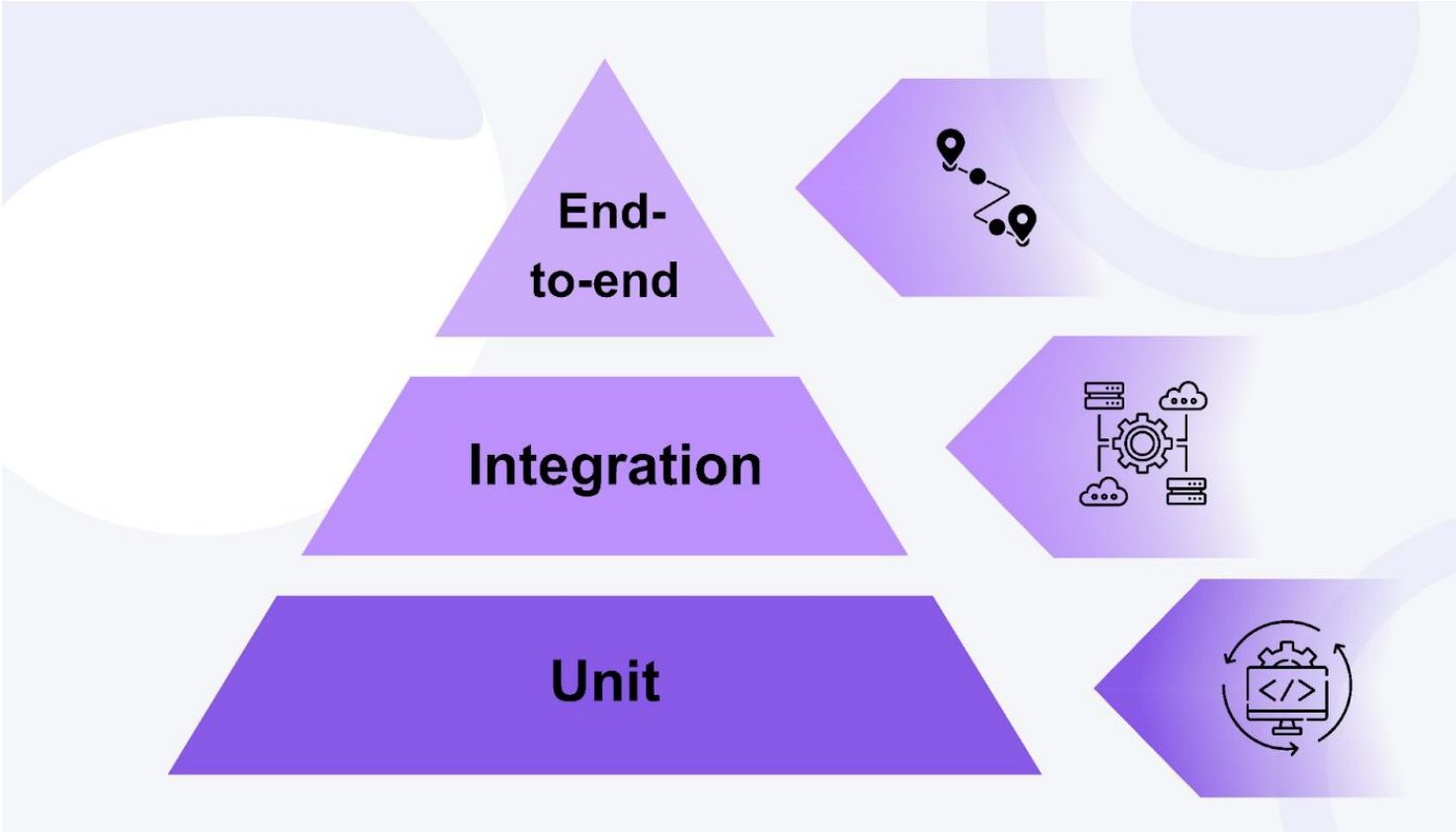
# Object-Oriented Programming

5. Lecture
LS 2025/2026
Author: Juraj Petrík

# Testing

- Unit
- Integration
- Functional
- Acceptance
- Performance
- E2E
- Security



# Unit testing (JUnit)

- We test units (the smallest functional units)
- An integral part of development – written by the programmer who implements the given functionality
- The code must be written in such a way that it can be tested
- E.g. with TDD (test-driven development), I write the tests first and then implement them
- We test logic, edge cases, exception handling, etc.
- Assert once

