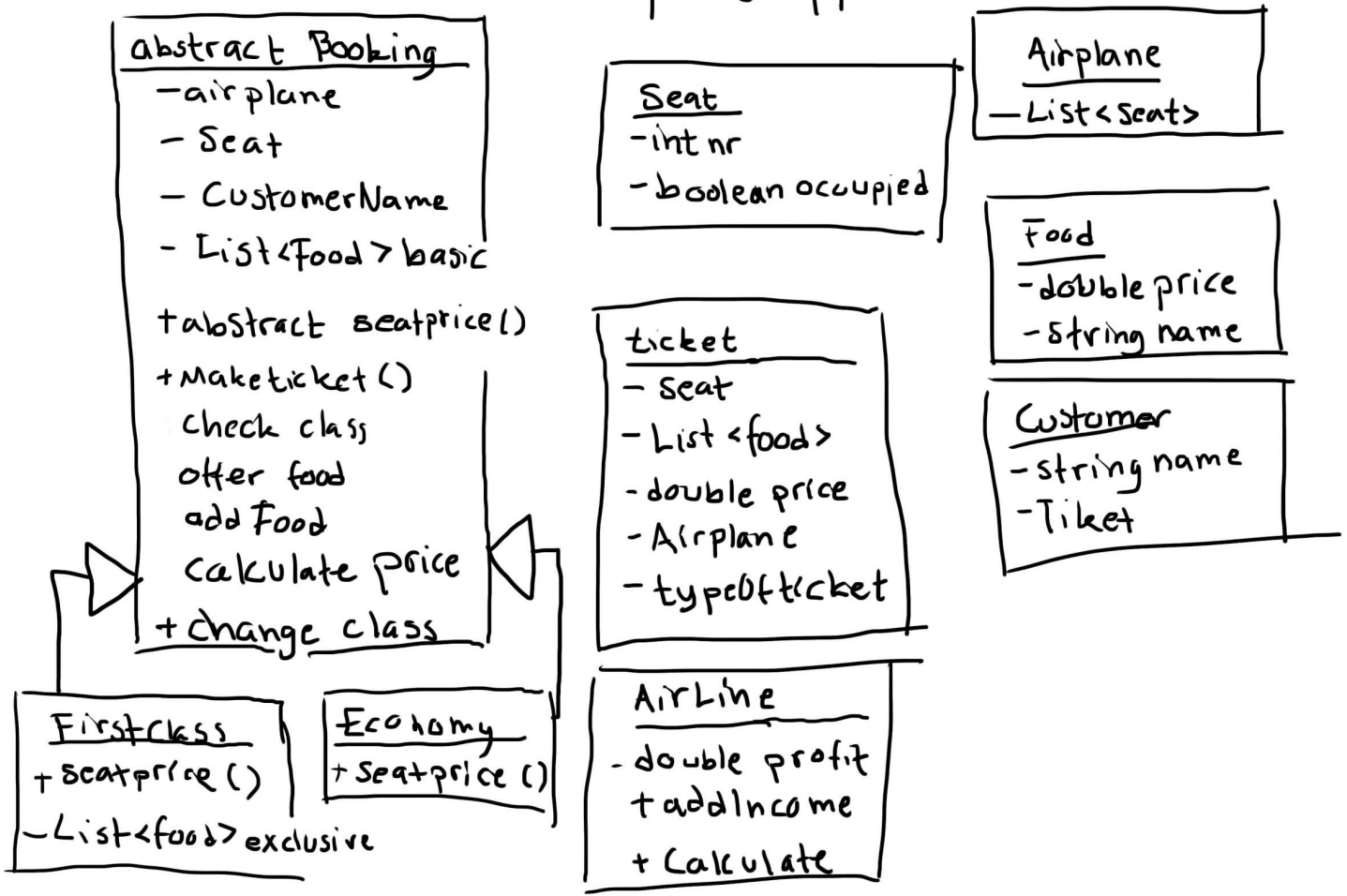
Airlines Application

Tony, Maria, Surekha

Airplane app



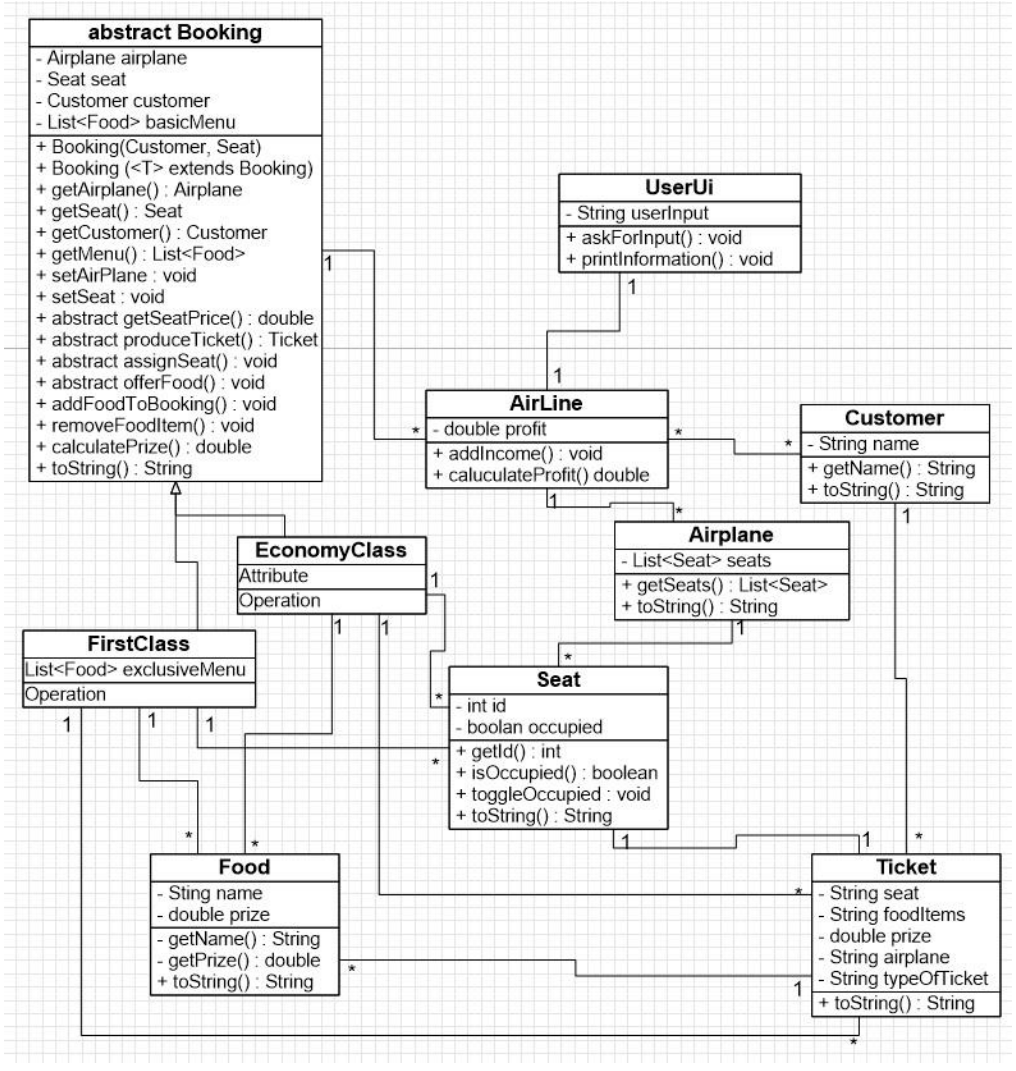
