John Høeg Semester 1 prøve PRO1

GUI – App

```
package gui;

import controller.Controller;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.TabPane;
import javafx.scene.control.TabPane.TabClosingPolicy;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import java.util.ArrayList;
import java.util.Optional;
import javafx.beans.value.ChangeListener;
import javafx.collections.ListChangeListener;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.geometry.VPos;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.CheckBox;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.SelectionMode;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;

import model.Bil;
import model.Booking;
import model.Lift;
import model.Stjerner;
import storage.Storage;

public class App extends Application {

            public static void main(String[] args) {
                        Application.launch(args);
            }

            // Opgave 10
            @Override
            public void init() {
```

```
                        Controller.initStorage();
                        Controller.createAnbefaling(Stjerner.TO, "Han koerer for
hurtigt!");
            }

            @Override
            public void start(Stage stage) {
                        stage.setTitle("Samkoersel");
                        GridPane pane = new GridPane();
                        this.initContent(pane);

                        Scene scene = new Scene(pane);
                        stage.setScene(scene);
                        stage.show();
            }

            // ------------------------------------------------------------------
------

            // Attributes

            private Label lblUdbudteLift, lblpassager, lblopsamlingssted,
lblopsamlingstid, lblbookinger;
            private TextField txtdatoPaaLift, txtpassager, txtopsamlingssted,
txtopsamlingstid, txtbookinger;
            private Button btVisLift, btOpretBooking;
            private ListView<Booking> lvwBookings;
            private ListView<Lift> lvwListAfLift;
            private Lift lift;
            private Bil bil;
            private Booking booking;

            // ------------------------------------------------------------------
------

            // Opgave 11
            private void initContent(GridPane pane) {
                        pane.setGridLinesVisible(false);
                        pane.setPadding(new Insets(10));
                        pane.setHgap(10);
                        pane.setVgap(10);
                        pane.setStyle("-fx-border-color: black");

                        lblUdbudteLift = new Label("Udbudte lift paa dato:");
                        pane.add(lblUdbudteLift, 0, 0);
                        lblUdbudteLift.setPrefWidth(125);
                        txtdatoPaaLift = new TextField();
                        pane.add(txtdatoPaaLift, 1, 0);
                        btVisLift = new Button("Vis lift");
                        pane.add(btVisLift, 2, 0);
                        btVisLift.setPrefWidth(50);
                        btVisLift.setOnAction(event -> getLift());

                        lvwListAfLift = new ListView();
                        pane.add(lvwListAfLift, 0, 1, 4, 7);
                        lvwListAfLift.setPrefWidth(200);

                        lblpassager = new Label("Passager:");
                        pane.add(lblpassager, 4, 2);
```

```
                        txtpassager = new TextField();
                        pane.add(txtpassager, 5, 2);

                        lblopsamlingssted = new Label("Opsamlingssted:");
                        pane.add(lblopsamlingssted, 4, 3);
                        txtopsamlingssted = new TextField();
                        pane.add(txtopsamlingssted, 5, 3);
                        lblopsamlingssted.setPrefWidth(100);

                        lblopsamlingstid = new Label("Opsamlings tid:");
                        pane.add(lblopsamlingstid, 4, 4);
                        txtopsamlingstid = new TextField();
                        pane.add(txtopsamlingstid, 5, 4);

                        btOpretBooking = new Button("Opret Booking");
                        pane.add(btOpretBooking, 5, 5);
                        btOpretBooking.setOnAction(event -> createBooking());

                        lblbookinger = new Label("Bookinger:");
                        pane.add(lblbookinger, 4, 6);
                        lvwBookings = new ListView();
                        pane.add(lvwBookings, 4, 7, 2, 1);
                        lvwBookings.setPrefWidth(200);
                        lvwBookings.getItems().setAll(Storage.getBooking());

            }

            private void getLift() {
                        LocalDateTime liftTidspunkt =
LocalDateTime.parse(txtdatoPaaLift.getText().trim(), formatter2);
                        for (int i = 0; i < bil.getLs().size(); i++) {
                                    if (bil.getLs().get(i).getTidspunkt() ==
liftTidspunkt) {
                                                ArrayList<Lift> tempLift = new
ArrayList<>();
                                                for (int j = 0; j <
bil.getLs().size(); j++) {

            tempLift.add(bil.getLs().get(j));
                                                }

            lvwListAfLift.getItems().setAll(tempLift);
                                    }
                        }


            }

            DateTimeFormatter formatter = DateTimeFormatter.ISO_LOCAL_TIME;
            DateTimeFormatter formatter2 = DateTimeFormatter.ISO_LOCAL_DATE_TIME;

            private void createBooking() {
                        String passagerNavn = txtpassager.getText().trim();
                        String opsamlingssted =
txtopsamlingssted.getText().trim();
                        LocalTime opsamlingstid =
LocalTime.parse(txtopsamlingstid.getText().trim(), formatter);
                        if (passagerNavn.length() != 0 && opsamlingssted.length()
!= 0) {
```

```
                                    Booking bookinger =
Controller.createBooking(passagerNavn, opsamlingssted, opsamlingstid,

            lvwListAfLift.getSelectionModel().getSelectedItem());

                        }
                        lvwBookings.getItems().setAll(Storage.getBooking());
                        clearFields();
            }

            private void clearFields() {
                        txtpassager.clear();
                        txtopsamlingssted.clear();
                        txtopsamlingstid.clear();
            }

}
```

