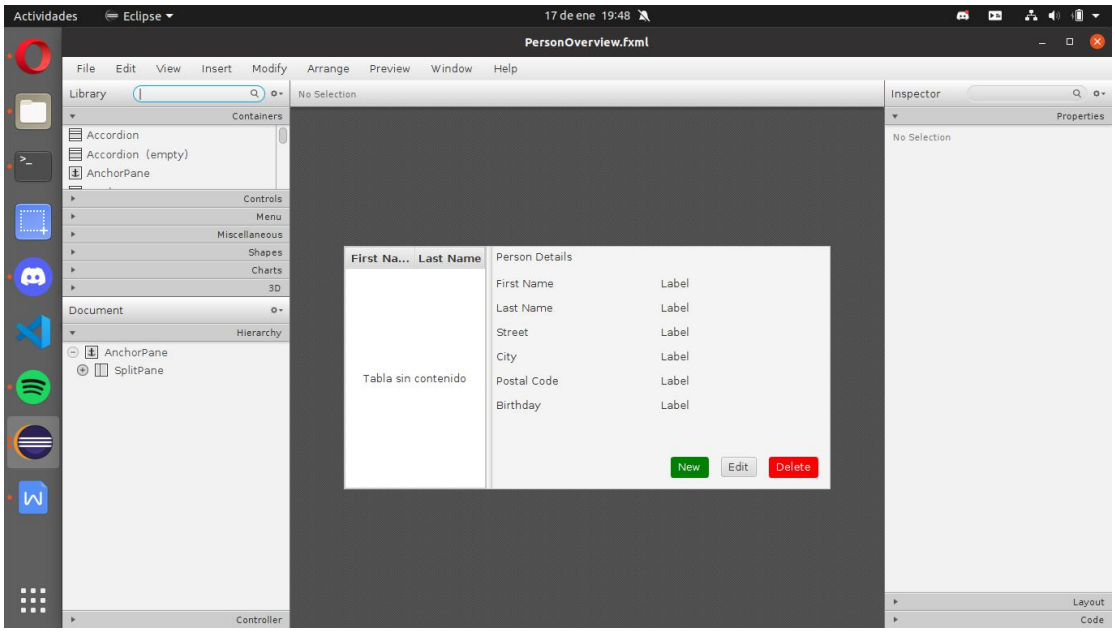
Jesús Serrano
17/01/2022

1. Seguir els passos indicats al tutorial fins arribar a tenir l'AddressApp en l'estat que es mostra al final. Enregistrar en un diari cada pas que es dona, els problemes que es troben i la manera en què se solucionen. Incloure-hi captures de pantalla comentades de cada pas i del resultat final.



2. Analitzar y explicar els mètodes initRootLayout() y showPersonOverview(). (Consultar la documentació del package javafx) Localitzar clarament el moment en que la informació expressada en FXML origina un objecte Java, manipulable des del codi.

```
/**
 * Initializes the root layout.
 */
public void initRootLayout() {
    try {
        // Load root layout from fxml file.
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(MainApp.class.getResource("view/RootLayout.fxml"));
        rootLayout = (BorderPane) loader.load();

        // Show the scene containing the root layout.
        Scene scene = new Scene(rootLayout);
        primaryStage.setScene(scene);
        primaryStage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

loader.load(); carga el objeto java que es indicado en loader.setLocation (en este caso RootLayout. Despues creas un scene con ese layout y con primaryStage.show() la muestras

```java
/**
 * Shows the person overview inside the root layout.
 */
public void showPersonOverview() {
    try {
        // Load person overview.
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(MainApp.class.getResource("view/PersonOverview.fxml"));
        AnchorPane personOverview = (AnchorPane) loader.load();

        // Set person overview into the center of root layout.
        rootLayout.setCenter(personOverview);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Aqui estas cargando un anchorPane de rootLayout. Tambien se carga el objeto en el loader.load y se empieza a mostrar en rootLayout.setCenter

3. En aquesta sessió es descriu l'aspecte de la interfície amb un llenguatge de marcat (FXML), utilitzant un editor gràfic (SceneBuilder). Una aproximació alternativa, i complementària, seria programar en Java els objectes que hi intervenen i la seva relació. En l'exercici realitzat, quan s'utilitza cadascun d'aquests dos enfocs ?

En aquest exercici, es dissenya tota la UI mitjançant SceneBuilder, i programant en Java s'especifica la funcionalitat de cadascún dels objectes en pantalla (labels, buttons, taules, etc...)

