1 STRUCTURAL DESCRIPTION

1.1 DESCRIPTION OF THE CLASSES

<the following section for each class:>

1.1.1 App

Responsibilities

Responsible for GUI

Attributes

-sysManager: SystemManager	Has one single manager for handling the functionality

Methods

App()	Constructor, calls all othere functions responsible for each panel setup
SetMainView()	Sets the main panel that has buttons that upon pressing show another panel
SetUserView()	Sets the person creation panel with appropriate components
SetFlightView()	Sets the flight creation panel with appropriate components
SetTicketView()	Sets the ticket assignment (to person & to flight) panel with appropriate components
SetCancelView()	Sets the ticket cancelation (remove from person & from flight) panel with appropriate components
SetListFlightView()	Sets the flight details list panel with appropriate components where the flight can be selected from ComboBox
SetListPersonView()	Sets the person details list panel with appropriate components where the flight can be selected from ComboBox
UpdateJlist()	Updates the Jlist with content of people/flights if any is added/removed

1.1.2 System Manager

Responsibilities

Responsible for serialization of entire data and calling main functions for both person & flight classes

Attributes

-people: HashMap<String, Person>	Hash map of all the people in the system, the key is a person's email
-flights: HashMap<String, Flight>	Hash map of all the flights in the system, the key is a flight's flight number

Methods

SystemManager()	Constructs, initializes hash maps
createPerson(name: String, age: int, email: String)	Checks if the input data is valid, if yes then creates a person and adds to hashmap
createFlight(flightNumber: String, origin: String, destination: String, planeModel: String, capacity: int, businessCapacity: int, day: int, month: int, year: int)	Checks if the input data is valid, if yes then creates a flight and adds to hashmap
buyTicket(email: String, flightNum: String, seat: String)	Checks data calling `isValidEmailAndFlightNum()` function, adds then flight to person's flights, and adds the person to flight's passengers.
cancelTicket(email: String, flightNum: String)	Checks data calling `isValidEmailAndFlightNum()` function, removers flight from person's flights, and the person from flight's passengers.
isValidEmailAndFlightNum(email: String, flightNum: String)	Checks if the email or flight number already exists in the system

1.1.3 Person

Responsibilities

Responsible for creating, the person and storing flights of that person

Attributes

-flights: HashMap<String, Flight>	Hash map of all flights the person has tickets for
-email: String	Email of the person, used as unique identifier
-age: int	Age of a person
-name: String	Name of a person

Methods

addTicket(flight: Flight, seat: String): void	Add flight to the persons flights, to corrsponding seat
Person(name: String, age: int, email: String)	constructor