Main

1. Name
2. First or economy?
3. Free seats?

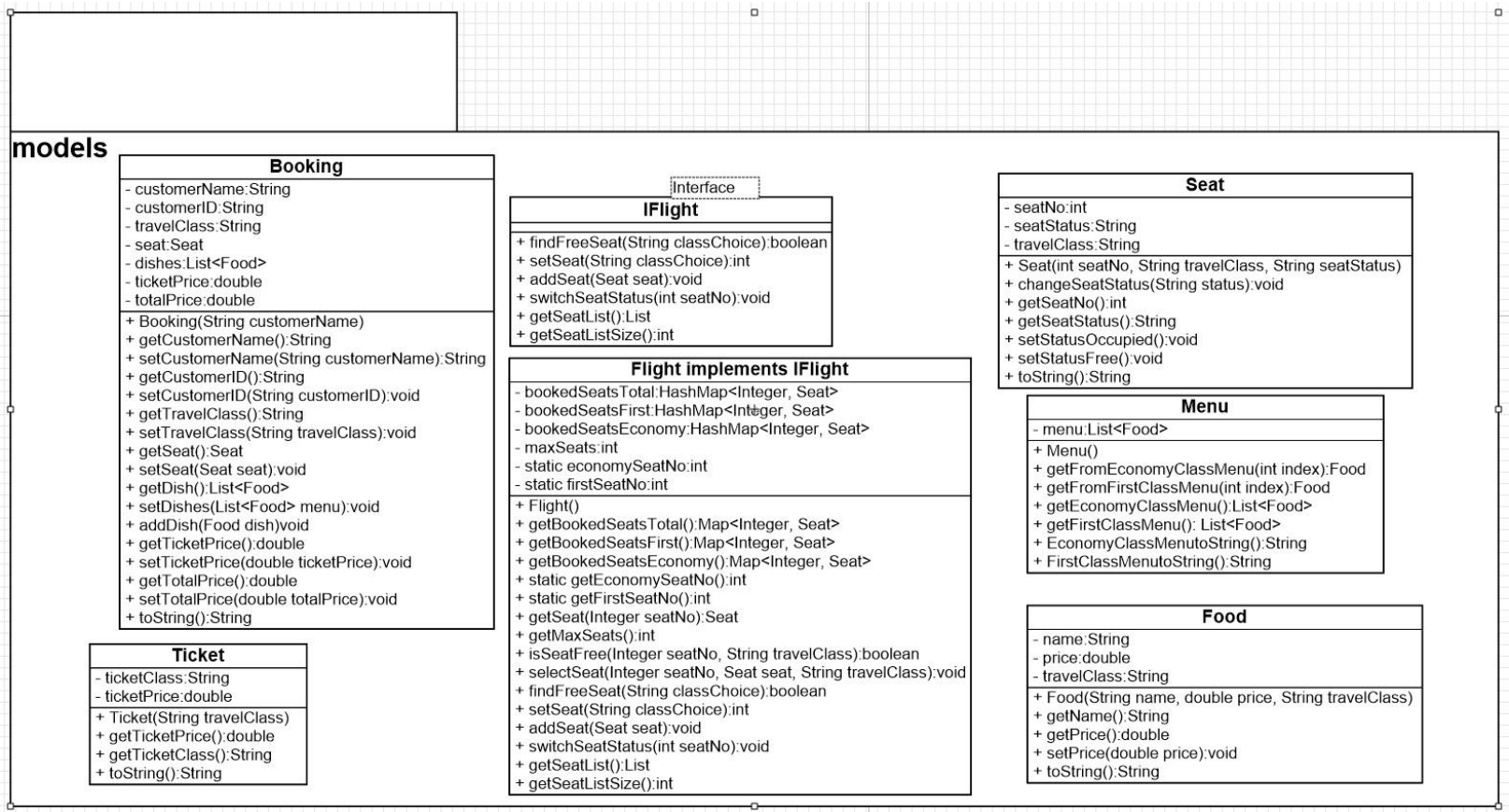
   Change class?

   Allt fullt - sorry
4. Choose a seat from list of seats/give one
5. Food
6. choose from menu
7. Seat price + Food price
8. Customer price
9. Add to income
10. Calc profit

