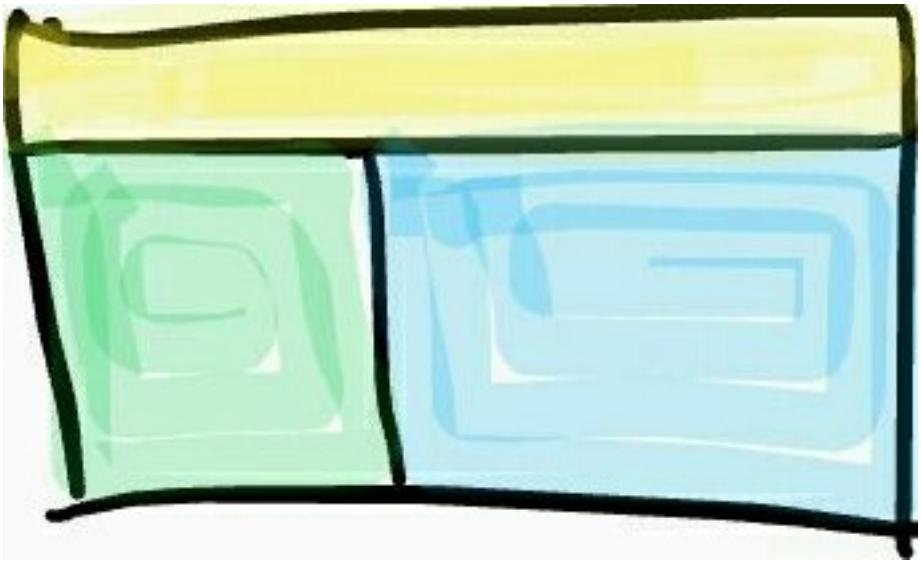
GUI 구현 상황

기존 CLI 구현 단계에서는 `Menu ->> Console ->> Service` 로 클래스 역할 분담을 하였었습니다. GUI 구현 단계에서는 기존의 클래스 역할 분담을 `(View + Window) ->> Controller ->> Service` 로 바꾸었습니다.

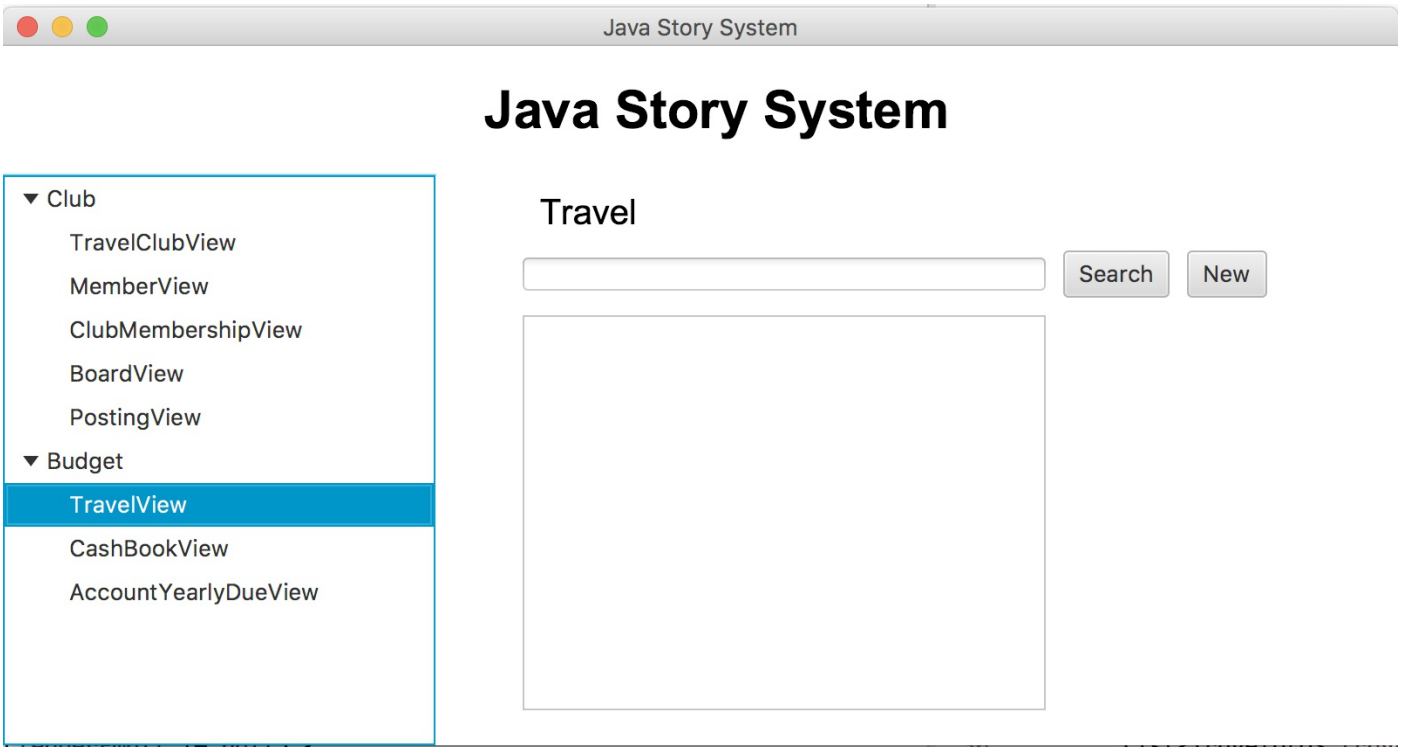
이미지를 참고해 말씀드리자면 아래 그림은 간략하게 View 클래스의 형태를 나타낸 것 입니다. 여기서 View클래스는 전체적인 틀을 구성하는 MainView 클래스를 의미합니다.