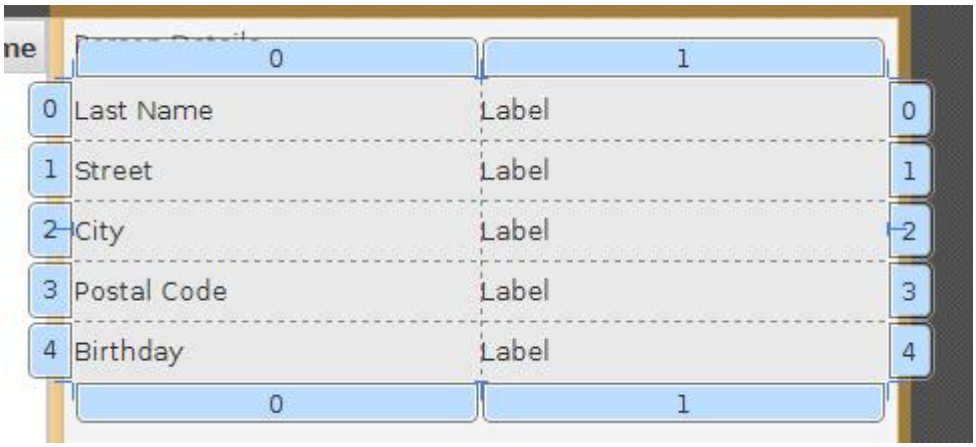
4. Descriure cada element gràfic utilitzat en l'exercici (contenidors i nodes). Indicar com es reflexen al codi FXML les modificacions sobre objectes realitzades amb el SceneBuilder.

Grid:

```
<GridPane layoutX="108.0" layoutY="104.0" AnchorPane.leftAnchor="5.0" AnchorPane.rightAnchor="5.0"
AnchorPane.topAnchor="30.0"> Declaración del GridPane y sus propiedades (Tamaño y márgenes)
                    <columnConstraints>   Declaración de columnas (2). Cada una tiene sus propiedades individuales
                      <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                      <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                    </columnConstraints>
                    <rowConstraints> Declaración de las diferentes filas y las propiedades de cada una
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                    </rowConstraints>
                     <children>   Declaración de los diferentes label que hay en cada celda. Se indica el texto y posición
                        <Label text="First Name" />
                        <Label text="Last Name" GridPane.rowIndex="1" />
                        <Label text="Street" GridPane.rowIndex="2" />
                        <Label text="Label" GridPane.columnIndex="1" />
                        <Label text="Label" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                        <Label text="Label" GridPane.columnIndex="1" GridPane.rowIndex="2" />
                        <Label text="City" GridPane.rowIndex="3" />
                        <Label text="Postal Code" GridPane.rowIndex="4" />
                        <Label text="Label" GridPane.columnIndex="1" GridPane.rowIndex="3" />
                        <Label text="Label" GridPane.columnIndex="1" GridPane.rowIndex="4" />
                        <Label text="Birthday" GridPane.rowIndex="5" />
                        <Label text="Label" GridPane.columnIndex="1" GridPane.rowIndex="5" />
                      </children>
                  </GridPane>
```

Jesús Serrano
17/01/2022

Previsualización del grid:



Label:

```
<Label layoutX="8.0" layoutY="7.0" text="Person Details" AnchorPane.leftAnchor="5.0" AnchorPane.topAnchor="5.0" />
```

Label tiene una posición fija, un texto, y unos márgenes.

Previsualización del Label:

Buttons:

```
<HBox layoutX="220.0" layoutY="259.0" spacing="15.0">
```

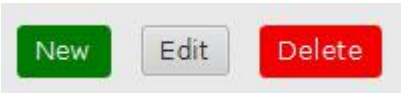
Los buttons se pueden agrupar en un HBox. Este contiene la posición del grupo de botones, y el espacio entre estos.

```
        <children>
            <Button layoutX="99.0" layoutY="241.0" mnemonicParsing="false" style="-fx-background-color: green;" text="New" textFill="WHITE" />
```

Cada button tiene unos estilos diferentes, y una posición dentro del HBox.

```
            <Button layoutX="177.0" layoutY="241.0" mnemonicParsing="false" style="-fx-background-color: ;" text="Edit" />
            <Button layoutX="254.0" layoutY="241.0" mnemonicParsing="false" style="-fx-background-color: red;" text="Delete" textFill="WHITE" />
        </children>
    </HBox>
```

Previsualización de los Buttons:



5. Explicar què és l'arquitectura (o arquitectures...) MVC, per a què s'utilitza i com s'acostuma a implementar en Java. Explicar la jerarquia de paquets que s'utilitza en el projecte AddressApss. Per què no són tots els packages al mateix nivell?

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El modelo es el responsable de:
Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
Lleva un registro de las vistas y controladores del sistema.
Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

Jesús Serrano
17/01/2022

El controlador es responsable de:
Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las vistas son responsables de:
Recibir datos del modelo y los muestra al usuario.
Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

AdressApp separa el modelo y la view en dos carpetas diferentes. Después MainApp está fuera de estas carpetas a un nivel superior porque es quien las junta al iniciar la aplicacion.