JavaFX MVC

AUDITORNE VJEŽBE

Sadržaj

Primjer kreiranja izbornika

Primjer kreiranja *ToolBar* komponente

Primjer kreiranja *Tab* komponenti

Primjer kreiranja *TableView* komponente

Primjer korištenja *TitledPane* komponente

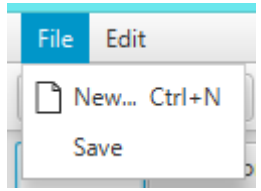
Primjer korištenja *SplitPane* i *TreeView* komponenti

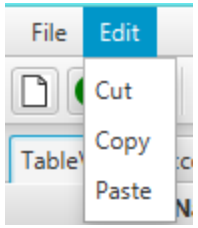
Primjer korištenja *TreeTableView* komponente

Primjer korištenja *ScrollPane* komponente

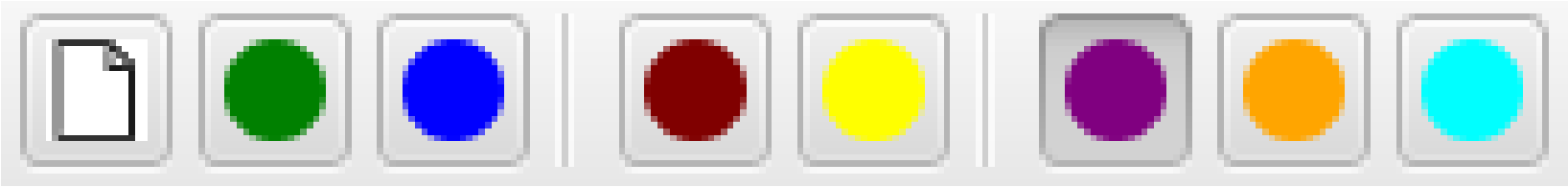
Primjer kreiranja izbornika

```
MenuItem itemNew = new MenuItem("New...", new ImageView(
        new Image(getClass().getResourceAsStream("images/paper.png"))));
itemNew.setAccelerator(KeyCombination.keyCombination("Ctrl+N"));
itemNew.setOnAction(e -> System.out.println(e.getEventType()
        + " occurred on MenuItem New"));
MenuItem itemSave = new MenuItem("Save");
Menu menuFile = new Menu("File");
menuFile.getItems().addAll(itemNew, itemSave);
MenuItem itemCut = new MenuItem("Cut");
MenuItem itemCopy = new MenuItem("Copy");
MenuItem itemPaste = new MenuItem("Paste");
Menu menuEdit = new Menu("Edit");
menuEdit.getItems().addAll(itemCut, itemCopy, itemPaste);
MenuBar menuBar = new MenuBar();
menuBar.getMenus().addAll(menuFile, menuEdit);
```





Primjer kreiranja *ToolBar* komponente



```
Button newButton = new Button();
newButton.setGraphic(new ImageView(new
        Image(getClass().getResourceAsStream("images/paper.png"))));
newButton.setId("newButton");
newButton.setTooltip(new Tooltip("New Document... Ctrl+N"));
newButton.setOnAction(e -> System.out.println("New toolbar button clicked"));
Button editButton = new Button();
editButton.setGraphic(new Circle(8, Color.GREEN));
editButton.setId("editButton");
Button deleteButton = new Button();
deleteButton.setGraphic(new Circle(8, Color.BLUE));
deleteButton.setId("deleteButton");
ToggleButton boldButton = new ToggleButton();
boldButton.setGraphic(new Circle(8, Color.MAROON));
boldButton.setId("boldButton");
```