Main UI

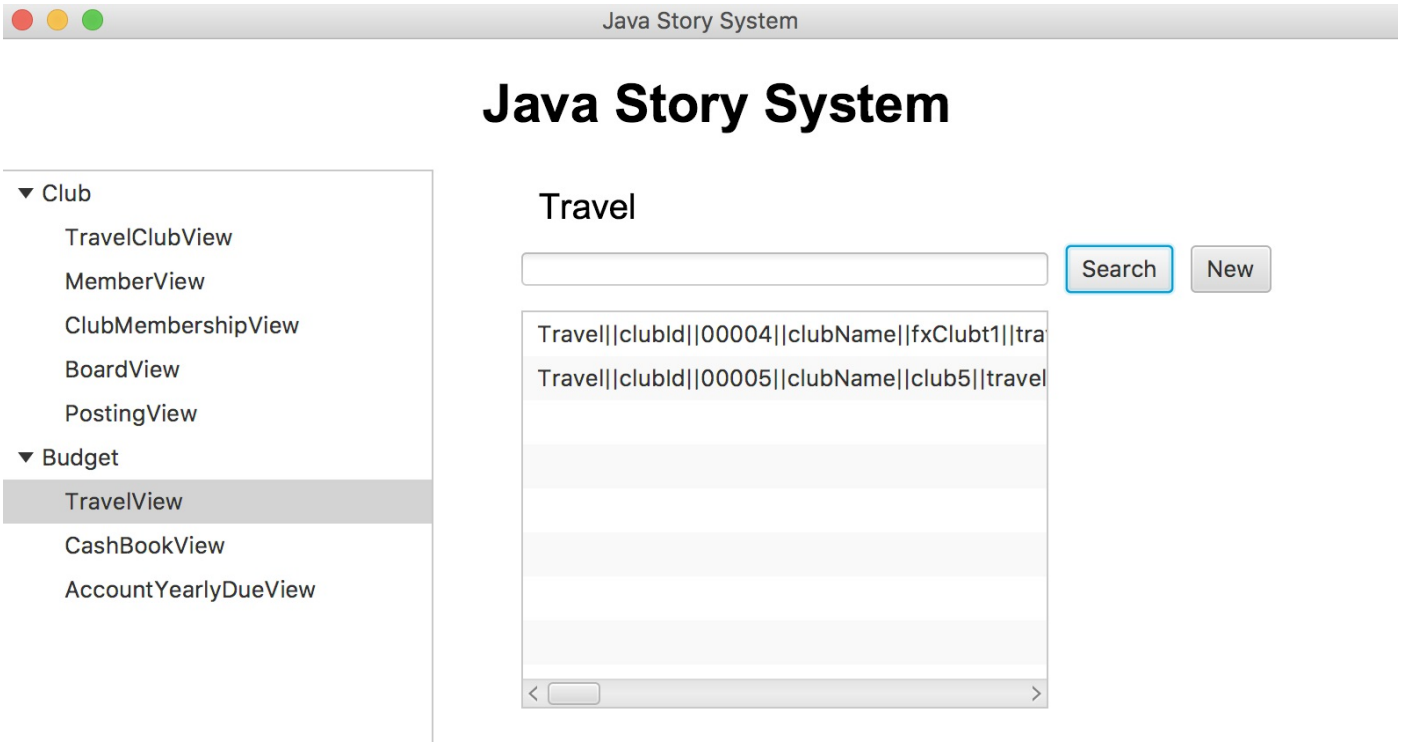


- 노란색 영역: Global Title
- 녹색 영역 : TreeView로 구성된 전체 메뉴(Global View)
- 파란색 영역 : GridPane으로 구성된 각 메뉴의 View

