Jesús Serrano Pérez
3/2/2022

1. Seguir els passos indicats al tutorial fins arrribar a tenir l'AddressApp en l'estat que es mostra al final.
Enregistrar cada pas que es dona.

Afegir nous métodes a la classe PersonOverviewController.

```java
/**
 * Fills all text fields to show details about the person.
 * If the specified person is null, all text fields are cleared.
 *
 * @param person the person or null
 */
private void showPersonDetails(Person person) {
    if (person != null) {
        // Fill the labels with info from the person object.
        firstNameLabel.setText(person.getFirstName());
        lastNameLabel.setText(person.getLastName());
        streetLabel.setText(person.getStreet());
        postalCodeLabel.setText(Integer.toString(person.getPostalCode()));
        cityLabel.setText(person.getCity());
        birthdayLabel.setText(DateUtil.format(person.getBirthday()));
    } else {
        // Person is null, remove all the text.
        firstNameLabel.setText("");
        lastNameLabel.setText("");
        streetLabel.setText("");
        postalCodeLabel.setText("");
        cityLabel.setText("");
        birthdayLabel.setText("");
    }
}
```

```java
/**
 * Called when the user clicks on the delete button.
 */
@FXML
private void handleDeletePerson() {
    int selectedIndex = personTable.getSelectionModel().getSelectedIndex();
    if (selectedIndex >= 0) {
        personTable.getItems().remove(selectedIndex);
    } else {
        // Nothing selected.
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("No Selection");
        alert.setHeaderText("No Person Selected");
        alert.setContentText("Please select a person in the table.");
        alert.showAndWait();
    }
}

/**
 * Called when the user clicks the new button. Opens a dialog to edit
 * details for a new person.
 */
@FXML
private void handleNewPerson() {
    Person tempPerson = new Person();
    boolean okClicked;
    okClicked = mainApp.showPersonEditDialog(tempPerson);
    if (okClicked) {
        mainApp.getPersonData().add(tempPerson);
    }
}
```

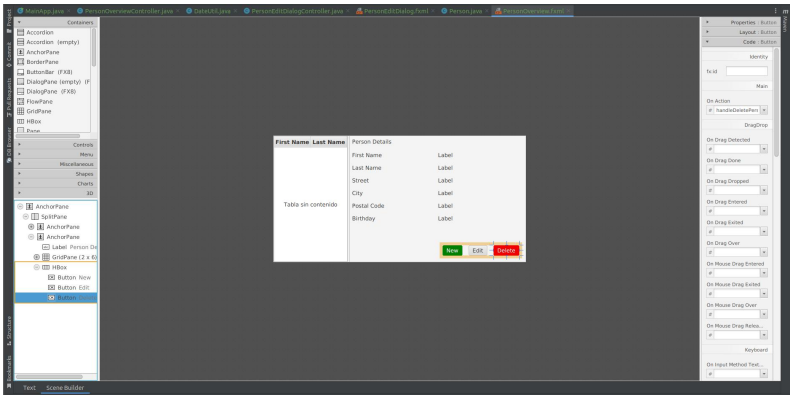
```java
/**
 * Called when the user clicks the edit button. Opens a dialog to edit
 * details for the selected person.
 */
@FXML
private void handleEditPerson() {
    Person selectedPerson = personTable.getSelectionModel().getSelectedItem();
    if (selectedPerson != null) {
        boolean okClicked = mainApp.showPersonEditDialog(selectedPerson);
        if (okClicked) {
            showPersonDetails(selectedPerson);
        }

    } else {
        // Nothing selected.
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("No Selection");
        alert.setHeaderText("No Person Selected");
        alert.setContentText("Please select a person in the table.");
        alert.showAndWait();
    }
}
```