Primjer kreiranja *ToolBar* komponente

```
boldButton.setOnAction(e -> {
    ToggleButton tb = ((ToggleButton) e.getTarget());
    System.out.print(e.getEventType() + " occurred on ToggleButton "
            + tb.getId());
    System.out.print(", and selectedProperty is: ");
    System.out.println(tb.selectedProperty().getValue());
});
ToggleButton italicButton = new ToggleButton();
italicButton.setGraphic(new Circle(8, Color.YELLOW));
italicButton.setId("italicButton");
italicButton.setOnAction(e -> {
    ToggleButton tb = ((ToggleButton) e.getTarget());
    System.out.print(e.getEventType() + " occurred on ToggleButton "
            + tb.getId());
    System.out.print(", and selectedProperty is: ");
    System.out.println(tb.selectedProperty().getValue());
});
```

Primjer kreiranja *ToolBar* komponente

```
final ToggleGroup alignToggleGroup = new ToggleGroup();
ToggleButton leftAlignButton = new ToggleButton();
leftAlignButton.setGraphic(new Circle(8, Color.PURPLE));
leftAlignButton.setId("leftAlignButton");
leftAlignButton.setToggleGroup(alignToggleGroup);
ToggleButton centerAlignButton = new ToggleButton();
centerAlignButton.setGraphic(new Circle(8, Color.ORANGE));
centerAlignButton.setId("centerAlignButton");
centerAlignButton.setToggleGroup(alignToggleGroup);
ToggleButton rightAlignButton = new ToggleButton();
rightAlignButton.setGraphic(new Circle(8, Color.CYAN));
rightAlignButton.setId("rightAlignButton");
rightAlignButton.setToggleGroup(alignToggleGroup);
```

Primjer kreiranja *ToolBar* komponente

```
ToolBar toolBar = new ToolBar(
        newButton, editButton, deleteButton,
        new Separator(Orientation.VERTICAL),boldButton,italicButton,
        new Separator(Orientation.VERTICAL),leftAlignButton,
        centerAlignButton,rightAlignButton
);

alignToggleGroup.selectToggle(alignToggleGroup.getToggles().get(0));
alignToggleGroup.selectedTogglePropery().addListener(
(ov, oldValue, newValue) -> {
    ToggleButton tb = ((ToggleButton)
     alignToggleGroup.getSelectedToggle());
    if (tb != null) {
        System.out.println(tb.getId() + " selected");
    }
});
```

Primjer kreiranja *Tab* komponenti


```
  final WebView webView = new WebView();
Tab tableTab = new Tab("TableView");
tableTab.setContent(createTableDemoNode());
tableTab.setClosable(false);
Tab accordionTab = new Tab("Accordion/TitledPane");
accordionTab.setContent(createAccordionTitledDemoNode());
accordionTab.setClosable(false);
Tab splitTab = new Tab("SplitPane/TreeView/ListView");
splitTab.setContent(createSplitTreeListDemoNode());
splitTab.setClosable(false);
Tab treeTableTab = new Tab("TreeTableView");
treeTableTab.setContent(createTreeTableDemoNode());
treeTableTab.setClosable(false);
Tab scrollTab = new Tab("ScrollPane/Miscellaneous");
scrollTab.setContent(createScrollMiscDemoNode());
scrollTab.setClosable(false);
```

Primjer kreiranja *Tab* komponenti

```
Tab htmlTab = new Tab("HTMLEditor");
htmlTab.setContent(createHtmlEditorDemoNode());
htmlTab.setClosable(false);
Tab webViewTab = new Tab("WebView");
webViewTab.setContent(webView);
webViewTab.setClosable(false);
webViewTab.setOnSelectionChanged(e -> {
    String randomWebSite = model.getRandomWebSite();
    if (webViewTab.isSelected()) {
        webView.getEngine().load(randomWebSite);
        System.out.println("WebView tab is selected, loading: "
                + randomWebSite);
    }
});
```

Primjer kreiranja *Tab* komponenti

```
TabPane tabPane = new TabPane();
tabPane.getTabs().addAll(
        tableTab,
        accordionTab,
        splitTab,
        treeTableTab,
        scrollTab,
        htmlTab,
        webViewTab
);
```

