Objektovo orientované programovanie

Učiteľ:
Ing. Jozef Wagner, PhD.

Učebnica:
https://oop.wagjo.com/

OPG
Teória 26

1. Model-View-Controller
2. Ciele a vlastnosti
3. MVC v JavaFX

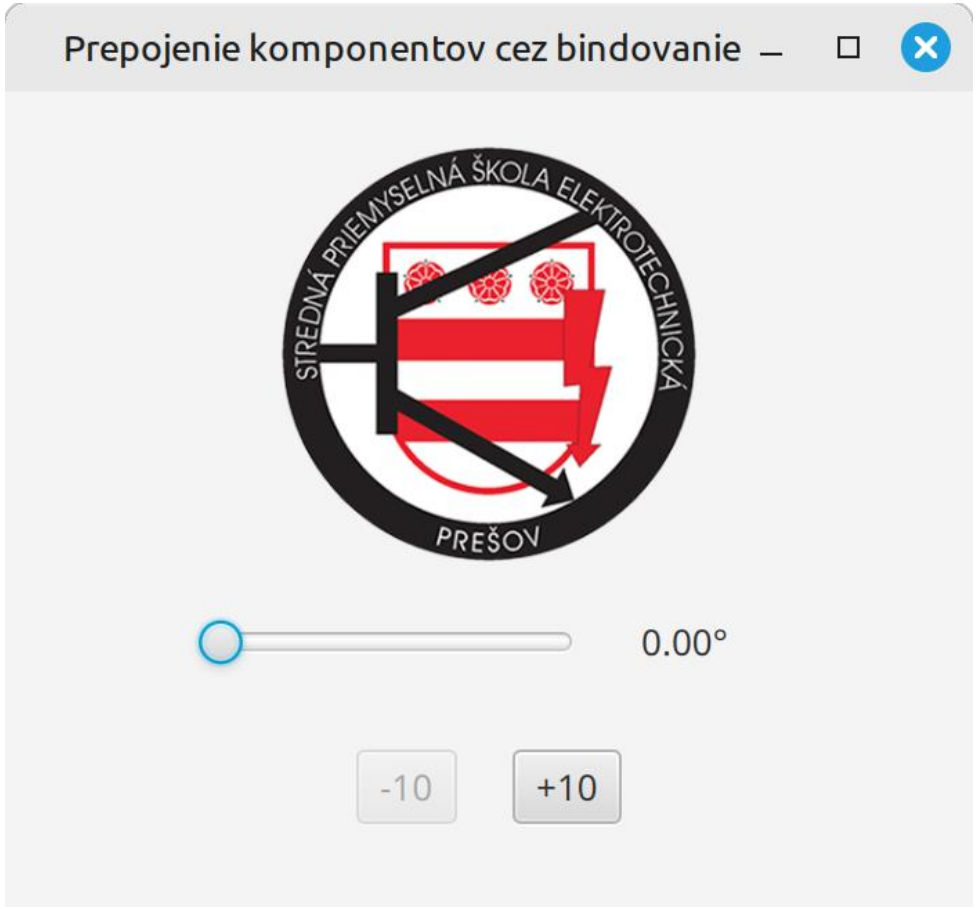
Opakovanie

Ovládacie prvky sú prepojené pomocou property objektov a `bind`

Všetky ovládacie prvky sú prepojené so sliderom

Hlavná hodnota – uhol natočenia – je uložená v slideri

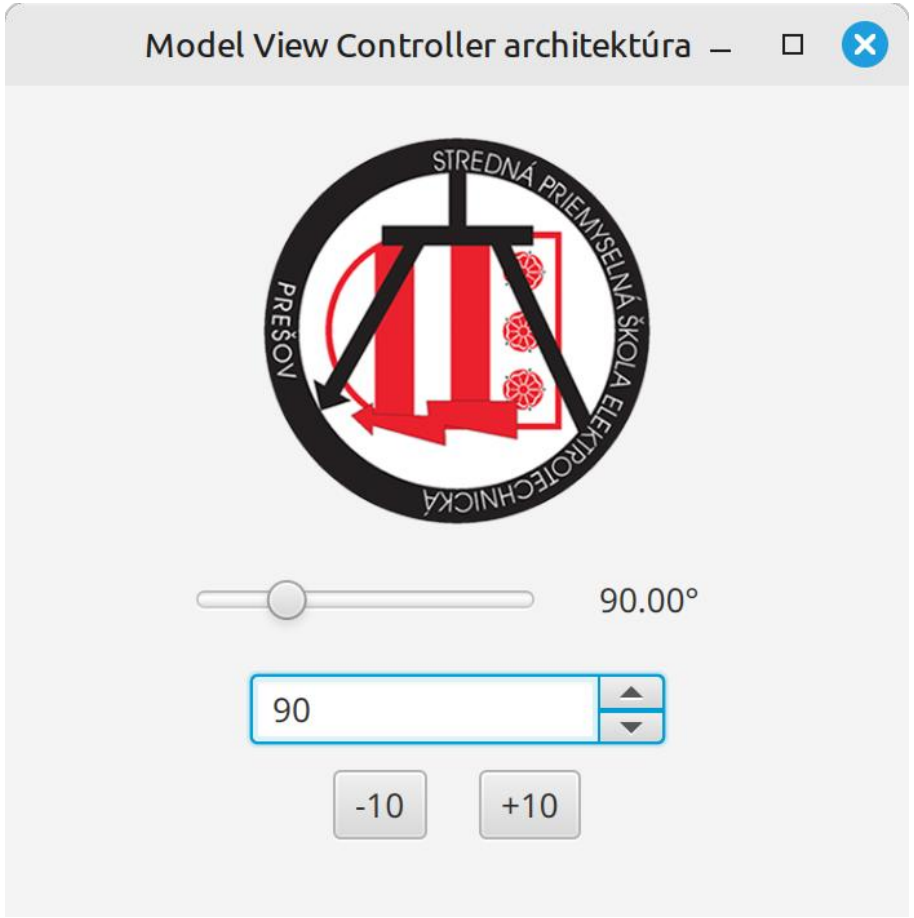
Z obyčajného komponentu sme urobili veľmi dôležitý prvok, držiaci aplikáciu pokope



Problém

Pridáme do programu spinner – kto má teraz mať na starosti uhol natočenia?

Dôležité dáta a hodnoty, ktoré majú význam naprieč aplikáciou, by mali byť umiestnené samostatne.



Model-View-Controller

MVC definuje tri spolupracujúce vrstvy.

Umožňuje **oddelenie zodpovedností** (separation of concerns).

Ciele MVC

- ľahšie testovanie, udržiavanie a rozšíriteľnosť
- lepšia spolupráva v rámci tímu
- znížiť prelínanie (coupling) medzi vrstvami

