Documentation 01

Project Structure

```
ImplementingACrispNavigation/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── ImplementingACrispNavigation/
│   │   │   │   ├── App.java
│   │   │   │   ├── controller/
│   │   │   │   │   └── StudentController.java
│   │   │   │   ├── model/
│   │   │   │   │   ├── Student.java
│   │   │   │   │   └── StudentModel.java
│   │   │   │   └── view/
│   │   │   │       ├── MainMenuView.java
│   │   │   │       ├── DataInputView.java
│   │   │   │       └── DisplayView.java
│   │   └── resources/
│   │       └── (fichiers de ressources si nécessaire)
│   └── test/
│       └── java/
│           └── ImplementingACrispNavigation/
│               ├── controller/
│               │   └── StudentControllerTest.java
│               ├── model/
│               │   ├── StudentTest.java
│               │   └── StudentModelTest.java
│               └── view/
│                   ├── MainMenuViewTest.java
│                   ├── DataInputViewTest.java
│                   └── DisplayViewTest.java
├── pom.xml
└── README.md
```

Problem Objective

The problem requires creating a Java Swing application that allows smooth navigation between multiple screens. The application must include a main menu that allows users to navigate to at least two sections: one section for data input (with various Swing components) and one section to display stored information. Transitions between screens should be seamless, and users should be able to navigate back and forth without losing data.

Proposed Solution

1. Model

- **Student**: Represents a student's information with attributes such as ID, name, department, and address.
- **StudentModel:** Manages the collection of students, providing methods to add, retrieve, update, and delete students.

2. Controller

- **StudentController:** Acts as an intermediary between the model and views. It handles user interactions and updates the views accordingly.

3. View

- **MainMenuView:** Main menu allowing navigation to data input and data display sections.
- **DataInputView:** User interface for inputting student information. Includes components like `JComboBox`, `JTextField`, `JTextArea`, `JRadioButton`, and buttons for adding data and returning to the main menu.
- **DisplayView:** User interface for displaying student information using a `JTree`.

4. Main Class

- **App:** Entry point of the application. Initializes the models, views, and controllers, and configures event listeners for navigation between views.

Code Functionality

Initialization:

- The `App` class initializes the models (**StudentModel**), views **(MainMenuView, DataInputView, DisplayView)**, and controller **(StudentController)**.

Navigation:

- The `MainMenuView` contains two buttons: one to access the data input section (`DataInputView`) and another to access the data display section (`DisplayView`).
- Event listeners are configured to show the respective views when a button is clicked.

Data Input:

- In `DataInputView`, the user can input student information. Data is captured through Swing components like `JTextField`, `JComboBox`, `JTextArea`, etc.
- When the user clicks the "Add" button, the information is sent to the controller (`StudentController`), which adds the student to the model (`StudentModel`).

Data Display:

- In **DisplayView,** student information is displayed in a `JTree`.
- The **DisplayView** is updated with the list of students each time it is shown.

Return to Main Menu:

- The "Back" buttons in `DataInputView` and `DisplayView` allow the user to return to the main menu without losing the entered data.

This solution meets the problem requirements by providing smooth and responsive navigation between multiple screens, using appropriate Swing components for data input and display, and implementing event-driven navigation.