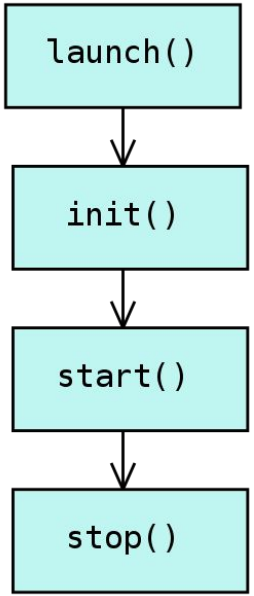
Object-Oriented Analysis & Java

Lecture 7

Florentin Bekier

JavaFX

Lifecycle of an application



UI controls

- **Label**: Component for placing text
- **Button**: Labeled button
- **ColorPicker**: Pane of controls designed to allow a user to manipulate and select a color
- **CheckBox**: Graphical component that can be in either an ON (true) or OFF (false) state

UI controls

- **RadioButton**: Graphical component, which can either be in a ON (true) or OFF (false) state in a group
- **ListView**: Component presents the user with a scrolling list of text items
- **TextField**: Text component that allows for the editing of a single line of text
- **PasswordField**: Text component specialized for password entry

UI controls

- **Scrollbar**: Scroll bar component in order to enable user to select from range of values
- **FileChooser**: Dialog window from which the user can select a file
- **ProgressBar**: As the task progresses towards completion, the progress bar displays the task's percentage of completion
- **Slider**: Lets the user graphically select a value by sliding a knob within a bounded interval

Layout panes

- Components in a scene are grouped within containers called Panes with a specific layout (arrangement)
- Following are the various Layout panes (classes) provided by JavaFX

Layout panes

- **HBox**: Arranges all the nodes in our application in a single horizontal row
- **VBox**: Arranges all the nodes in our application in a single vertical column
- **BorderPane**: Arranges the nodes in our application in top, left, right, bottom and center positions
- **StackPane**: Arranges the nodes in our application on top of another just like in a stack

Layout panes

- **TextFlow**: Arranges multiple text nodes in a single flow
- **AnchorPane**: Anchors the nodes in our application at a particular distance from the pane
- **TilePane**: Adds all the nodes of our application in the form of uniformly sized tiles
- **GridPane**: Arranges the nodes in our application as a grid of rows and columns. This layout comes handy while creating forms using JavaFX
- **FlowPane**: Wraps all the nodes in a flow. It can be either horizontal or vertical

Event handling

Types of events

- Clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen
- 2 types of events:
  - Foreground events (require the direct interaction of a user)
    e.g., clicking on a button, moving the mouse, etc.
  - Background events (require the interaction of end user)
    e.g., operating system interruptions, failure, etc.

Events

- Example of events in JavaFX:
  - Mouse event (mouse clicked, pressed, released, moved, mouse, etc.)
  - Key event (key pressed, key released and key typed)
  - Drag event (drag entered, drag dropped, drag entered target, drag exited target, drag over, etc.)
  - Window event (window hiding, window shown, window hidden, window showing, etc.)

Event handling

- Mechanism that controls the event and decides what should happen if an event occurs
- Event handler code that is executed when an event occurs
- In JavaFX every event has:
  - A target: the node on which an event occurred
  - A source: the source from which the event is generated will be the source of the event
  - A type: type of the occurred event (last slide)

Example

```
// Creating the mouse event handler
EventHandler<MouseEvent> eventHandler = new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent e) {
        // Code to handle the event
    }
};
// Adding event Filter
Target.addEventFilter(MouseEvent.MOUSE_CLICKED, eventHandler);
```