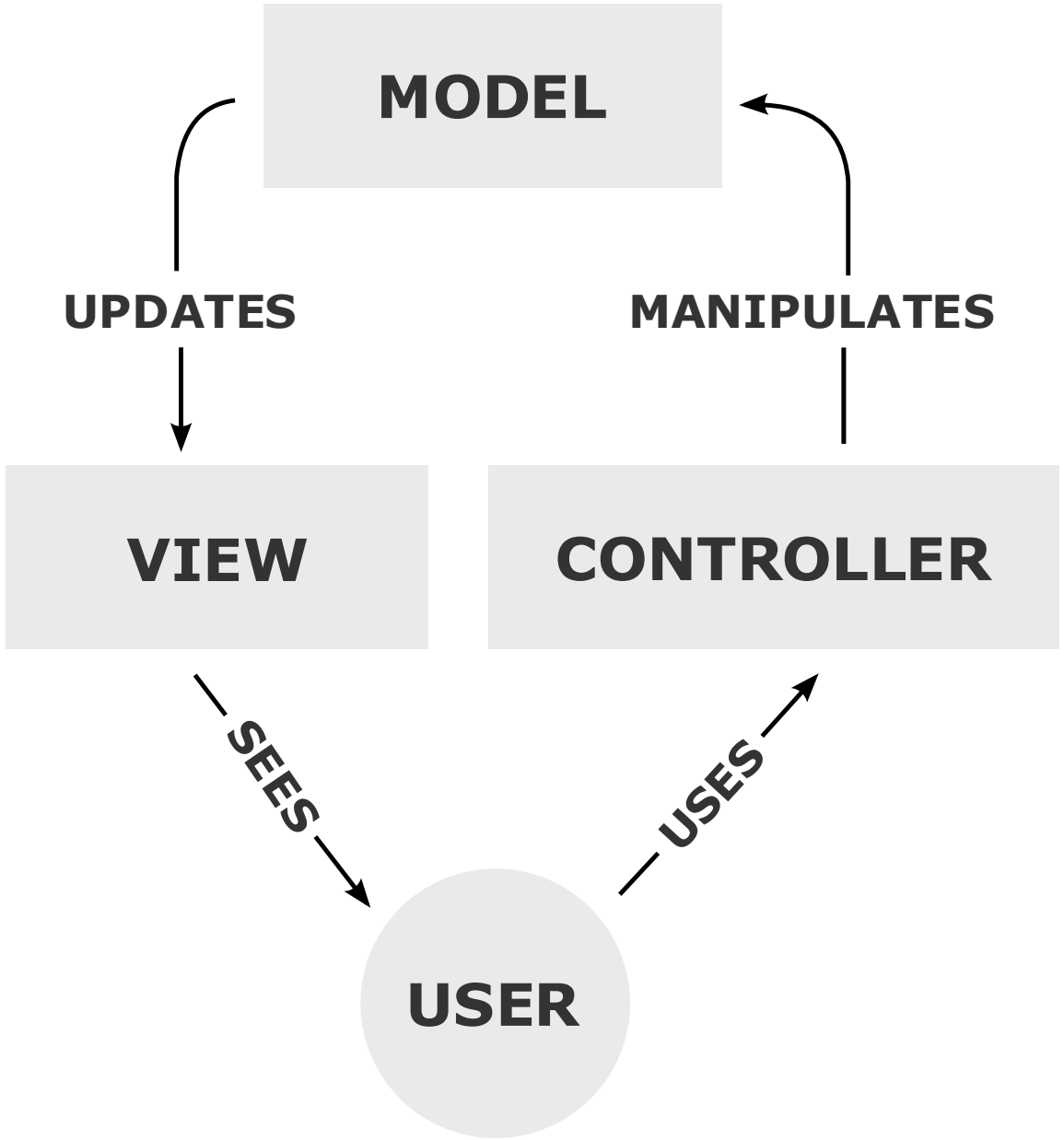
Model-View-Controller

Model - dáta a biznis logiku aplikácie (validácia hodnôt, výpočty, komunikácia s databázou)

View - užívateľské rozhranie (UI), zobrazuje hodnoty z Modelu. Je to hlavne FXML.

Controller - spracovanie udalostí (eventov), aktualizácia Modelu, refreshovanie View

MVC



Fungovanie MVC

1. Užívateľ vykoná akciu vo View (napr. klik myšou).
2. Controller zachytí udalosť.
4. Controller zavolá metódy na Modeli (napr. model.saveUser()).
5. Model vykoná logiku a zmení svoj stav.
6. View sa aktualizuje s novými hodnotami Modelu.

Vlastnosti MVC

Pri malých aplikáciách je MVC príliš komplikovaný

Možnosť testovať model bez UI

Opätovná použiteľnosť (tviacero View: desktop, web, mobil).

Jednoduchšia údržba (zmena UI neovplyvní biznis logiku).

```
public record Model (DoubleProperty rotacia) {

        public static Model createModel() {
            return new Model(new SimpleDoubleProperty(0.0));
        }

        public void rotuj(double delta) {
            double hodnota = rotacia.getValue();
            hodnota += delta;
            hodnota = Math.max(hodnota, 0);
            hodnota = Math.min(hodnota, 360);
            rotacia.setValue(hodnota);
        }
    }
```

```
public class Controller {
  Model model;

  public ImageView obrazok;
  public Slider slider;
  public Label label;
  public Button mButton;

  public void initialize() {
    model = Model.createModel();
    obrazok.rotateProperty().bind(model.rotacia());
    label.textProperty().bind(Bindings.format("%.2f°", model.rotacia()));
    slider.valueProperty().bindBidirectional(model.rotacia());
    mButton.disableProperty().bind(model.rotacia().lessThanOrEqualTo(0));
  }

  public void minus10(ActionEvent actionEvent) {
    model.rotuj(-10);
  }
}
```