TravelView 클래스



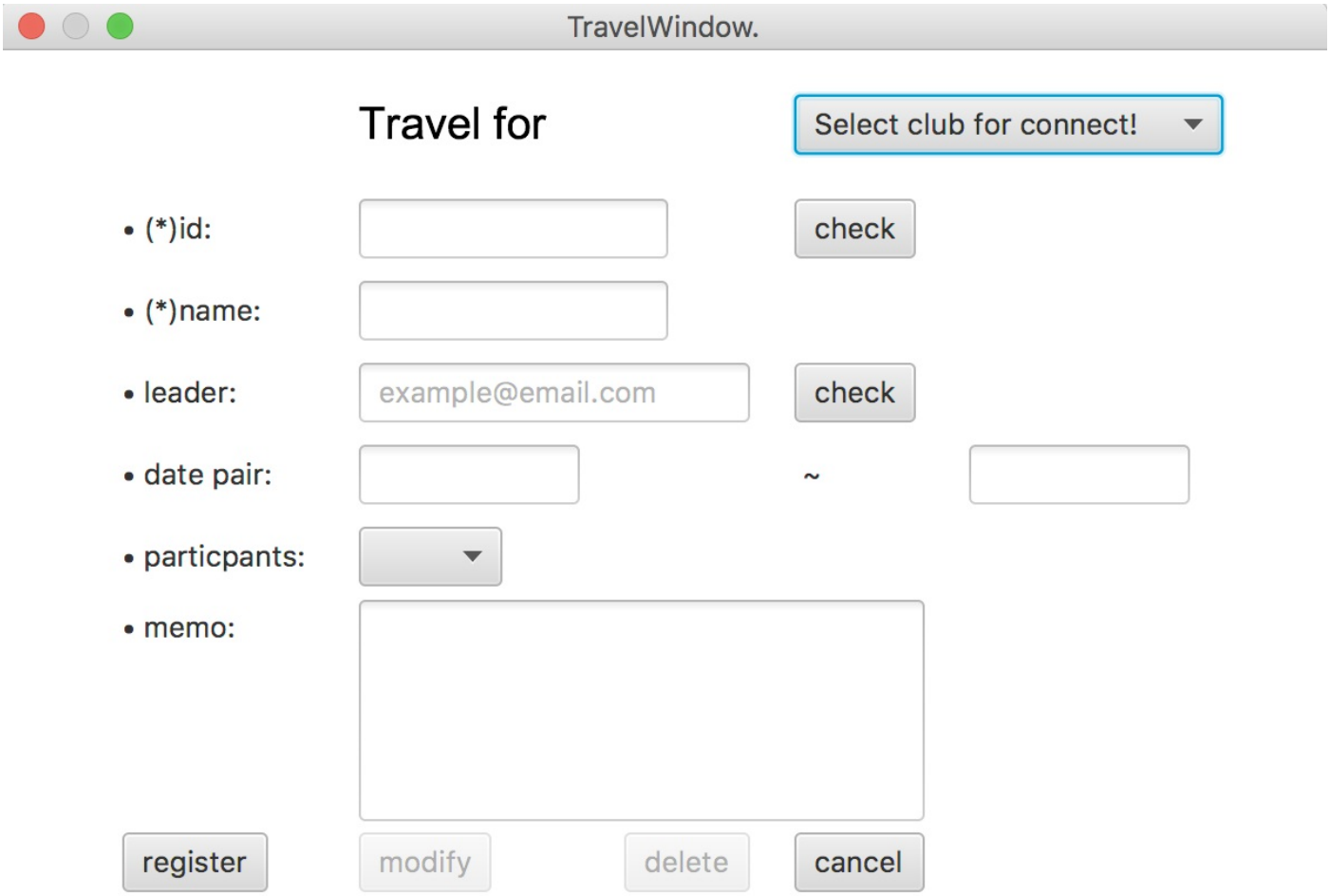
녹색영역의 TreeView에서 TravelView 메뉴를 클릭하면 나타나는 모습입니다. 나머지 다른 View들도 이와 같거나 비슷한 형태를 띄고 있습니다.