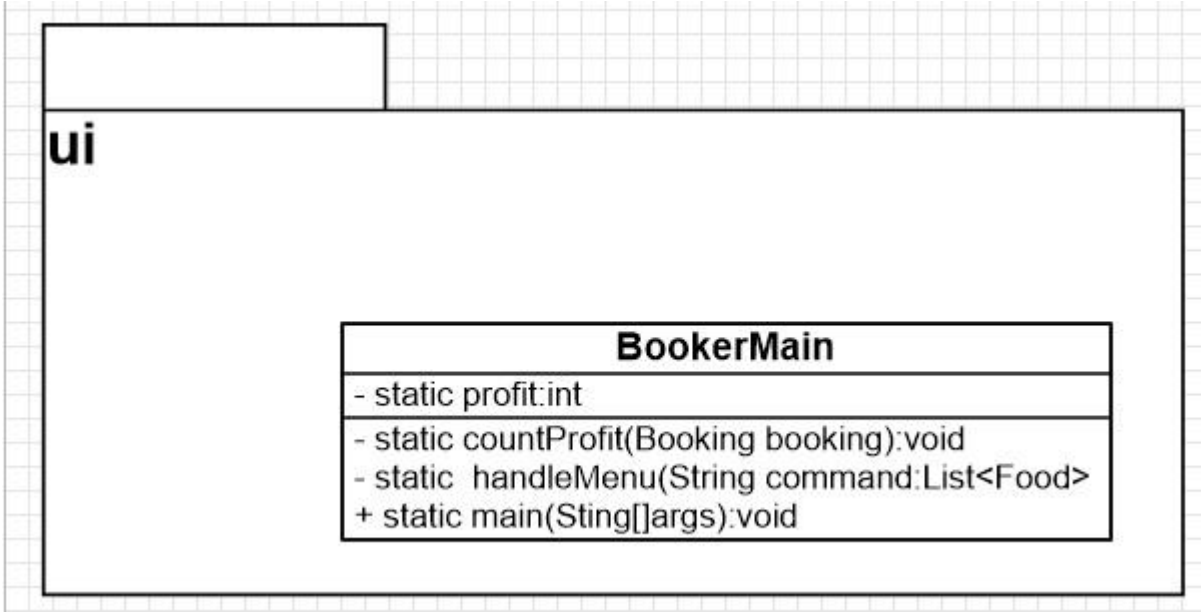
Starting design



Finished Design



Finished Design



Main

User enters a name and then a empty booking is created ready to get filled up with information depending on user input. Example “f” and “e”.

```
System.out.print("Please enter your name : ");
passengerName = in.nextLine();
Booking booking = new Booking(passengerName);
System.out.println("Welcome " + booking.getCustomerName());
System.out.println(" ");

System.out.print("Please choose First Class or Economy Class (F/E): ");
classChoice = in.nextLine();

if (classChoice.equalsIgnoreCase("F")) {
    freeSeat = flight.findFreeSeat(classChoice);

    if (freeSeat == true) {
        Ticket ticket = new Ticket(classChoice);
        seatNo = flight.setSeat(classChoice);
        Seat seat = new Seat(seatNo, classChoice, "Occupied");
        flight.selectSeat(seatNo, seat, ticket.getTicketClass());
        booking.setSeat(seat);
        booking.setTravelClass(ticket.getTicketClass());
        booking.setTicketPrice(ticket.getTicketPrice());
        System.out.println("Your seat nr in " + booking.getTravelClass() + " is " + booking.getSeat().getSe
        System.out.println(" ");
        booking.setDishes(handleMenu("f"));
        countProfit(booking);
        break;
    } else if (freeSeat == false) {
        if (flight.findFreeSeat("e") == false) {
            System.out.println("Sorry but the the flight is fully booked.");
            System.out.println(" ");
            break;
        } else if (flight.findFreeSeat("e") == true) {
            System.out.println("Sorry. There are no free seats in First Class");
            System.out.print("Would you like to check for free seats in Economy Class (Y/N)? ");
            System.out.println(" ");
            wannaSwitchClass = in.nextLine();
            if (wannaSwitchClass.equalsIgnoreCase("Y")) {
```

Main

Asking user to add food to their menu. Example input “1,2,3” will add the tree dishes to the menu. We parse the input to an array [1,2,3] using split(“,”).

```
private static List<Food> handleMenu(String command) {
    Menu m = new Menu();
    ArrayList<Food> bookedMenuForEconomyClass = new ArrayList<>();
    ArrayList<Food> bookedMenuForFirstClass = new ArrayList<>();
    Scanner sc = new Scanner(System.in);

    System.out.println("Book a menu");
    System.out.println("Menu: ");

    switch (command) {

        //First Class
        case "f":


            System.out.println(m.FirstClassMenutoString());
            System.out.println("Example: 1,2,3");
            String menuOrdering2 = sc.nextLine();
            if (!menuOrdering2.equals("")) {

                String[] toParse2 = menuOrdering2.split(",");
                try {
                    for (String s : toParse2) {
                        int parsed = Integer.parseInt(s);
                        bookedMenuForFirstClass.add(m.getFromFirstClassMenu(parsed));
                    }

                } catch (NumberFormatException e) {
                    System.out.println("Wrong format");
                }
            }
            System.out.println("You have chosen: " + (!(menuOrdering2.equals("")) ? menuOrdering2 : "to not order any food"));
            bookedMenuForFirstClass.forEach(System.out::println);
            double sum = 0;
            for(Food food : bookedMenuForFirstClass){
                sum+=food.getPrice();
            }
            System.out.println("Menu total :"+ sum+"kr");
            return bookedMenuForFirstClass;
```

Interface

Cooperating with required methods and returns by using interface.

```
public interface IFlight {

    /**
     *
     * @param classChoice
     * @return
     * Accepts a string of "F" or "E" depending on what class the passenger has chosen.
     * Should search the seats in the chosen class and return true if there is a free seat.
     */
    public abstract boolean findFreeSeat(String classChoice);

    /**
     *
     * @param classChoice
     * @return
     * Accepts a string of "F" or "E" depending on what class the passenger has chosen.
     * Should search for the first free seat in the chosen class and return a seatNo.
     */
    public abstract int setSeat(String classChoice);



    public abstract void addSeat(Seat seat);

    public abstract void switchSeatStatus(int seatNo);

    public abstract List getSeatList();

    public abstract int getSeatListSize();

}
```

Constructors

Creating an empty Booking, ready for later input.