Primjer kreiranja *TableView* komponente

First Name	Last Name	Phone Number
FirstName1	LastName1	Phone1
FirstName2	LastName2	Phone2
FirstName3	LastName3	Phone3
FirstName4	LastName4	Phone4
FirstName5	LastName5	Phone5
FirstName6	LastName6	Phone6
FirstName7	LastName7	Phone7
FirstName8	LastName8	Phone8
FirstName9	LastName9	Phone9
FirstName10	LastName10	Phone10

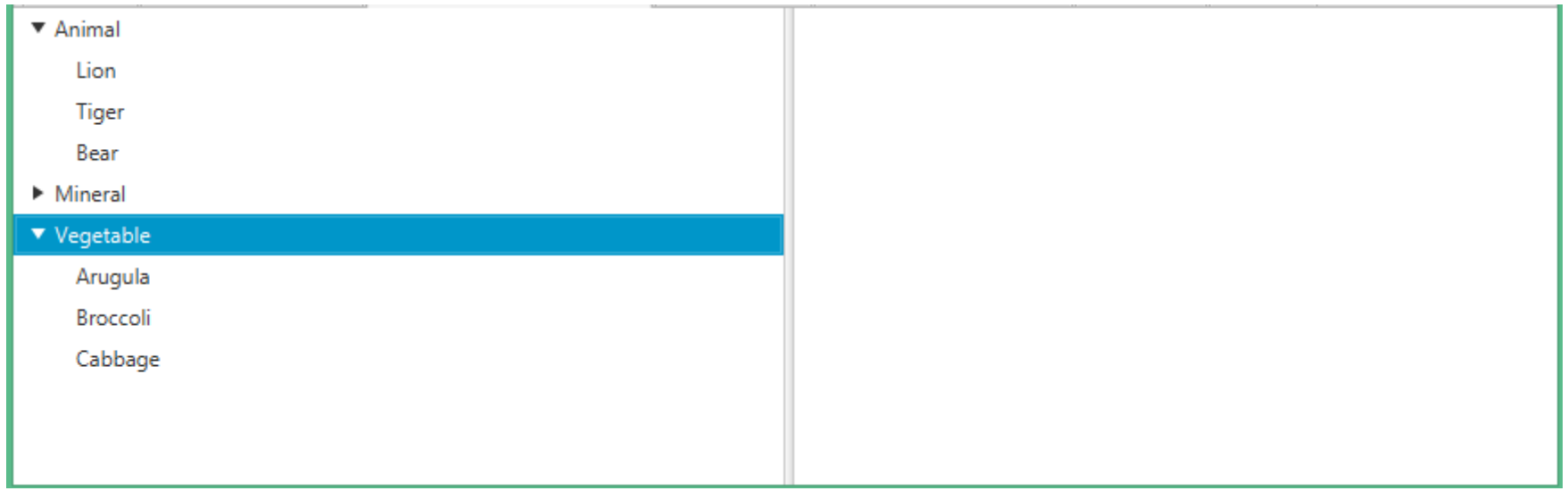
Primjer kreiranja *TableView* komponente

```
TableView table = new TableView(model.getTeamMembers());
TableColumn firstNameColumn = new TableColumn("First Name");
firstNameColumn.setCellValueFactory(new PropertyValueFactory("firstName"));
firstNameColumn.setPrefWidth(180);
TableColumn lastNameColumn = new TableColumn("Last Name");
lastNameColumn.setCellValueFactory(new PropertyValueFactory("lastName"));
lastNameColumn.setPrefWidth(180);
TableColumn phoneColumn = new TableColumn("Phone Number");
phoneColumn.setCellValueFactory(new PropertyValueFactory("phone"));
phoneColumn.setPrefWidth(180);
table.getColumns().addAll(firstNameColumn, lastNameColumn, phoneColumn);
table.getSelectionModel().selectedItemProperty()
        .addListener((ObservableValue observable, Object oldValue, Object newValue) -> {
            Person selectedPerson = (Person) newValue;
            System.out.println(selectedPerson + " chosen in TableView");
        });
```

