🧩 Einführung in MVC (Model-View-Controller)

📖 Was ist MVC?

MVC (Model-View-Controller) ist ein Architekturprinzip zur Trennung von Logik, Daten und Benutzeroberfläche.

- **Model** – verwaltet die Daten und Geschäftslogik
- **View** – zeigt die Benutzeroberfläche an und empfängt Eingaben
- **Controller** – vermittelt zwischen Model und View, verarbeitet Aktionen

---

☑️ Vorteile von MVC

- Trennung von Verantwortlichkeiten
- Erleichtert Wartung und Tests
- Mehrfachverwendung von Komponenten (z. B. unterschiedliche Views für dasselbe Model)
- Klare Struktur für größere Anwendungen

💡 Anwendungsbeispiele für MVC

- Desktop-GUIs (z. B. Java Swing / JavaFX)
- Web-Anwendungen (Spring MVC, ASP.NET MVC)
- Mobile Apps (Android MVC/MVVM)
- Spieleentwicklung (z. B. Spiellogik vs. Darstellung)

💻 Beispiel: Einfaches Formular mit Name & E-Mail

Projektstruktur

```
└── src/formularmvc/
    ├── UserModel.java       // Model: enthält Name und E-Mail
    ├── UserView.java        // View: GUI-Elemente (Textfelder, Button)
    ├── UserController.java  // Controller: verarbeitet Klick und meldet zurück
    └── Main.java            // Einstiegspunkt
```

1️⃣ Model

```
package formularmvc;

/**
 * Model-Klasse für Benutzerdaten (Name und E-Mail).
 */
```

```
public class UserModel {
    private String name;
    private String email;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

2 View

```
package formularmvc;

import javax.swing.*;
import java.awt.*;

/**
 * View-Klasse für das Formular.
 * Stellt Textfelder für Name und E-Mail sowie einen Button dar.
 */
public class UserView extends JFrame {
    private JTextField nameField = new JTextField(20);
    private JTextField emailField = new JTextField(20);
    private JButton submitButton = new JButton("Abschicken");
    private JLabel messageLabel = new JLabel("");

    public UserView() {
        this.setTitle("Benutzerdaten Formular (MVC)");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(350, 200);
        this.setLayout(new GridLayout(4, 2));

        this.add(new JLabel("Name:"));
        this.add(nameField);
        this.add(new JLabel("E-Mail:"));
        this.add(emailField);
        this.add(submitButton);
        this.add(messageLabel);
    }
```

```
    public JTextField getNameField() {
        return nameField;
    }

    public JTextField getEmailField() {
        return emailField;
    }

    public JButton getSubmitButton() {
        return submitButton;
    }

    public void showMessage(String message) {
        messageLabel.setText(message);
    }
}
```

3 Controller

```
package formularmvc;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Controller-Klasse zur Verarbeitung der Formulardaten.
 */
public class UserController {
    private UserModel model;
    private UserView view;

    public UserController(UserModel model, UserView view) {
        this.model = model;
        this.view = view;

        this.view.getSubmitButton().addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                model.setName(view.getNameField().getText());
                model.setEmail(view.getEmailField().getText());
                view.showMessage("Gespeichert: " + model.getName() + " (" +
model.getEmail() + ")");
            }
        });
    }
}
```

4 Main

```java
package formularmvc;

/**
 * Einstiegspunkt der Formular-Anwendung.
 */
public class Main {
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(() -> {
            UserModel model = new UserModel();
            UserView view = new UserView();
            new UserController(model, view);
            view.setVisible(true);
        });
    }
}
```

📌 Zusammenfassung

Dieses Beispiel zeigt eine **einfache Umsetzung von MVC mit Java Swing**:

- **Model**: hält die Daten (Name, E-Mail)
- **View**: zeigt die Eingabemaske
- **Controller**: verarbeitet Klicks und aktualisiert die View

Ideal für Lernzwecke oder als Grundlage für komplexere Formulare.