Adding the food directly in constructor.

```
List<Food> menu;

public Menu() {
    menu = new ArrayList<>();
    menu.add(new Food("Beef with fries in wine sauce", 129.90, "First Class"));
    menu.add(new Food("Finest Lobster menu", 139.90, "First Class"));
    menu.add(new Food("Warm soup menu", 49.90, "First Class"));
    menu.add(new Food("Beer", 59.90, "First Class"));
    menu.add(new Food("Wine", 89.90, "First Class"));
    menu.add(new Food("Juice", 34.90, "First Class"));

    menu.add(new Food("Burger", 49.90, "Economy Class"));
    menu.add(new Food("Carbonara", 69.90, "Economy Class"));
    menu.add(new Food("Double sandwich", 39.90, "Economy Class"));
    menu.add(new Food("Hot dog", 29.90, "Economy Class"));
    menu.add(new Food("Coffee", 29.00, "Economy Class"));
    menu.add(new Food("Soda", 36.50, "Economy Class"));
    menu.add(new Food("Water", 10.00, "Economy Class"));

}
```

```
public class Booking {

    private String customerName;
    private String customerID;
    private String travelClass;
    private Seat seat;
    private List<Food> dishes;
    private double ticketPrice;
    private double totalPrice;

    public Booking(String customerName) {
        this.customerID = UUID.randomUUID().toString();
        this.customerName = customerName;
        this.travelClass = "None";
        this.seat = null;
        this.dishes = null;
        this.ticketPrice = 0;
        this.totalPrice = 0;
    }
```

Strings

Using strings for if/else and occupying seats.

```java
package models;

public class Ticket {

    private String ticketClass;
    private double ticketPrice;

    public Ticket(String travelClass) {
        if (travelClass.equalsIgnoreCase("f")){
            ticketClass = "First Class";
            ticketPrice = 20000;
        } else if (travelClass.equalsIgnoreCase("e")){
            ticketClass = "Economy Class";
            ticketPrice = 5000;
        }
    }
```

```java
public class Seat {

        private int seatNo;
        private String seatStatus;
        private String travelClass;

        public Seat(int seatNo, String travelClass, String seatStatus){
                this.seatNo = seatNo;
                this.travelClass = travelClass;
                this.seatStatus = seatStatus;
        }

        public void changeSeatStatus(String status) {
        this.seatStatus = seatStatus;
        }

        public int getSeatNo(){
                return seatNo;
        }

        public String getSeatStatus(){
                return seatStatus;
        }

        public void setStatusOccupied(){
                seatStatus = "OCCUPIED";
        }

        public void setStatusFree(){
                seatStatus = "FREE";
        }

        @Override
        public String toString() {
                return "Seat [Seat No="  + getSeatNo()+ ", isOccupied()=" + getSeatStatus() + "
        }
```

HashMaps

HashMaps representing the booked seats white static Integers keep track on the free seats.

```
public class Flight implements IFlight {

    private HashMap<Integer, Seat> bookedSeatsTotal;
    private HashMap<Integer, Seat> bookedSeatsFirst;
    private HashMap<Integer, Seat> bookedSeatsEconomy;
    private int maxSeats;
    private static int economySeatNo = 6;
    private static int firstSeatNo = 1;

    public Flight() {
        this.maxSeats = 10;
        bookedSeatsTotal = new HashMap<>();
        bookedSeatsFirst = new HashMap<>();
        bookedSeatsEconomy = new HashMap<>();

    }
```

Methods

Special method to use for output String of the two Lists in Menu class, economy and first class.

A static countProfit() that adds a complete booking's total cost and increment the profit every time it's called.

```
public String EconomyClassMenutoString() {
    StringBuilder sb = new StringBuilder();
    int i = 1;
    for (Food food : this.getEconomyClassMenu()) {
        sb.append(i);
        sb.append(". ");
        i++;
        sb.append(food);
        sb.append("\n");
    }
    return sb.toString();
}
```

```
private static void countProfit(Booking booking){
    profit+=booking.getTotalPrice();
    System.out.println("Total company profit: "+(profit*0.3));
}
```