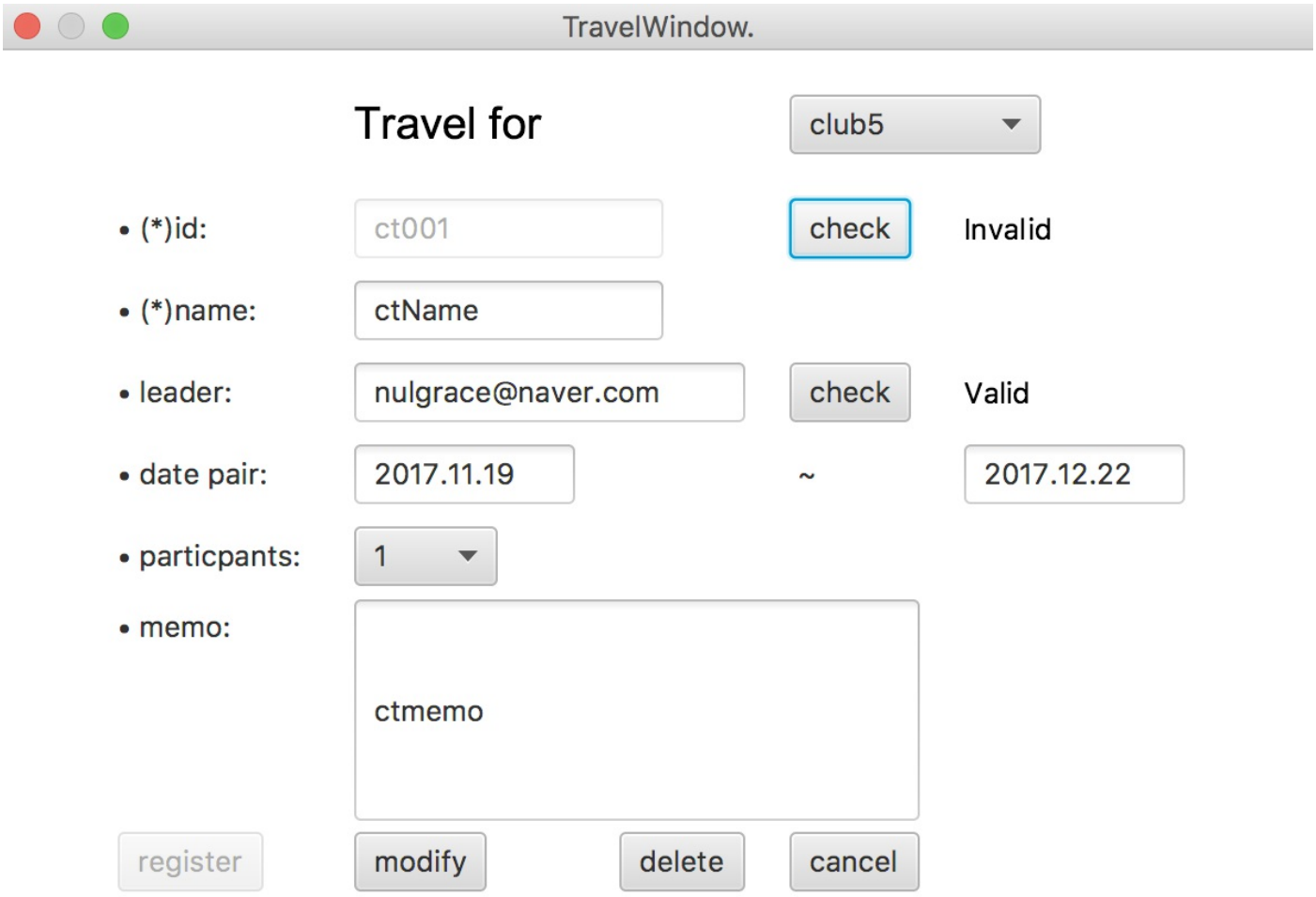
Search 버튼을 누렀을 때 검색 바에 입력된 Travel Name이 존재하면 존재하는 것들을 보여주고, 존재하지 않거나 아무것도 검색하지 않으면 전체 Travel들을 보여 줍니다.