1.1.4 Flight

Responsibilities

Responsible for creating, the flgiht and storing passengers of this flight

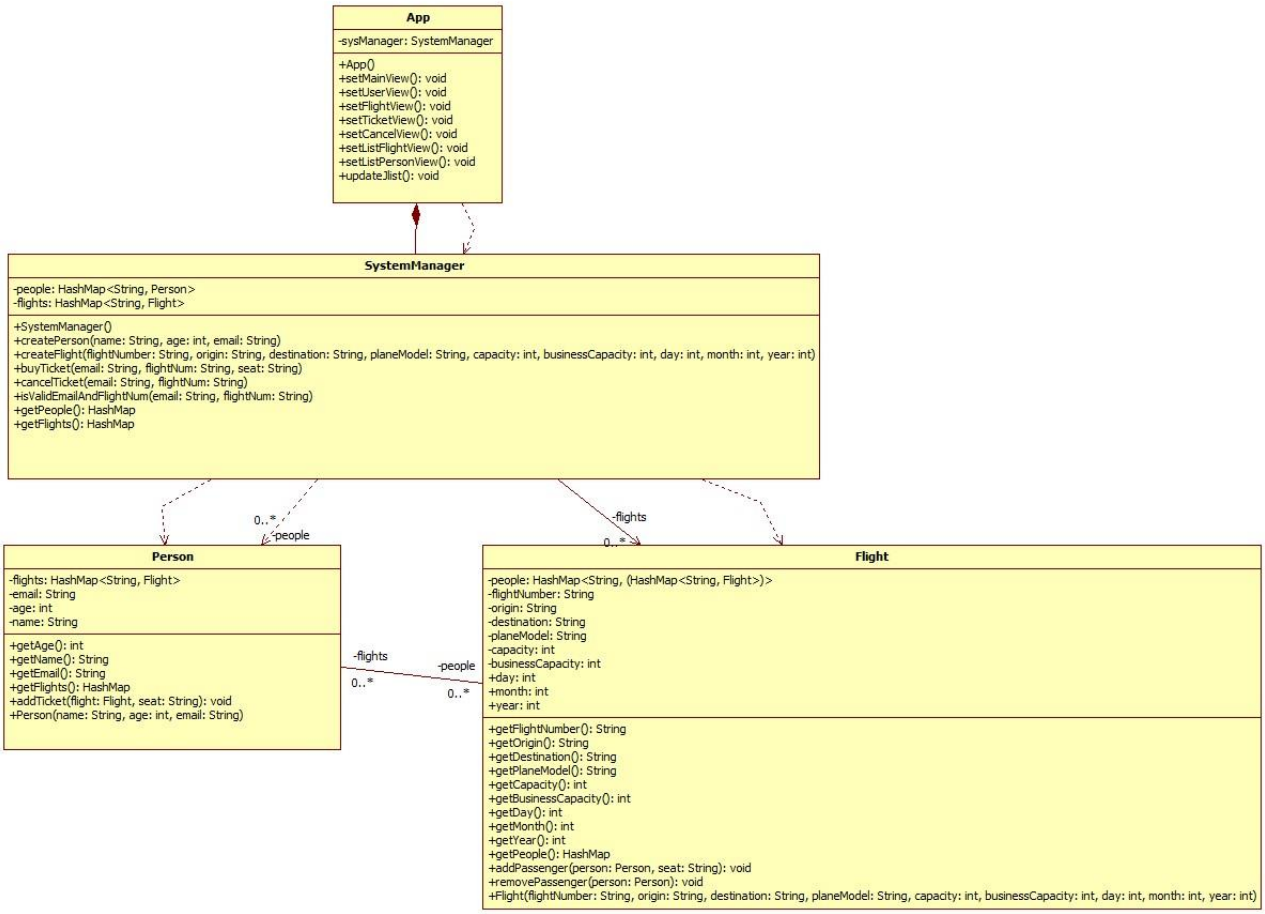
Attributes

- people: HashMap<String, (HashMap<String, Flight>)>	Stores the passengers of this flight
flightNumber: String	Flight number is used as unique identificator
origin: String	Origin city of the flight
destination: String	Destination city of the flight
planeModel: String	Plane model
capacity: int	Total capacity of the plane
businessCapacity: int	Capacity of business seats ont this plane
day: int	Day of the departure
month: int	month of the departure
year: int	year of the departure

Methods

addPassenger(person: Person, seat: String): void	Adds the passenger to the flight
removePassenger(person: Person): void	Removes the passenger from the flight
Flight(flightNumber: String, origin: String, destination: String, planeModel: String, capacity: int, businessCapacity: int, day: int, month: int, year: int)	constructor

