód sa kompiluje do bytecode a beží nad JVM
- motto Javy: Write Once, Run Anywhere *(WORA)*
- virtuálny stroj JVM *(Java Virtual Machine)*
- Java API: vždy dostupná sada knižníc na bežné použitie