TravelWindow 클래스

아래 이미지는 New 버튼을 눌렀을 때 나타나는 Register Mode 입니다.



아래 이미지는 TravelView의 리스트 뷰에서 검색되어진 Travel 중에 하나를 더블클릭했을 때, 나타나는 Modify Mode 입니다.



TravelWindow.java 소스코드

```
package javastory.fx.window;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javastory.budget.service.dto.TravelDto;
import javastory.budget.share.DatePair;
import javastory.budget.share.IdName;
import javastory.budget.share.Socialian;
import javastory.club.stage3.step4.service.dto.MemberDto;
import javastory.club.stage3.step4.service.dto.TravelClubDto;
```

```
import javastory.club.stage3.step4.ui.console.ClubConsole;
import javastory.club.stage3.step4.ui.console.MemberConsole;
import javastory.fx.controller.TravelController;
import javastory.fx.util.AlertBox;
import javastory.fx.util.ConfirmBox;

public class TravelWindow {
        //
        private String title;
        private TravelController travelController;
        private ComboBox<String> clubBox;
        private ComboBox<Integer> participantsCountBox;
        private TextField idInput, nameInput,
                        leaderInput, startDateInput, endDateInput, memoInput;
        private Button register, modify, delete, cancel;
        private Button idCheck, leaderCheck;
        private ClubConsole clubConsole;
        private MemberConsole memberConsole;
        private TravelDto currentTravel;
        private Text idValid, leaderValid;
        private IdName club;
        private boolean idChecked, emailChecked;

        public TravelWindow(String title, TravelController travelController) {
                //
                this.title = title;
                this.travelController = travelController;
                clubConsole = new ClubConsole();
                memberConsole = new MemberConsole();
                clubBox = new ComboBox<>();
                participantsCountBox = new ComboBox<>();
                idInput = new TextField();
                nameInput = new TextField();
                leaderInput = new TextField();
                startDateInput = new TextField();
                endDateInput = new TextField();
                memoInput = new TextField();
                register = new Button("register");
                modify = new Button("modify");
                delete = new Button("delete");
                cancel = new Button("cancel");
                idCheck = new Button("check");
                leaderCheck = new Button("check");
                idValid = new Text();
                leaderValid = new Text();
        }

        private void init() {
                //
                Stage window = new Stage();

                window.initModality(Modality.APPLICATION_MODAL);
                window.setTitle(title);
                window.setMinWidth(300);

                GridPane grid = new GridPane();
                grid.setStyle("-fx-background-color: #FFFFFF;");
                grid.setVgap(10);
```

```
        grid.setHgap(20);
        grid.setPadding(new Insets(20,20,20,55));

        //Headline
        Text headLine = new Text("Travel for ");
        headLine.setFont(Font.font("Arial", 20));
        grid.add(headLine, 1, 0, 2, 2);

        //clubBox
        clubBox.setPromptText("Select club for connect!");
        this.getClubItems(clubBox);
        this.checkClubBoxValue(clubBox);
        grid.add(clubBox, 3, 0, 2, 2);

        //q1 label
        Label q1 = new Label();
        q1.setText("• (*)id: ");
        q1.setAlignment(Pos.CENTER_RIGHT);
        grid.add(q1, 0, 3);

        //id Input
        idInput.setMaxWidth(140);
        grid.add(idInput, 1, 3, 2, 1);

        //id duplication check button
        grid.add(idCheck, 3, 3);
        idCheck.setOnAction(e -> this.checkIdDuplication());

        //id valid check text
        grid.add(idValid, 4, 3);

        //q2 label
        Label q2 =  new Label();
        q2.setText("• (*)name: ");
        q2.setAlignment(Pos.CENTER_RIGHT);
        grid.add(q2, 0, 4);

        //name input
        nameInput.setMaxWidth(140);
        grid.add(nameInput, 1, 4, 2, 1);

        //q3 label
        Label q3 = new Label();
        q3.setText("• leader: ");
        q3.setAlignment(Pos.CENTER_RIGHT);
        grid.add(q3, 0, 5);

        //leader input
        leaderInput.setPromptText("example@email.com");
        leaderInput.setMaxWidth(200);
        grid.add(leaderInput, 1, 5, 2, 1);

        //leader valid check button
        grid.add(leaderCheck, 3, 5);
        leaderCheck.setOnAction(e -> this.checkLeaderValid());

        //leader valid check text
        grid.add(leaderValid, 4, 5);
```

```
                //q4 label
                Label q4 = new Label();
                q4.setText("• date pair: ");
                q4.setAlignment(Pos.CENTER_RIGHT);
                grid.add(q4, 0, 6);

                //startDate input
                startDateInput.setMaxWidth(100);
                grid.add(startDateInput, 1, 6);

                // ~ text
                Label waveMark = new Label();
                waveMark.setText(" ~ ");