Primjer kreiranja *TitledPane* komponente

```
TitledPane paneA = new TitledPane("TitledPane A", new TextArea("TitledPane A content"));
TitledPane paneB = new TitledPane("TitledPane B", new TextArea("TitledPane B content"));
TitledPane paneC = new TitledPane("TitledPane C", new TextArea("TitledPane C content"));
Accordion accordion = new Accordion();
accordion.getPanes().addAll(paneA, paneB, paneC);
accordion.setExpandedPane(paneA);
```

Primjer kreiranja *SplitPane* i *TreeView* komponenti



Primjer kreiranja *SplitPane* i *TreeView* komponenti

```
TreeItem animalTree = new TreeItem("Animal");
animalTree.getChildren().addAll(new TreeItem("Lion"), new TreeItem("Tiger"),
        new TreeItem("Bear"));
TreeItem mineralTree = new TreeItem("Mineral");
mineralTree.getChildren().addAll(new TreeItem("Copper"), new TreeItem("Diamond"),
        new TreeItem("Quartz"));
TreeItem vegetableTree = new TreeItem("Vegetable");
vegetableTree.getChildren().addAll(new TreeItem("Arugula"), new TreeItem("Broccoli"),
        new TreeItem("Cabbage"));

TreeItem root = new TreeItem("Root");
root.getChildren().addAll(animalTree, mineralTree, vegetableTree);
TreeView treeView = new TreeView(root);
treeView.setMinWidth(150);
treeView.setShowRoot(false);
treeView.setEditable(false);
```

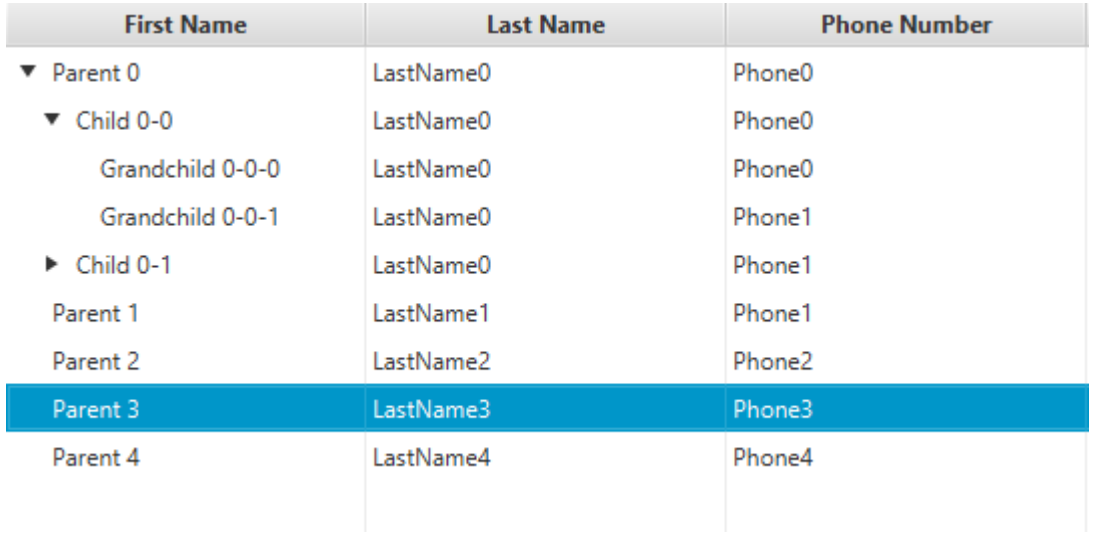
Primjer kreiranja *SplitPane* i *TreeView* komponenti

```
ListView listView = new ListView(model.listViewItems);

SplitPane splitPane = new SplitPane();
splitPane.getItems().addAll(treeView, listView);

treeView.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
treeView.getSelectionModel().selectedItemProperty()
        .addListener((ObservableValue observable, Object oldValue, Object newValue) ->
       {
            TreeItem treeItem = (TreeItem) newValue;
            if (newValue != null && treeItem.isLeaf()) {
                model.listViewItems.clear();
                for (int i = 1; i <= 10000; i++) {
                    model.listViewItems.add(treeItem.getValue() + " " + i);
                }
            }
        });
```