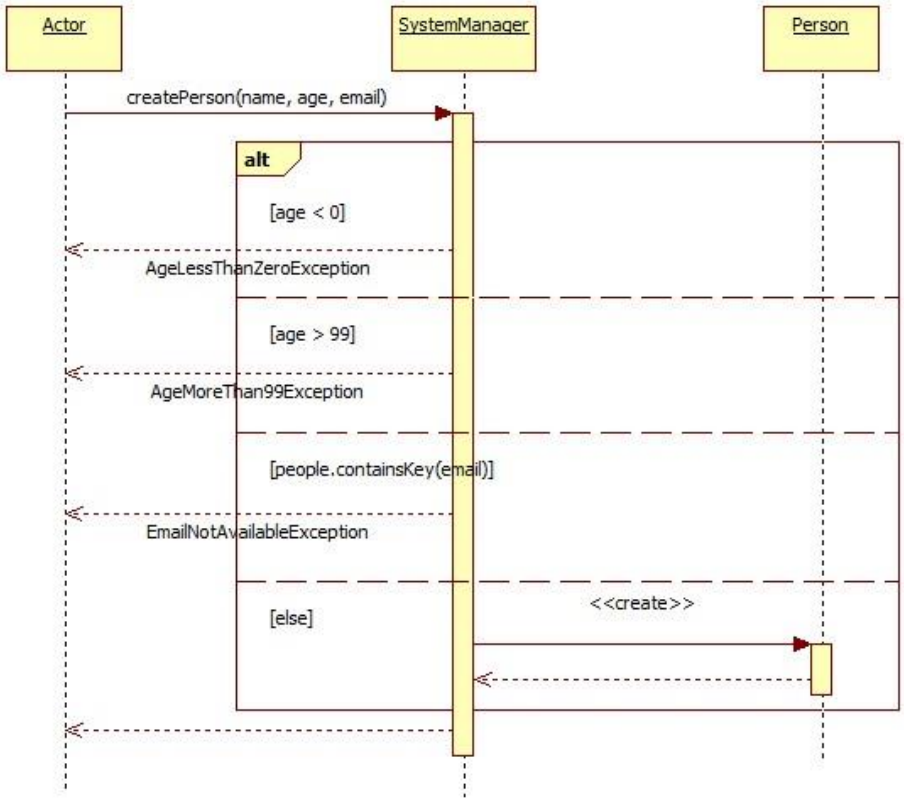
1.2 CLASS DIAGRAM



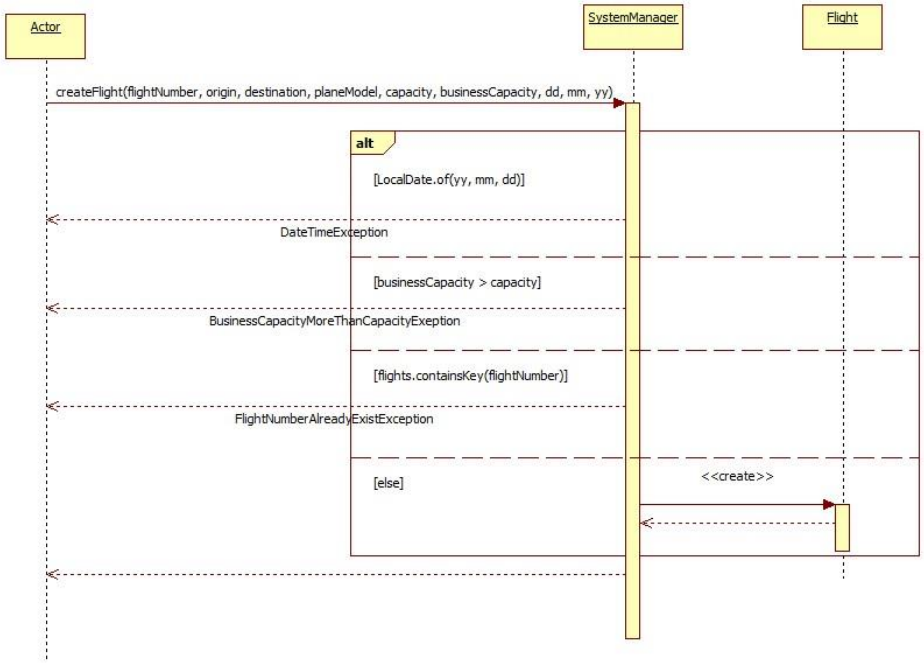
2 BEHAVIORAL DESCRIPTION

2.1 SEQUENCE DIAGRAMS

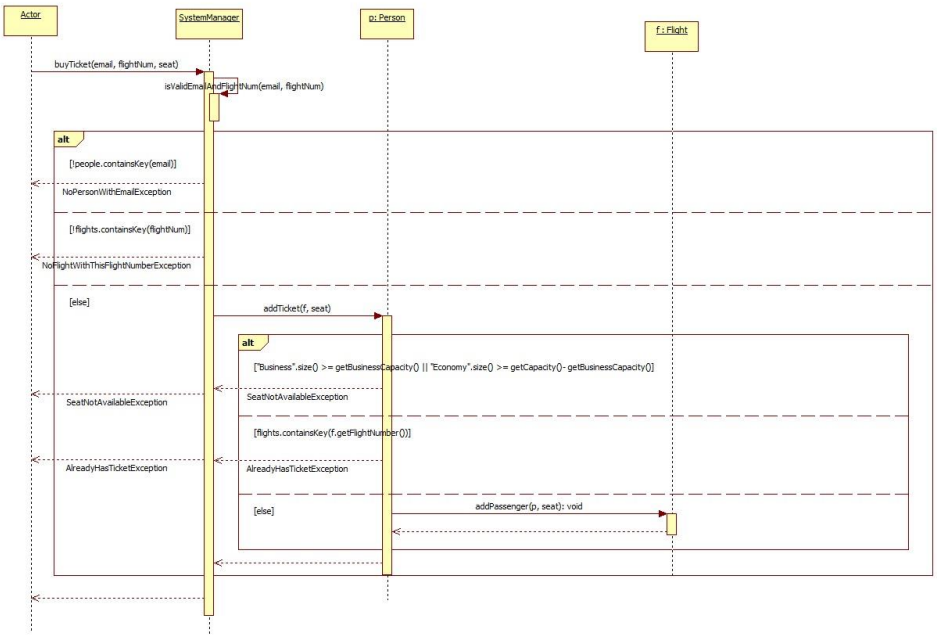
2.1.1 Create Person



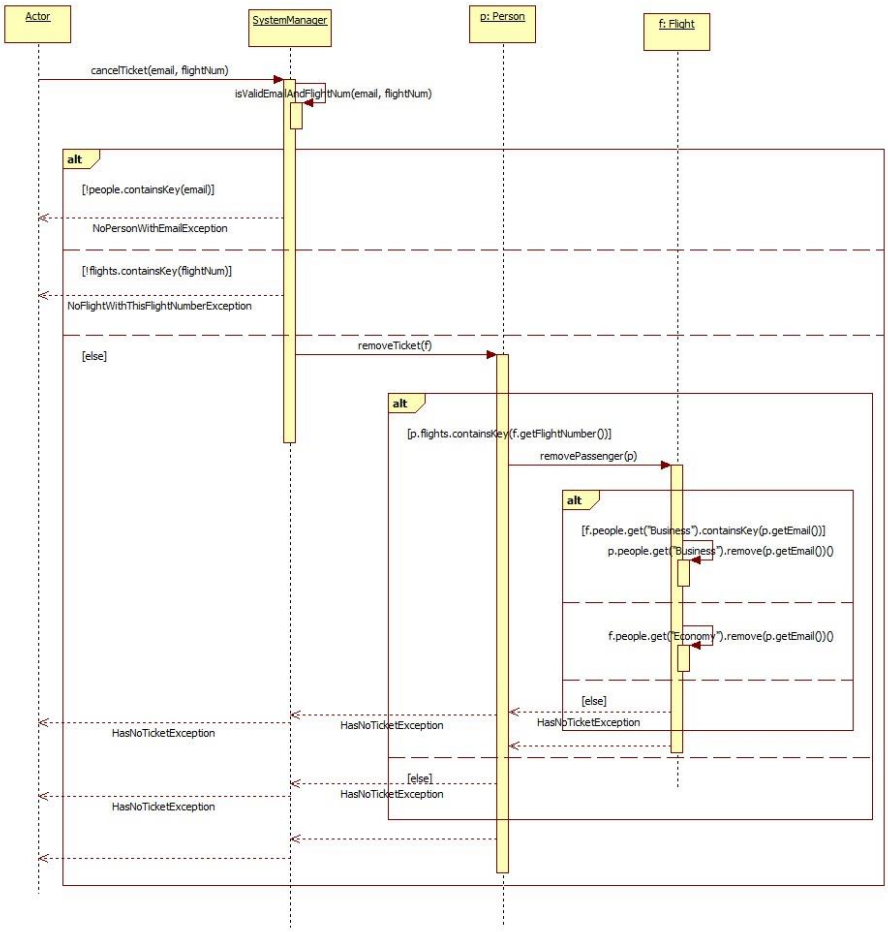
2.1.2 Create Flight



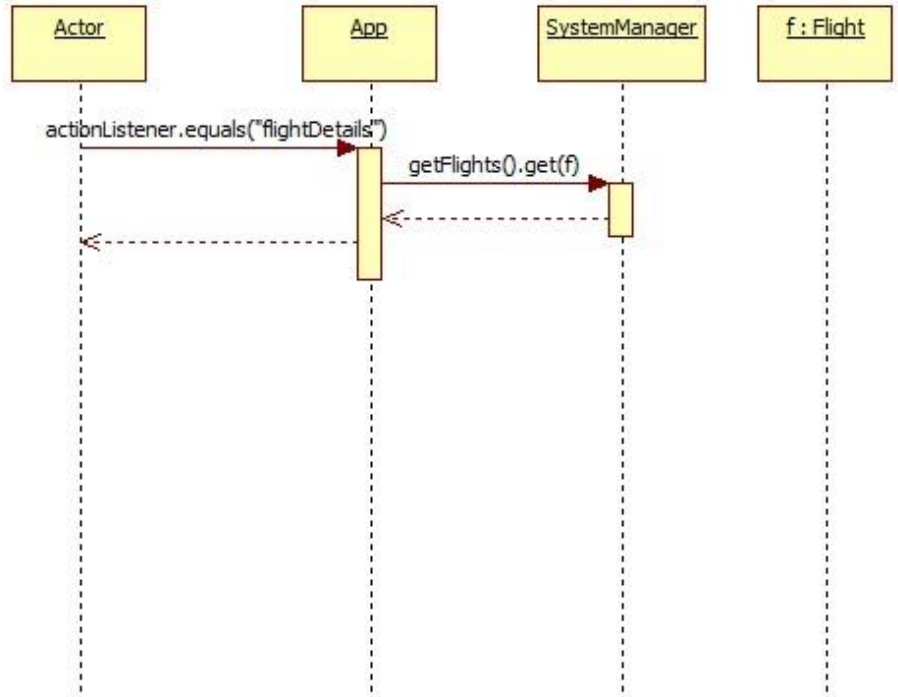
2.1.3 Buy TIcket

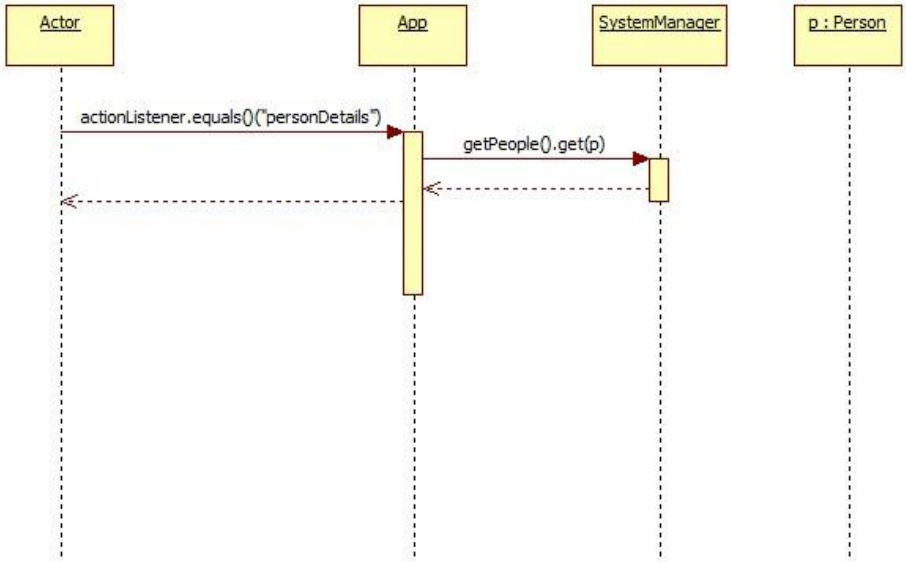


2.1.4 Cancel Ticket



2.1.5 List Flight/Person details – action listener





3 TESTING

3.1 DESCRIPTION OF THE UNIT TESTS

3.1.1 FlightTest

void setUp() creates initial sample data to variables p1, p2, f1, f2, f3

void addPassenger() checks if the flight f1 has no passengers initially, then adds two people p1, p2 and checks if the passengers of the flight equals to the p1, p2