                grid.add(waveMark, 3, 6);

                //endDate input
                endDateInput.setMaxWidth(100);
                grid.add(endDateInput, 4, 6);

                //q5 label
                Label q5 = new Label();
                q5.setText("• particpants: ");
                q5.setAlignment(Pos.CENTER_RIGHT);
                grid.add(q5, 0, 7);

                //particpatns count input
                this.setParticipatnsCountBox();
//              particpantsCountBox.setEditable(true);
                grid.add(participantsCountBox, 1, 7);

                //q6 label
                Label q6 = new Label();
                q6.setText("• memo: ");
                q6.setAlignment(Pos.CENTER_RIGHT);
                grid.add(q6, 0, 8);

                //memo input
                memoInput.setMinHeight(100);
                grid.add(memoInput, 1, 8, 3, 3);

                //register button
                grid.add(register, 0, 11);
                register.setOnAction(e -> {
                        if (!idChecked) {
                                AlertBox.display("Duplication check Warning", "No ID
duplication check has been performed.");
                                return;
                        }
                        this.setCurrentTravel();
                        if (currentTravel != null) {
                                travelController.register(currentTravel);
                        }
                        window.close();
                });

                //modify button
                grid.add(modify, 1, 11);
```

```
                modify.setOnAction(e -> {
                        if(!emailChecked) {
                                AlertBox.display("Duplication check Warning", "No
duplicate checks were performed.");
                                return;
                        }
                        // Check if it has changed.
                        this.checkModified();
                        window.close();
                });

                //delete button
                grid.add(delete, 2, 11);
                delete.setOnAction(e -> {
                        boolean result = ConfirmBox.display("Remove Travel", "Are you
sure you want to remove this membership?");
                        if (result) {
                                travelController.delete(currentTravel);
                                window.close();
                        }
                });

                //cancel button
                grid.add(cancel, 3, 11);
                cancel.setOnAction(e -> {
                        window.close();
                });

                Scene scene = new Scene(grid, 600, 400);
                window.setScene(scene);
                window.showAndWait();
        }

        private void checkModified() {
                //
                if (currentTravel.getId().equals(idInput.getText())
                        && currentTravel.getName().equals(nameInput.getText())
                        &&
currentTravel.getLeader().getEmail().equals(leaderInput.getText())
                        &&
currentTravel.getTravelTerm().getStartDate().equals(startDateInput.getText())
                        &&
currentTravel.getTravelTerm().getEndDate().equals(endDateInput.getText())
                        && (currentTravel.getParticipantsCount() ==
participantsCountBox.getValue())
                        && currentTravel.getMemo().equals(memoInput.getText())){
                        return;
                }
                travelController.modify(currentTravel);
        }

        private void setCurrentTravel() {
                //check required items
                if (club == null || idInput.getText() == null || nameInput.getText()
== null) {
                        AlertBox.display("Required items Warning", "You did not meet
the required entries.");
                        return;
```

```
                }
                IdName travel = new IdName(idInput.getText(), nameInput.getText());

                //create currentTravel
                currentTravel = new TravelDto(club, travel);

                //check leader input
                String leaderEmail = null;
                if (leaderInput.getText() != null) {
                        leaderEmail = leaderInput.getText();
                }
                MemberDto memberDto = null;
                if (leaderEmail != null) {
                        memberDto = memberConsole.searchByEmail(leaderEmail);
                }
                Socialian leader = null;
                if (memberDto != null) {
                        leader = new Socialian(memberDto.getBirthDay(),
memberDto.getNickName(),
                                        memberDto.getName(), memberDto.getEmail());
                }
                //set leader
                currentTravel.setLeader(leader);

                // check startDate input
                String startDate = null;
                if (startDateInput.getText() == null) {