Controller – Controller

```
package controller;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;

import model.Anbefaling;
import model.Bil;
import model.Booking;
import model.Lift;
import model.Stjerner;
import storage.Storage;

public class Controller {

            // create
            public static Bil createBil(String regNr, String chauffoer) {
                        Bil bil = new Bil(regNr, chauffoer);
                        Storage.addBiler(bil);
                        return bil;

            }

            public static Lift createLift(LocalDateTime tidspunkt, int pris,
String fraAdresse, String tilAdresse,
                                    int maxAntalPassager, Bil bil) {
                        Lift lift = new Lift(tidspunkt, pris, fraAdresse,
tilAdresse, maxAntalPassager, bil);
                        return lift;
            }

            public static Booking createBooking(String passager, String
opsamlingssted, LocalTime opsamlingstid, Lift lift) {
                        try {
                        if (lift.Ledigplads()) {
                                    Booking booking = new Booking(passager,
opsamlingssted, opsamlingstid, lift);
```

```
                                    Storage.addBookinger(booking);
                                    return booking;

                        }
                        } catch (RuntimeException e) {
                                    System.out.println("Ingen ledige pladser");
                        }
                        return null;

            }


            public static Anbefaling createAnbefaling(Stjerner stjerner, String
kommentar) {
                        Anbefaling anbefaling = new Anbefaling(stjerner,
kommentar);
                        return anbefaling;
            }

            //Opgave 7
            public static Lift liftPaaDato(LocalDate dato) {

                        return null;
            }

            //Opgave 8
            public static int anbefalingsFrekvens(ArrayList<Anbefaling>
anbefalinger, Stjerner stjerner) {
                        int antalStjerner = 0;
//                      for (int i = 0; i < anbefalinger.size(); i++) {
//                                  if (i = 0) {
//                                              System.out.println(stjerner.EN);
//                                  }
//                      }

                        return antalStjerner;
            }

            //Opgave 9
            public static String findChauffoerOgAnbefaling(ArrayList<Booking>
booking, String passager, String opsamlingssted, LocalTime opsamlingstid) {
                        boolean found = false;
        int i = 0;
        while (!found && i < booking.size()) {
            if (booking.get(i).equals(passager) &&
booking.get(i).equals(opsamlingstid) && booking.get(i).equals(opsamlingssted)) {
                        found = true;
            } else {
                        i++;
            }
        }
        if (found) {
            return booking.get(i).getAf().toString();
        } else {
            return null;
        }
            }

            // delete
```

```
            // update

            // get


            public static void initStorage() {

                        //Creater biler
                        Bil b1 = Controller.createBil("AB 12 345", "Jane Jensen");
                        Bil b2 = Controller.createBil("XY 98 765", "Lene Hansen");

                        //Adder Lift
                        LocalDateTime dt1 = LocalDateTime.of(2019, 6, 22, 10, 00);
                        LocalDateTime dt2 = LocalDateTime.of(2019, 6, 24, 20, 00);
                        LocalDateTime dt3 = LocalDateTime.of(2019, 6, 28, 10, 00);
                        Lift l1 = Controller.createLift(dt1, 90, "Aarhus",
"Odense", 3, b1);
                        Lift l2 = Controller.createLift(dt2, 80, "Odense",
"Aarhus", 2, b1);
                        Lift l3 = Controller.createLift(dt3, 85, "Aarhus",
"Odense", 1, b1);
                        LocalDateTime dt4 = LocalDateTime.of(2019, 6, 24, 20, 30);
                        Lift l4 = Controller.createLift(dt4, 100, "Aarhus",
"Odense", 3, b2);

                        //Adder Booking
                        LocalTime lt1 = LocalTime.of(10, 00);
                        LocalTime lt2 = LocalTime.of(20, 15);
                        LocalTime lt3 = LocalTime.of(10, 10);
                        Booking bo1 = Controller.createBooking("Mette", "Aros",
lt1, l1);
                        Booking bo2 = Controller.createBooking("Karina",
"Banegaarden", lt2, l2);
                        Booking bo3 = Controller.createBooking("Camilla",
"Musikhuset", lt3, l3);

            }

}
```