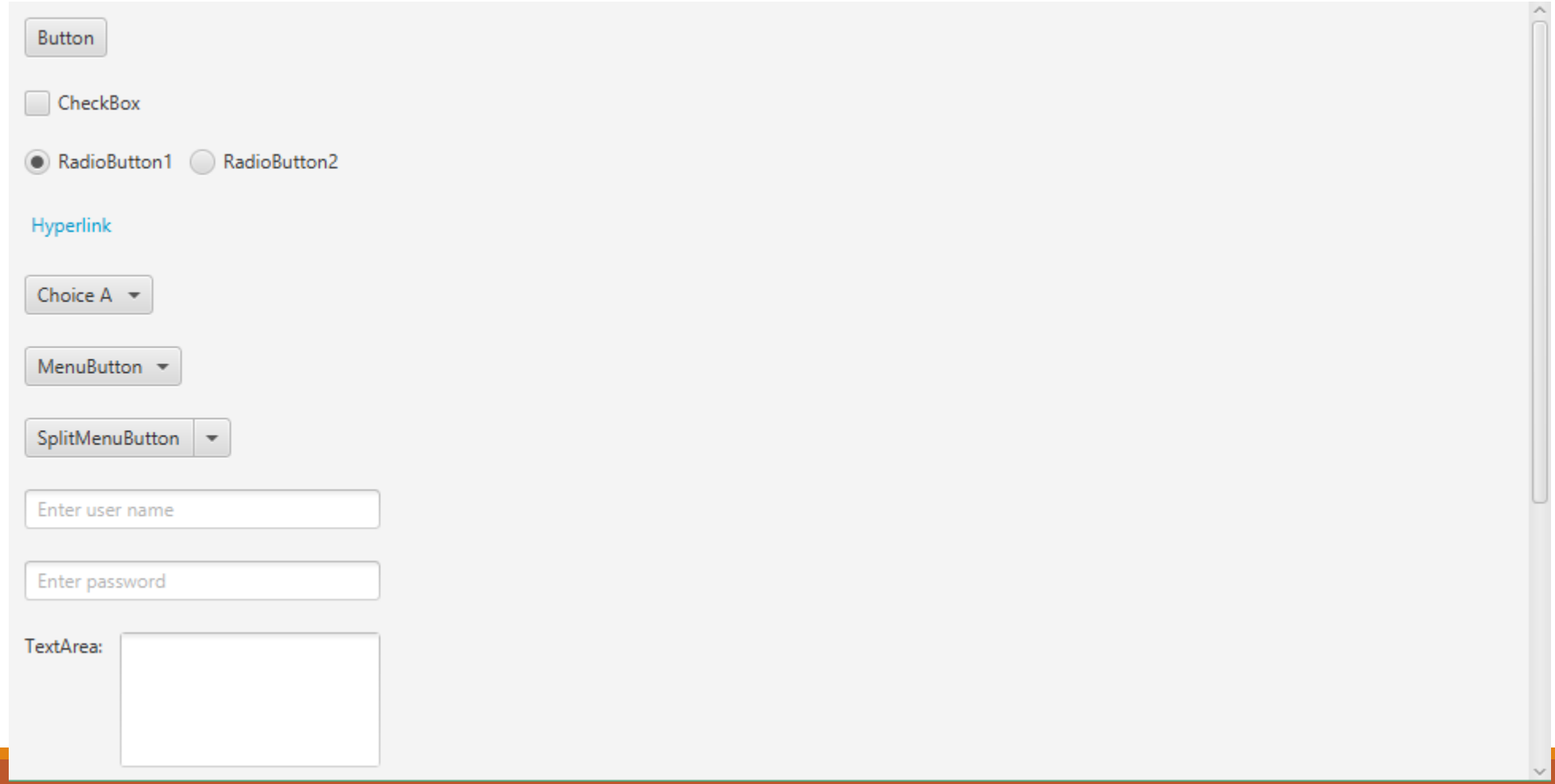
Primjer kreiranja *TreeTableView* komponente