# Basic principles of unit testing

- Tests should be simple
- Test behavior
- AAA – Arrange, Act, Assert
- Tests are deterministic, isolated, independent
- Data should be as close to reality as possible

- 100% line coverage vs. e.g. branch coverage
- Test setters and getters?

# Unit testing

- https://junit.org/junit5/docs/current/user-guide/
- https://github.com/DiUS/java-faker
- https://site.mockito.org/
- https://github.com/Jur1cek/jUnit_examples/

# GUI

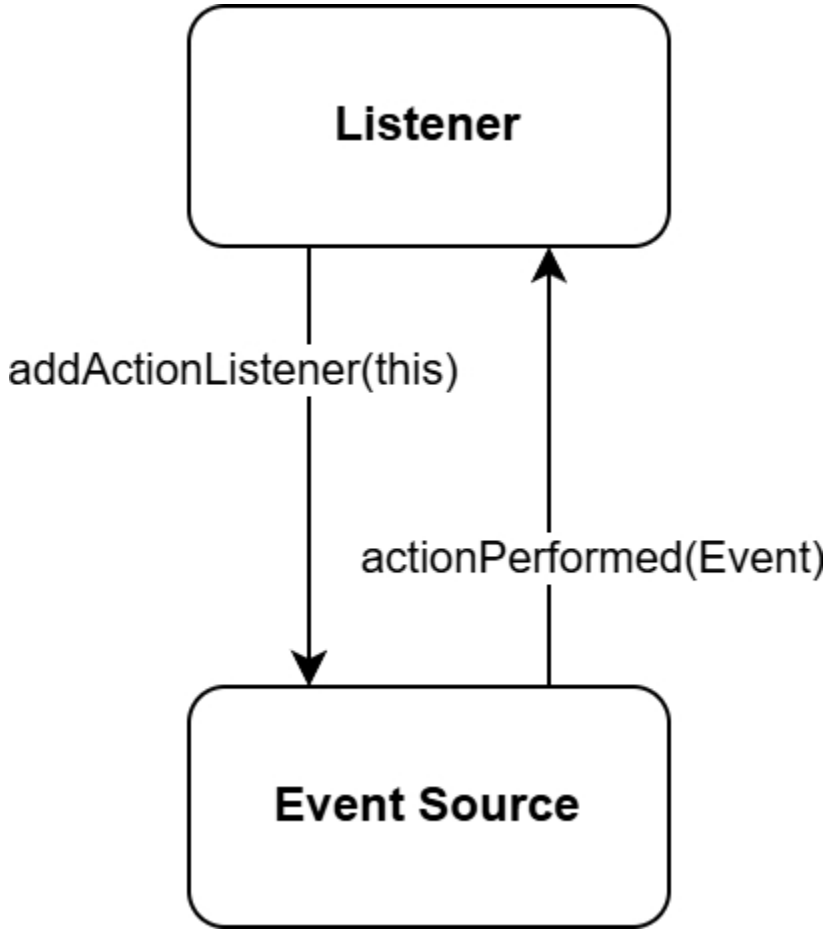
- GUI is always separated from application logic
- Swing, JavaFX, SWT, etc.