Storage – Storage

```
package storage;

import java.util.ArrayList;

import model.Bil;
import model.Booking;


public class Storage {

            private static ArrayList<Bil> biler = new ArrayList<>();
            private static ArrayList<Booking> bookinger = new ArrayList<>();

            //GET metode
```

```
	public static ArrayList<Bil> getBil() {
		return new ArrayList<Bil>(biler);
	}

	public static ArrayList<Booking> getBooking() {
		return new ArrayList<Booking>(bookinger);
	}

	//ADD metode
	public static void addBiler(Bil bil) {
		biler.add(bil);
	}

	public static void addBookinger(Booking booking) {
		bookinger.add(booking);
	}

	//Remove metode

}
```

Model – Bil

```
package model;

import java.time.LocalDateTime;
import java.util.ArrayList;

public class Bil {

	private String regNr;
	private String chauffoer;
	private ArrayList<Lift> ls = new ArrayList<>();

	public Bil(String regNr, String chauffoer) {
		this.chauffoer = chauffoer;
		this.regNr = regNr;
	}



	@Override
	public String toString() {
		return this.chauffoer;
	}



	public Lift createLift(LocalDateTime tidspunkt, int pris, String
fraAdresse, String tilAdresse,
				int maxAntalPassager, Bil bil) {
		Lift newLift = new Lift(tidspunkt, pris, fraAdresse,
tilAdresse, maxAntalPassager, bil);
		ls.add(newLift);
		return newLift;
	}
```

```
	public void removeLift(Lift newLift) {
		if (ls.contains(newLift)) {
			ls.remove(newLift);
			newLift.removeBil();
		}
	}

	public String getRegNr() {
		return regNr;
	}

	public String getChauffoer() {
		return chauffoer;
	}

	public ArrayList<Lift> getLs() {
		return ls;
	}

	// - - - - - - - - - - - - - - - - - - - - -
	// Opgave 3

	public int indtægt() {
		int pris = 0;
		for (int i = 0; i < ls.size(); i++) {
			pris = pris + ls.get(i).getPris();
		}
		return pris;
	}

	// - - - - - - - - - - - - - - - - - - - - -

}
```