Primjer kreiranja *TreeTableView* komponente

```
TreeTableView<Person> treeTableView = new TreeTableView(model.getFamilyTree());
TreeTableColumn<Person, String> firstNameColumn = new TreeTableColumn("First Name");
firstNameColumn.setCellValueFactory(new TreeItemPropertyValueFactory("firstName"));
firstNameColumn.setPrefWidth(180);
TreeTableColumn lastNameColumn = new TreeTableColumn("Last Name");
lastNameColumn.setCellValueFactory(new TreeItemPropertyValueFactory("lastName"));
lastNameColumn.setPrefWidth(180);
TreeTableColumn phoneColumn = new TreeTableColumn("Phone Number");
phoneColumn.setCellValueFactory(new TreeItemPropertyValueFactory("phone"));
phoneColumn.setPrefWidth(180);
treeTableView.getColumns().addAll(firstNameColumn, lastNameColumn, phoneColumn);
treeTableView.getSelectionModel().selectedItemProperty().addListener(
 (ObservableValue<? extends TreeItem<Person>> observable, TreeItem<Person> oldValue,
  TreeItem<Person> newValue) -> {
     Person selectedPerson = newValue.getValue();
     System.out.println(selectedPerson + " chosen in TreeTableView");
});
treeTableView.setShowRoot(false);
```

Primjer kreiranja *ScrollPane* komponente



Primjer kreiranja *ScrollPane* komponente

```
Button button = new Button("Button");
button.setOnAction(e -> System.out.println(e.getEventType() +
      " occurred on Button"));
final CheckBox checkBox = new CheckBox("CheckBox");
checkBox.setOnAction(e -> {
    System.out.print(e.getEventType() + " occurred on CheckBox");
    System.out.print(", and selectedProperty is: ");
    System.out.println(checkBox.selectedProperty().getValue());
});

final ToggleGroup radioToggleGroup = new ToggleGroup();
RadioButton radioButton1 = new RadioButton("RadioButton1");
radioButton1.setToggleGroup(radioToggleGroup);
RadioButton radioButton2 = new RadioButton("RadioButton2");
radioButton2.setToggleGroup(radioToggleGroup);
HBox radioBox = new HBox(10, radioButton1, radioButton2);
```

Primjer kreiranja *ScrollPane* komponente

```
Hyperlink link = new Hyperlink("Hyperlink");
link.setOnAction(e -> System.out.println(e.getEventType() + " occurred on Hyperlink"));

ChoiceBox choiceBox;
choiceBox = new ChoiceBox(model.choiceBoxItems);
choiceBox.getSelectionModel().selectFirst();
choiceBox.getSelectionModel().selectedItemProperty()
        .addListener((observable, oldValue, newValue) -> {
            System.out.println(newValue + " chosen in ChoiceBox");
        });

MenuItem menuA = new MenuItem("MenuItem A");
menuA.setOnAction(e -> System.out.println(e.getEventType() +
       " occurred on Menu Item A"));
MenuItem menuB = new MenuItem("MenuItem B");
MenuButton menuButton = new MenuButton("MenuButton");
menuButton.getItems().addAll(menuA, menuB);
```

Primjer kreiranja *ScrollPane* komponente

```
MenuItem splitMenuA = new MenuItem("MenuItem A");
splitMenuA.setOnAction(e -> System.out.println(e.getEventType()
        + " occurred on Menu Item A"));
MenuItem splitMenuB = new MenuItem("MenuItem B");
SplitMenuButton splitMenuButton = new SplitMenuButton(splitMenuA, splitMenuB);
splitMenuButton.setText("SplitMenuButton");
splitMenuButton.setOnAction(e -> System.out.println(e.getEventType()
        + " occurred on SplitMenuButton"));

final TextField textField = new TextField();
textField.setPromptText("Enter user name");
textField.setPrefColumnCount(16);
textField.textProperty().addListener((ov, oldValue, newValue) -> {
    System.out.println("TextField text is: " + textField.getText());
});
```