# Swing

- Directly in Java
- Lightweight
- Component framework
- MVC pattern:
  - Model – component data
  - View – visual representation of data
  - Controller – user input from view reflects into model data

# Event Listeners

- ActionListener
- ComponentListener
- ItemListener
- KeyListener
- MouseListener
- WindowListener
- AdjustmentListener
- ContainerListener
- MouseMotionListener
- FocusListener
- …



# Event

- Contains event data
- ActionEvent
- ComponentEvent
- ItemEvent
- …
- getActionCommand()
- getSource()
- getWhen()

# Event handling

- In the class itself
- In another class
- Anonymous class
- Lambda expression

# Nested classes

- Class within a class
- Can use attributes of the "outer" class, even private ones
- The "inner" object is bound to the "outer" one

- MyOuter outerObj = new MyOuter();
- MyOuter.MyInner innerObj = outerObj.new MyInner();

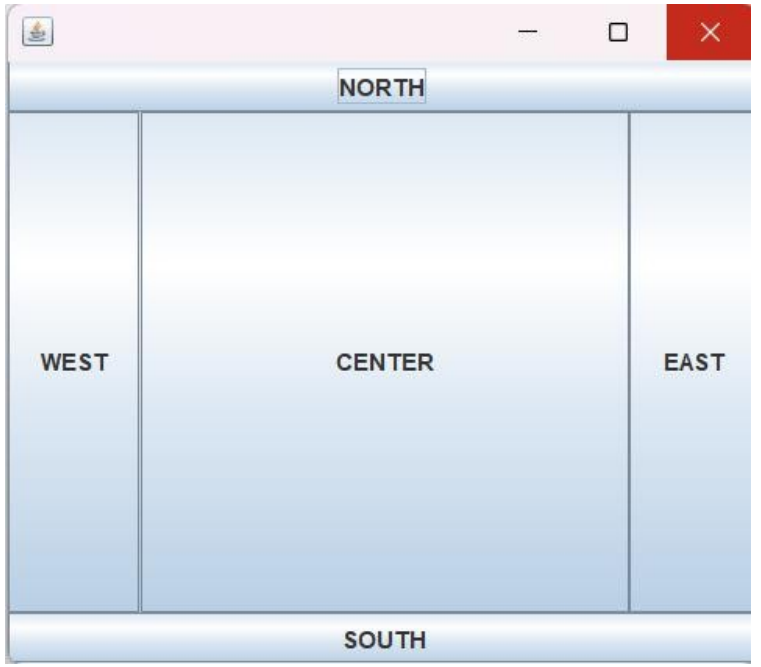
# Lambda expression

- label.addActionListener(event -> label.setText("Hello!"));

- Only when using a functional interface (SAM interface) – contains one abstract method (may also have other types)

# Layout

- Adding widgets (components) to the frame
- Drawing 2D graphics
- Inserting an image




- https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

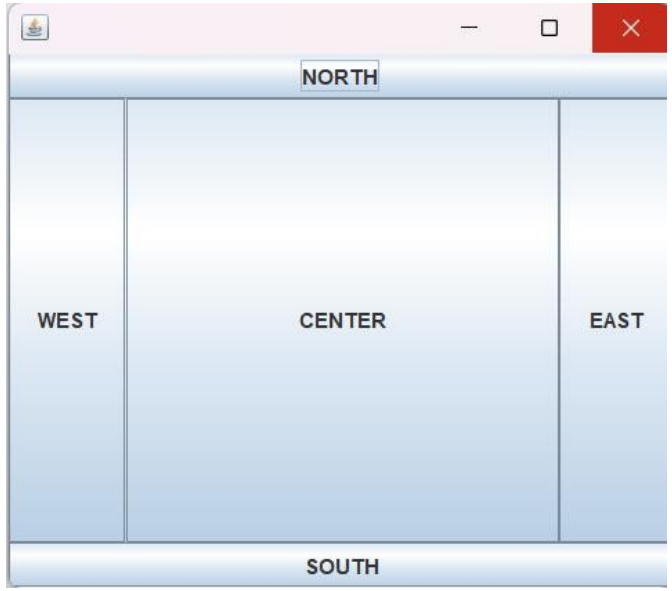
# Layout managers

- Components can be nested within each other as standard

- BorderLayout – one component per region, usually does not respect component dimensions (default for frame)
- FlowLayout – left to right, respects component dimensions, wrapping (default for panel)
- BoxLayout – similar to FlowLayout but from top to bottom, does not automatically wrap components
- GridBagLayout -