Model – Lift

```
package model;

import java.time.LocalDateTime;
import java.util.ArrayList;

public class Lift {

	private LocalDateTime tidspunkt;
	private int pris;
	private String fraAdresse;
	private String tilAdresse;
	private int maxAntalPassager;
	private Bil bil;
	private ArrayList<Booking> bs = new ArrayList<>();

	public Lift(LocalDateTime tidspunkt, int pris, String fraAdresse,
String tilAdresse, int maxAntalPassager,
			Bil bil) {
		super();
```

```
                this.tidspunkt = tidspunkt;
                this.pris = pris;
                this.fraAdresse = fraAdresse;
                this.tilAdresse = tilAdresse;
                this.maxAntalPassager = maxAntalPassager;
                this.bil = bil;
        }

        public void addBooking(Booking booking) {
                if (!bs.contains(booking)) {
                        bs.add(booking);
                        booking.setLift(this);
                }
        }

        public void removeBooking(Booking booking) {
                if (bs.contains(booking)) {
                        bs.remove(booking);
                        booking.removeLift();
                }
        }

        public void removeBil() {
                this.bil = null;
        }

        public LocalDateTime getTidspunkt() {
                return tidspunkt;
        }

        public int getPris() {
                return pris;
        }

        public String getFraAdresse() {
                return fraAdresse;
        }

        public String getTilAdresse() {
                return tilAdresse;
        }

        public int getMaxAntalPassager() {
                return maxAntalPassager;
        }

        public Bil getBil() {
                return bil;
        }

        // - - - - - - - - - - - - - - - - - - - - - -
        // Opgave 2
        public boolean Ledigplads() {
                boolean plads = false;
                if (this.maxAntalPassager <= bs.size()) {
                        plads = false;
                } else {
                        plads = true;
                }
```

```
                return plads;
        }

        // - - - - - - - - - - - - - - - - - - - - - - -
        // Opgave 4

        public interface Comparable<T> {
                public int compareTo(T o);
        }

        // - - - - - - - - - - - - - - - - - - - - - - -

}
```

Model – Booking

```
package model;

import java.time.LocalTime;
import java.util.ArrayList;

public class Booking {

        private String passager;
        private String opsamlingssted;
        private LocalTime opsamlingstid;
        private Lift lift;
        private ArrayList<Anbefaling> af = new ArrayList<>();

        public Booking(String passager, String opsamlingssted, LocalTime
opsamlingstid, Lift lift) {
                super();
                this.passager = passager;
                this.opsamlingssted = opsamlingssted;
                this.opsamlingstid = opsamlingstid;
                this.lift = lift;
        }

        @Override
        public String toString() {
                return this.passager + " " + this.opsamlingssted + " " +
this.opsamlingstid;
        }

        public Anbefaling createAnbefaling(Stjerner stjerner, String
kommentar) {
                Anbefaling anbefaling = new Anbefaling(stjerner,
kommentar);
                af.add(anbefaling);
                return anbefaling;
        }

        public void removeAnbefaling(Anbefaling anbefaling) {
                if (af.contains(anbefaling)) {
                        af.remove(anbefaling);
                }
```

```
	}

	public void setLift(Lift lf) {
		if (this.lift == null) {
			this.lift = lf;
			lf.addBooking(this);
		} else if (lf == null) {
			this.lift.removeBooking(this);
			this.lift = lf;
		} else if (this.lift != lf) {
			this.lift.removeBooking(this);
			this.lift = lf;
			lf.addBooking(this);
		}
	}

	public void removeLift() {
		this.lift = null;
	}

	public String getPassager() {
		return passager;
	}

	public String getOpsamlingssted() {
		return opsamlingssted;
	}

	public LocalTime getOpsamlingstid() {
		return opsamlingstid;
	}

	public Lift getLift() {
		return lift;
	}

	public ArrayList<Anbefaling> getAf() {
		return af;
	}

}
```

Model – Anbefaling

```
package model;

public class Anbefaling {

	private Stjerner stjerner;
	private String kommentar;


	public Anbefaling(Stjerner stjerner, String kommentar) {
		super();
		this.stjerner = stjerner;
		this.kommentar = kommentar;
```

```
		}

		public Stjerner getStjerner() {
			return stjerner;
		}

		public String getKommentar() {
			return kommentar;
		}




}
```

Model – Stjerner

```
package model;

public enum Stjerner {

		EN, TO, TRE, FIRE, FEM

}
```