                        startDate =
LocalDate.now().format(DateTimeFormatter.ISO_LOCAL_DATE);
                }
                // check endDate input
                String endDate = null;
                if (endDateInput.getText() == null) {
                        endDate = "";
                }
                DatePair travelTerm = new DatePair(startDate, endDate);
                // set travelTerm
                currentTravel.setTravelTerm(travelTerm);

                //check participants count
                int participantsCount = 0;
                if (participantsCountBox.getValue() == null) {
                        participantsCount = 1;
                }
                // set participants count
                currentTravel.setParticipantsCount(participantsCount);

                // check memo input
                String memo = null;
                if (memoInput.getText() == null) {
                        memo = "";
                }
                currentTravel.setMemo(memo);
        }

        private void setParticipatnsCountBox() {
                //
                for (int count = 1; count < 21; count++) {
```

```
                        participantsCountBox.getItems().add(count);
                }
        }

        private void checkLeaderValid() {
                //
                MemberDto member =
memberConsole.searchByEmail(leaderInput.getText());
                if (member != null) {
                        leaderValid.setText("Valid");
                        leaderValid.setStyle("-fx-text-fill: blue;"); //doesn't work
                        emailChecked = true;
                } else {
                        leaderValid.setText("Invalid");
                        idValid.setStyle("-fx-text-fill: #FF0000;");
                        emailChecked = false;
                }
        }

        private void checkIdDuplication() {
                //
                boolean isExist = travelController.exists(idInput.getText());
                if (!isExist) {
                        idValid.setText("Valid");
                        idValid.setStyle("-fx-text-fill: #0000FF;");
                        idChecked = true;
                } else {
                        idValid.setText("Invalid");
                        idValid.setStyle("-fx-text-fill: #FF0000;");
                        idChecked = false;
                }
        }

        private void checkClubBoxValue(ComboBox<String> clubBox) {
                //
                clubBox.getSelectionModel().selectedItemProperty().addListener((v,
oldValue, newValue) -> {
                        if (newValue == null) {
                                club = null;
                                return;
                        }
                        String clubId = clubConsole.searchByName(newValue).getUsid();
                        club = new IdName(clubId, newValue);
                });

        }

        private void getClubItems(ComboBox<String> clubBox) {
                //
                List<TravelClubDto> clubList = clubConsole.findAll();
                for (TravelClubDto club : clubList) {
                        clubBox.getItems().add(club.getName());
                }
        }

        public void registerMode() {
                //
                this.fieldClear();
```

```
                modify.setDisable(true);
                delete.setDisable(true);
                init();
        }

        public void modifyMode(TravelDto targetTravel) {
                //
                this.fieldClear();
                register.setDisable(true);
                clubBox.setValue(targetTravel.getClubName());
                idInput.setText(targetTravel.getId());
                idInput.setDisable(true);
                nameInput.setText(targetTravel.getName());
                leaderInput.setText(targetTravel.getLeader().getEmail());
                startDateInput.setText(targetTravel.getTravelTerm().getStartDate());
                endDateInput.setText(targetTravel.getTravelTerm().getEndDate());
                participantsCountBox.setValue((targetTravel.getParticipantsCount()));
                memoInput.setText(targetTravel.getMemo());
                currentTravel = targetTravel;
                init();
        }

        private void fieldClear() {
                //
                register.setDisable(false);
                modify.setDisable(false);
                delete.setDisable(false);
                idInput.setDisable(false);
                idInput.setText(null);
                nameInput.setText(null);
                leaderInput.setText(null);
                startDateInput.setText(null);
                endDateInput.setText(null);
                participantsCountBox.setValue(null);
                memoInput.setText(null);
                clubBox.setValue(null);
                clubBox.setPromptText("Select club for connect!");
                idValid.setText(null);
                leaderValid.setText(null);
        }
}
```

