KODO ANALIZĖ

Šis Java klasės kodas yra grafine vartotojo sąsaja (GUI), skirta įvesti ir įrašyti mokėjimo duomenis į duomenų bazę naudojant JDBC (Java Database Connectivity).

Kodas	**Analizė**
package org.example.utils; import javax.swing.*; import java.awt.*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.SQLException;	**Paketų importavimas ir klasių aprašymas:** Įkeliama būtina Swing biblioteka (javax.swing.*), AWT biblioteka (java.awt.*) ir kitos reikalingos klasės, pvz., ActionEvent ir ActionListener skirtos veiksmams su mygtukais.

public class ApmokejimasForma extends JPanel {
  private JTextField sumaField;
  private JTextField mokejimoBudasField;
  private JTextField asmuoIdField;
  private JTextField kursaiIdField;
  public ApmokejimasForma() {
  setupForm();
  }
	ApmokejimasForma klasės aprašas:
	ApmokejimasForma klasė paveldi JPanel klasę, kurioje yra laukai (JTextField), skirti įvesti duomenis apie sumą, mokėjimo būdą, asmenį ir kursus.

private void setupForm() {
  setLayout(new GridLayout(5, 2));
  JLabel sumaLabel = new JLabel("Suma:");
  add(sumaLabel);
  sumaField = new JTextField(20);
  add(sumaField);
  JLabel mokejimoBudasLabel = new JLabel("Mokėjimo būdas:");
  add(mokejimoBudasLabel);
  mokejimoBudasField = new JTextField(20);
  add(mokejimoBudasField);
	setupForm() metodas:
	setupForm() metodas nustato formos išdėstymą (GridLayout) ir prideda etiketes, teksto laukus (JTextField) bei mygtukus (JButton) "Išsaugoti" ir "Išvalyti".
	saveButton mygtuko ActionListener iškviečia saveApmokejimas() metodą, o clearButton mygtuko ActionListener iškviečia clearForm() metodą.

JLabel asmuoIdLabel = new JLabel("Asmuo ID:");
add(asmuoIdLabel);
asmuoIdField = new JTextField(20);
add(asmuoIdField);
JLabel kursaiIdLabel = new JLabel("Kursai ID:");
add(kursaiIdLabel);
kursaiIdField = new JTextField(20);
add(kursaiIdField);
JButton saveButton = new JButton("Išsaugoti");
saveButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
saveApmokejimas();
}
});
add(saveButton);
JButton clearButton = new JButton("Išvalyti");
clearButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
clearForm();
}
});
add(clearButton);
}	

private void saveApmokejimas() {
try {
String sumaText = sumaField.getText().trim();
double suma = Double.parseDouble(sumaText);
String mokejimoBudas = mokejimoBudasField.getText().trim();
String asmuoIdText = asmuoIdField.getText().trim();
int asmuoId = Integer.parseInt(asmuoIdText);
String kursaiIdText = kursaiIdField.getText().trim();
	saveApmokejimas() metodas:
	saveApmokejimas() metodas išsaugo įvestus duomenis į duomenų bazę.
	Įvedami duomenys gaunami iš teksto laukų (JTextField).
	Sukuriamas JDBC ryšys su MySQL duomenų baze.
	Paruošiama SQL užklausa, naudojant PreparedStatement, ir įdedami parametrai.
	executeUpdate() metodas išvykdo užklausą.
	Jei operacija pavyksta, parodomas pranešimas JOptionPane ir išvaloma

int kursaiId = Integer.parseInt(kursaiIdText);
String url = "jdbc:mysql://localhost:3306/baigiamasis";
String user = "root";
String password = "jakuliene";
Connection connection = DriverManager.getConnection(url, user, password);
String sql = "INSERT INTO apmokejimas (Suma, Mokejimo_budas, Asmuo_ID, Kursai_ID) VALUES (?, ?, ?, ?)";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setDouble(1, suma);
statement.setString(2, mokejimoBudas);
statement.setInt(3, asmuoId);
statement.setInt(4, kursaiId);
int rowsInserted = statement.executeUpdate();
if (rowsInserted > 0) {
JOptionPane.showMessageDialog(this, "Įrašas išsaugotas sėkmingai.");
clearForm();
}
connection.close();
} catch (NumberFormatException | SQLException ex) {
JOptionPane.showMessageDialog(this, "Klaida įrašant duomenis: " + ex.getMessage(), "Klaida", JOptionPane.ERROR_MESSAGE);
}
}
	forma naudojant clearForm() metodą.
	Įvykus klaidai, rodomas klaidos pranešimas.