# Layouts

- Note other specifics, this is the "default" behavior





# BorderLayout

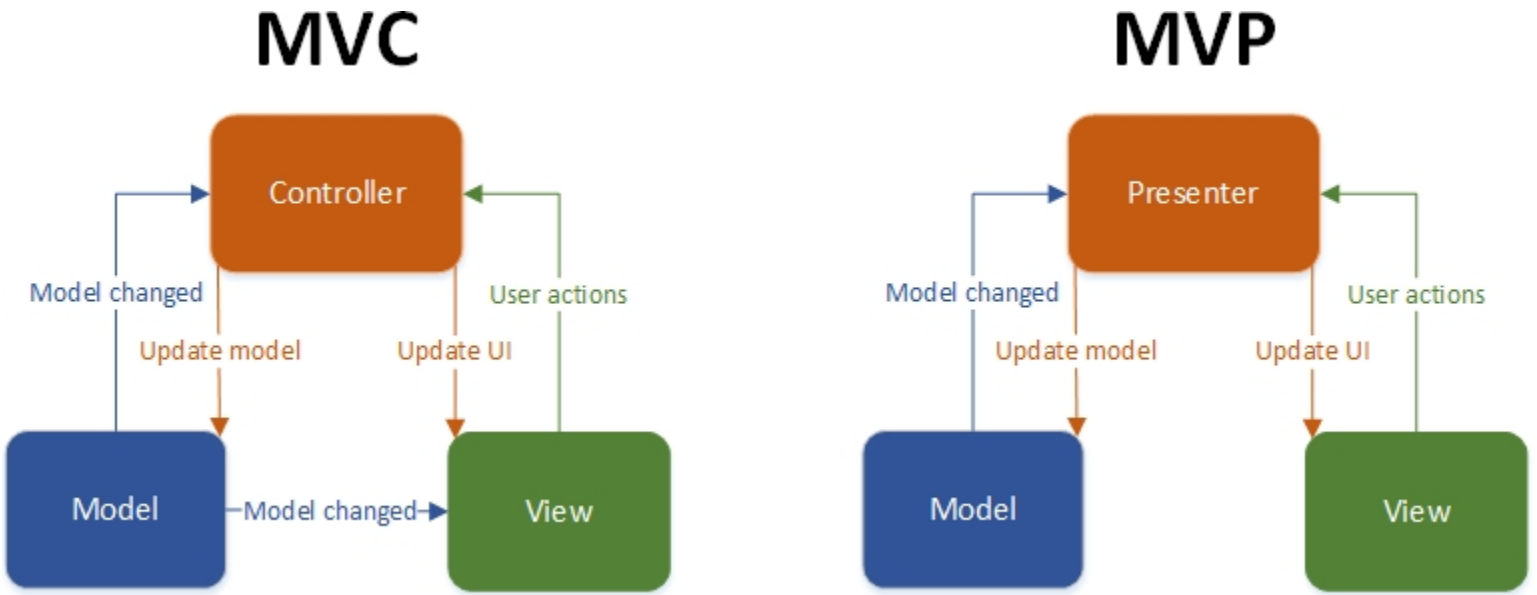
- How do regions behave?
- 1 component per region
- What about the size of components?
- It depends...

# Components in Swing

- Background (JFrame, JPanel)
- Interactive components:
  - JButton
  - JLabel
  - JTextField
  - JTextArea
  - JList

# GUI in general

- What happens when something takes longer?
- Separate data, application logic, and presentation layer
- Use an appropriate pattern: MVC vs MVP vs MVVM vs ...



# Quiz time