제가 질문을 드린 이유와 제가 이 자료를 만들게 된 이유가 여기에 있습니다. 이전 다른 Window 클래스들을 구현할 때는 라인 수가 최대 207라인이였습니다. 그런데 위 TravelWindow 클래스는 총 라인수가 399라인 이나 됩니다. 위 메소드 중에 Controller 클래스로 옮겨야 하는 메소드를 판별하려 했지만 잘 모르겠습니다.

TravelController 클래스

```
package javastory.fx.controller;

import java.util.List;

import javastory.budget.logic.ServiceLogicLycler;
import javastory.budget.service.TravelService;
import javastory.budget.service.dto.TravelDto;
import javastory.budget.util.NoSuchTravelException;
```

```
import javastory.budget.util.TravelDuplicationException;

public class TravelController {
	//
	private TravelService travelService;

	public TravelController() {
		//
		travelService =
ServiceLogicLycler.shareInstance().createTravelService();
	}

	public boolean exists(String travelId) {
		//
		return travelService.exist(travelId);
	}

	public void register(TravelDto currentTravel) {
		//
		try {
			travelService.createTravel(currentTravel);
		} catch (TravelDuplicationException e) {
			System.out.println(e.getMessage());
		}
	}

	public List<TravelDto> searchByName(String travelName) {
		//
		List<TravelDto> travelDtos = null;
		try {
			travelDtos = travelService.findTravelByName(travelName);
		} catch (NoSuchTravelException e) {
			System.out.println(e.getMessage());
		}
		return travelDtos;
	}

	public List<TravelDto> findAll() {
		//
		List<TravelDto> travelDtos = null;
		try {
			travelDtos = travelService.findAll();
		} catch (NoSuchTravelException e) {
			System.out.println(e.getMessage());
		}
		return travelDtos;
	}

	public void modify(TravelDto currentTravel) {
		//
		try {
			travelService.modify(currentTravel);
		} catch (NoSuchTravelException e) {
			System.out.println(e.getMessage());
		}
	}

	public void delete(TravelDto currentTravel) {
```

```
                //
                try {
                        travelService.delete(currentTravel.getId());
                } catch (NoSuchTravelException e) {
                        System.out.println(e.getMessage());
                }
        }
}
```

일단은 이렇게 Window 클래스와 Controller 클래스의 역할을 구분하였습니다. 어떻게 하면 두 클래스의 역할을 다시 재조정하면서 코드를 정리할 수 있을 지 모르겠습니다. 제가 어떠한 점을 잘못하고 있는지 알려주시면 감사 드리겠습니다.

etc. 나머지 소스코드는 github/namoosori 에 업로드 되어있습니다.