private void clearForm() {
sumaField.setText("");
mokejimoBudasField.setText("");
asmuoIdField.setText("");
kursaiIdField.setText("");
}
	clearForm() metodas:
	clearForm() metodas išvalo visus įvestus duomenis teksto laukuose.

public static void main(String[] args) {	**main() metodas:**

SwingUtilities.invokeLater(new Runnable() {
@Override
public void run() {
JFrame frame = new JFrame("Įrašų Forma");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.getContentPane().add(new ApmokejimasForma(),
BorderLayout.CENTER);
frame.pack();
frame.setVisible(true);
}
});
}
}
	main() metodas pradeda programą sukurdamas ir rodydamas pagrindinį JFrame langą su "Įrašų Forma" pavadinimu. SwingUtilities.invokeLater() užtikrina, kad Swing komponentai būtų sukurti ir rodomi teisingame nišoje.

SwaggerConfig

Šis kodas aprašo Swagger konfigūraciją Spring Boot projekte. Pagrindinis tikslas yra sukonfigūruoti Swagger dokumentacijos generavimą REST API endpoint'ams

Kodas	Analizė
package org.example.config; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import springfox.documentation.builders.PathSelectors; import springfox.documentation.builders.RequestHandlerSelectors; import springfox.documentation.service.ApiInfo; import springfox.documentation.service.Contact; import springfox.documentation.spi.DocumentationType; import springfox.documentation.spring.web.plugins.Docket; import springfox.documentation.swagger2.annotations.EnableSwagger2; import java.util.Collections;	**Paketų ir klasių importavimas:** Šios importų eilutės įtraukia reikalingas Spring ir Swagger klases bei anotacijas, kurios bus naudojamos konfigūruojant Swagger.
@Configuration @EnableSwagger2	**Klasių anotacijos:** @Configuration: Nurodo, kad ši klasė yra Spring konfigūracijos klasė. @EnableSwagger2: Įjungia Swagger 2 funkcionalumą šiame Spring Boot projekte.

public class SwaggerConfig {
@Bean
public Docket api() {
return new Docket(DocumentationType.SWAGGER_2)
.select()
 	
.apis(RequestHandlerSelectors.basePackage("org.example.controller"))
.paths(PathSelectors.any())
	Docket Bean apibrėžimas:
	@Bean: Spring konteineris šią metodą pavers Bean, kurį bus galima injektuoti į kitus komponentus.
	Docket: Pagrindinis Swagger API konfigūracijos komponentas.
	.select(): Pradeda ApiSelectorBuilder konstravimą.
	.apis(RequestHandlerSelectors.basePackage("org.example.controller")): Nurodo, kad Swagger turėtų nuskaityti tik org.example.controller paketo klases.

.build() .apiInfo(apiInfo()); }	.paths(PathSelectors.any()): Leidžia visus URL kelius. .build(): Sukonstruoja Docket objektą. .apiInfo(apiInfo()): Prideda papildomą informaciją apie API.

private ApiInfo apiInfo() {
return new ApiInfo(
"Course Registration API",
"API for Course Registration System.",
"1.0",
"Terms of service",
new Contact("Your Name", "www.example.com",
"your-email@example.com"),
"License of API",
"API license URL",
Collections.emptyList());
}
}
	Papildomos API informacijos apibrėžimas:
	ApiInfo: Sukuria objektą su papildoma informacija apie API.
	Parametrai apima:
	Pavadinimas ("Course Registration API").
	Aprašymas ("API for Course Registration System.").
	Versija ("1.0").
	Naudojimo sąlygos ("Terms of service").
	Kontaktinė informacija (new Contact("Your Name",
	"www.example.com", "your-email@example.com")).
	API licencija ("License of API").
	Licencijos URL ("API license URL").
	Vendorų plėtiniai (Collections.emptyList()).

Šis SwaggerConfig klasės kodas konfigūruoja Swagger 2 generavimą API dokumentacijai. Jis nuskaito visus kontrolerius iš org.example.controller paketo, leidžia visus URL kelius ir prideda papildomą informaciją apie API, kuri bus rodoma Swagger UI.