Primjer kreiranja *ScrollPane* komponente

```
final PasswordField passwordField = new PasswordField();
passwordField.setPromptText("Enter password");
passwordField.setPrefColumnCount(16);
passwordField.focusedProperty().addListener((ov, oldValue, newValue) -> {
    if (!passwordField.isFocused()) {
        System.out.println("PasswordField text is: "
                + passwordField.getText());
    }
});

final TextArea textArea = new TextArea();
textArea.setPrefColumnCount(12);
textArea.setPrefRowCount(4);
textArea.focusedProperty().addListener((ov, oldValue, newValue) -> {
    if (!textArea.isFocused()) {
        System.out.println("TextArea text is: " + textArea.getText());
    }
});
```

Primjer kreiranja *ScrollPane* komponente

```
 LocalDate today = LocalDate.now();
DatePicker datePicker = new DatePicker(today);
datePicker.setOnAction(e -> System.out.println("Selected date: " +
        datePicker.getValue()));

ColorPicker colorPicker = new ColorPicker(Color.BLUEVIOLET);
colorPicker.setOnAction(e -> System.out.println("Selected color: " +
        colorPicker.getValue()));

final ProgressIndicator progressIndicator = new ProgressIndicator();
progressIndicator.setPrefWidth(200);
progressIndicator.progressProperty().bind(model.rpm.divide(model.maxRpm));

final Slider slider = new Slider(-1, model.maxRpm, 0);
slider.setPrefWidth(200);
slider.valueProperty().bindBidirectional(model.rpm);
```

Primjer kreiranja *ScrollPane* komponente

```
final ProgressBar progressBar = new ProgressBar();
progressBar.setPrefWidth(200);
progressBar.progressProperty().bind(model.kph.divide(model.maxKph));

final ScrollBar scrollBar = new ScrollBar();
scrollBar.setPrefWidth(200);
scrollBar.setMin(-1);
scrollBar.setMax(model.maxKph);
scrollBar.valueProperty().bindBidirectional(model.kph);
```

Primjer kreiranja *ScrollPane* komponente

```
VBox variousControls = new VBox(20,
            button,
            checkBox,
            radioBox,
            link,
            choiceBox,
            menuButton,
            splitMenuButton,
            textField,
            passwordField,
            new HBox(10, new Label("TextArea:"), textArea),
            datePicker, colorPicker,
            progressIndicator, slider,
            progressBar, scrollBar);
```

Primjer kreiranja *ScrollPane* komponente

```
variousControls.setPadding(new Insets(10, 10, 10, 10));
radioToggleGroup.selectToggle(radioToggleGroup.getToggles().get(0));
radioToggleGroup.selectedTogglePropery().addListener(
(ov, oldValue, newValue) -> {
    RadioButton rb = ((RadioButton) radioToggleGroup.getSelectedToggle());
    if (rb != null) {
        System.out.println(rb.getText() + " selected");
    }
});

MenuItem contextA = new MenuItem("MenuItem A");
contextA.setOnAction(e -> System.out.println(e.getEventType()
        + " occurred on Menu Item A"));
MenuItem contextB = new MenuItem("MenuItem B");
final ContextMenu contextMenu = new ContextMenu(contextA, contextB);
```

Primjer kreiranja *ScrollPane* komponente

```
ScrollPane scrollPane = new ScrollPane(variousControls);
scrollPane.setHbarPolicy(ScrollPane.ScrollBarPolicy.NEVER);
scrollPane.setVbarPolicy(ScrollPane.ScrollBarPolicy.AS_NEEDED);
scrollPane.setOnMousePressed((MouseEvent me) -> {
    if (me.getButton() == MouseButton.SECONDARY) {
        contextMenu.show(stage, me.getScreenX(), me.getScreenY());
    }
});
```

Pitanja?