Jesús Serrano Pérez
3/2/2022

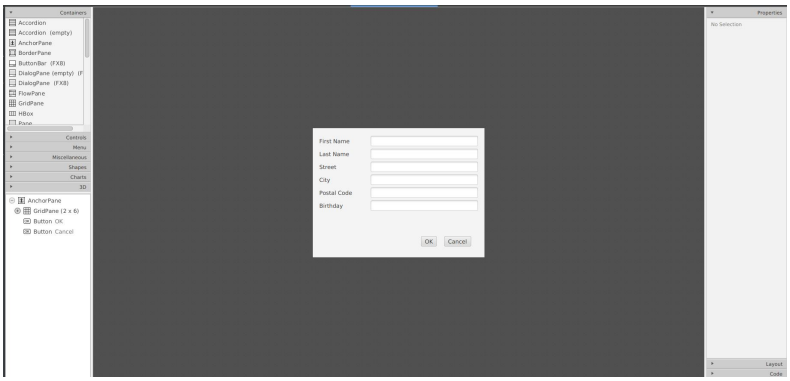
Crear una classe per controlar les dates.

```java
package com.example.newfirstapp;

import ...

/** Helper functions for handling dates. ...*/
public class DateUtil {

    /** The date pattern that is used for conversion. Change as you wish. */
    private static final String DATE_PATTERN = "dd.MM.yyyy";

    /** The date formatter. */
    private static final DateTimeFormatter DATE_FORMATTER =
            DateTimeFormatter.ofPattern(DATE_PATTERN);

    /** Returns the given date as a well formatted String. The above defined ...*/
    public static String format(LocalDate date) {
        if (date == null) {
            return null;
        }
        return DATE_FORMATTER.format(date);
    }

    /** Converts a String in the format of the defined {@link DateUtil#DATE_PATTERN} ...*/
    public static LocalDate parse(String dateString) {
        try {
            return DATE_FORMATTER.parse(dateString, LocalDate::from);
        } catch (DateTimeParseException e) {
            return null;
        }
    }

    /** Checks the String whether it is a valid date. ...*/
    public static boolean validDate(String dateString) {
        // Try to parse the String.
        return DateUtil.parse(dateString) != null;
    }
}
```

Associar els control·ladors amb els butons.



Crear una nova finestra PersonEditDialog.



Jesús Serrano Pérez
3/2/2022

Crear un nou control·lador per a la nova finestra.

```java
package com.example.newfirstapp;

import ...

/** Dialog to edit details of a person. ...*/
public class PersonEditDialogController {

    @FXML
    private TextField firstNameField;
    @FXML
    private TextField lastNameField;
    @FXML
    private TextField streetField;
    @FXML
    private TextField postalCodeField;
    @FXML
    private TextField cityField;
    @FXML
    private TextField birthdayField;


    private Stage dialogStage;
    private Person person;
    private boolean okClicked = false;

    /** Initializes the controller class. This method is automatically called ...*/
    @FXML
    private void initialize() {}

    /** Sets the stage of this dialog. */
    public void setDialogStage(Stage dialogStage) { this.dialogStage = dialogStage; }

    /** Sets the person to be edited in the dialog. */
    public void setPerson(Person person) {
        this.person = person;

        firstNameField.setText(person.getFirstName());
        lastNameField.setText(person.getLastName());
        streetField.setText(person.getStreet());
        postalCodeField.setText(Integer.toString(person.getPostalCode()));
        cityField.setText(person.getCity());
        birthdayField.setText(DateUtil.format(person.getBirthday()));
        birthdayField.setPromptText("dd.mm.yyyy");
    }

    /** Returns true if the user clicked OK, false otherwise. */
    public boolean isOkClicked() { return okClicked; }

    /** Called when the user clicks ok. */
    @FXML
    private void handleOk() {
        if (isInputValid()) {
            person.setFirstName(firstNameField.getText());
            person.setLastName(lastNameField.getText());
            person.setStreet(streetField.getText());
            person.setPostalCode(Integer.parseInt(postalCodeField.getText()));
            person.setCity(cityField.getText());
            person.setBirthday(DateUtil.parse(birthdayField.getText()));

            okClicked = true;
            dialogStage.close();
        }
    }

    /** Called when the user clicks cancel. */
    @FXML
    private void handleCancel() { dialogStage.close(); }

    /** Validates the user input in the text fields. ...*/
    private boolean isInputValid() {
        String errorMessage = "";

        if (firstNameField.getText() == null || firstNameField.getText().length() == 0) {
            errorMessage += "No valid first name!\n";
        }
        if (lastNameField.getText() == null || lastNameField.getText().length() == 0) {
            errorMessage += "No valid last name!\n";
        }
        if (streetField.getText() == null || streetField.getText().length() == 0) {
            errorMessage += "No valid street!\n";
        }

        if (postalCodeField.getText() == null || postalCodeField.getText().length() == 0) {
            errorMessage += "No valid postal code!\n";
        } else {
            // try to parse the postal code into an int.
            try {
                Integer.parseInt(postalCodeField.getText());
            } catch (NumberFormatException e) {
                errorMessage += "No valid postal code (must be an integer)!\n";
            }
        }

        if (cityField.getText() == null || cityField.getText().length() == 0) {
            errorMessage += "No valid city!\n";
        }

        if (birthdayField.getText() == null || birthdayField.getText().length() == 0) {
            errorMessage += "No valid birthday!\n";
        } else {
            if (!DateUtil.validDate(birthdayField.getText())) {
                errorMessage += "No valid birthday. Use the format dd.mm.yyyy!\n";
            }
        }

        if (errorMessage.length() == 0) {
            return true;
        } else {
            // Show the error message.
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Invalid Fields");
            alert.setHeaderText("Please correct invalid fields");
            alert.setContentText(errorMessage);
            alert.showAndWait();
            return false;
        }
    }
}
```

