Reservation System

Design

2016

# Class Diagrams



# Pseudocode

## Reserve Seat Method formal parameter list: Customer object.

BEGIN

FOR each Seat in unreserved Seating List

IF Seat Description EQUAL Customer preferred seat Description THEN

SET Seat reserved by Customer

add Seat to Customer reservation list.

add Seat reserved Seating List

Remove Seat from unreserved Seating List

RETURN successful message with number of seat reserved

END IF

END FOR

FOR each Seat in unreserved Seating List

SET counter to 0

IF Seat class NOT EQUAL Customer preferred seat class THEN

IF Seat is in first class THEN

SET missing Requirement to " is in first class"

ELSE

SET missing Requirement to " is in standard class"

END IF

ELSE

INCREMENT counter by 1

END IF

IF Seat position NOT EQUAL Customer preferred seat position THEN

CASE Seat position OF

Window : SET missing requirement to "is a window seat

Aisle : SET missing requirement to “is an aisle seat”

Single : SET missing requirement to “is a single seat”

ENDCASE

ELSE

INCREMENT counter by 1

END IF

IF Seat facing property NOT EQUAL Customer facing property THEN

IF Seat facing is forward THEN

SET missing Requirement to " is facing forward"

ELSE

SET missing Requirement to " is facing backwards"

END IF

ELSE

INCREMENT counter by 1

END IF

IF Seat table property NOT EQUAL Customer table property THEN

IF Seat has table THEN

SET missing Requirement to " is a table seat"

ELSE

SET missing Requirement to " is not a table seat"

END IF

ELSE

INCREMENT counter by 1

END IF

IF Seat ease of access NOT EQUAL Customer preferred ease of access THEN

IF Seat is ease of access seat THEN

SET missing Requirement to " is an ease of access seat"

ELSE

SET missing Requirement to " is not an ease of access seat"

END IF

ELSE

INCREMENT counter by 1

END IF

IF counter EQUAL 4 THEN

PRINT "No exact match found but there is a seat available but it"

plus missing Requirement plus " Would you like to book this seat?"

READ answer

IF answer EQUAL yes THEN

SET Seat reserved by Customer

add Seat to Customer reservation list.

add Seat reserved Seating List

Remove Seat from unreserved Seating List

RETURN successful message with number of seat reserved

END IF

END IF

END FOR

add customer to waiting list

RETURN "No match was found you have been added to waiting list."

END

## Cancel Seat method formal parameter list: Customer object.

BEGIN

SET match found to false

IF customer list is NOT empty THEN

FOR each existing Customer in customer list

IF Customer mail EQUAL existing Customer email in customer list THEN

FOR each Seat in existing reservations list

set match found = false

IF waiting list is NOT empty THEN

FOR each waiting Customer in waiting list

IF Seat description EQUAL waiting Customer seat description THEN

SET seat reserved by waiting Customer

add seat to waiting customer reservations

remove waiting customer from waiting list

set match Found to true

END IF

END FOR

END IF

IF match found NOT EQUL true THEN

SET Seat reserved to null

remove seat from reserved seating list

add to seat to unreserved seating list

END IF

END FOR

remove existing Customer from customer list

RETURN cancellation message

END IF

END FOR

END IF

RETURN "You do not have a reservation"

END

# Deriving the Designs

Having read over the specification, I have identified the different classes, their attributes and behaviour by:

* Checking for real world objects in description or real world things that might interact with program. Doing this I identified seat, carriage and customer.
* Deciding that a carriage contains a number of seats. I have a collection of seat objects in the carriage class. Observing that most of the program’s logic involves the attributes of each seat. I decided that to include the reserve seat and cancel seat methods in the Carriage class. I have keep most of programs logic in the carriage class.
* Knowing that customer’s preference for a seat must match the attributes of a seat object. The best way to model this is to have a separate class for a Seat description. So I can easily compare the seat configuration and the customers preferred seat configuration.
* For the algorithm for reserving and cancelling a seat I have followed the real world procedures of these actions in line with the requirements.

**Next best Match Feature**

For a next best match I think that 1 requirement missing is suitable. Informing the Customer of the missing attribute would also be important as it will help them decide whether or not reserve that seat.