Jesús Serrano Pérez
3/2/2022

Enllaçar el control·lador amb la finestra.

Afegir funcionalitat per obrir la finestra d'edició de persones, crear i eliminar.

```java
/** Opens a dialog to edit details for the specified person. If the user ...*/
public boolean showPersonEditDialog(Person person) {
    try {
        // Load the fxml file and create a new stage for the popup dialog.
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(MainApp.class.getResource( name: "PersonEditDialog.fxml"));
        AnchorPane page = loader.load();

        // Create the dialog Stage.
        Stage dialogStage = new Stage();
        dialogStage.setTitle("Edit Person");
        dialogStage.initModality(Modality.WINDOW_MODAL);
        dialogStage.initOwner(primaryStage);
        Scene scene = new Scene(page);
        dialogStage.setScene(scene);

        // Set the person into the controller.
        PersonEditDialogController controller = loader.getController();
        controller.setDialogStage(dialogStage);
        controller.setPerson(person);

        // Show the dialog and wait until the user closes it
        dialogStage.showAndWait();

        return controller.isOkClicked();
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

2. Quina diferència de comportament s'observa en els camps a la dreta en sel·leccionar una persona?

   Podem observar que en sel·leccionar a una persona, els camps de first i last name canvien, però en canvi la resta de camps es mantenen. Això es perque al crear-les només s'especifica nom i cognoms, i el programa utilitza els valors per defecte per la resta de camps sense definir.

3. Tipus de dialegs:
   - Informació.
   - Informació sense header.
   - Avís.
   - Error.
   - Excepció (mostra el stack trace)
   - Confirmació
   - Confirmació amb opcions personalitzades
   - Text input
   - Desplegable d'opcions
   - Login

4. Per a què serveix l'etiqueta @FXML

   L'etiqueta @FXML serveix per indicar a Java que l'objecte declarat es troba a un arxiu FXML, i aquest es accesible per l'aplicació.

5. Explicar com assignar un tractament com a resposta a un event associat a un node.
Explicar com fer-ho des del SceneBuilder i mostrar els canvis que això provoca en el fxml.

   Des de les classes controller podem alterar el text o contingut de l'FXML durant l'execució, però aquesta operació no es reflexa en l'arxiu FXML. Un exemple es cuan afegim persones, o quan en sel·leccionar una, els valor de la dreta varíen, pero aquests canvis no es reflexen en l'arxiu.