void removePassenger() adds passengers p1, p2 to the flight f1 and checks if the passengers are not empty, then removes the p1, p2 then checks if the passngers hashmap contains these emails.

void personHasNoTicket() removes the person p1 from f1, and expects the error "HasNoTicketException" to be thrown

All getters () checks if the returned value equals to the f1, f2's data

3.1.2 PersonTest

void setUp() creates initial sample data to variables p1, p2, f1, f2, f3

void addTicket() adds ticket of f1 to p1, checks if the person's flight is not empty, and if it containst the flight's number and if its equal to the f1

void SeatNotAvailable() f1 has only 1 Business seat therefore, when we add p1 and p p2 to this flight it expects the error "SeatNotAvailableException" to be trown. Same goes for seat "Economy" of the flight f2

void AlreadyHasTicket() adds f3 to the p1 two times, therefore expecting the error "AlreadyHasTicketException" to be thrown.

void removeTicket() adds f1, f2 to p1 then removes f2 and checks if this flight number is not in the person's flights, similar for f2 flight's, checks if the passengers does not contain the person's email (id)

void HasNoTicket() removes the flight f1 from p1 but since this person has no ticket for this flight it expects the error "HasNoTicketException" to be thrown

All getters () checks if the returned value equals to the p1, p2's data

3.1.3 SystemManagerTest

void setUp() initializes systemManager

void createPerson() creates person, checks if people hashm map is not empty, if it contains the person's id (email), and checks if the getters return the same value as was passed to the createPerson() function

void ageIsLessThanZero() passes negative integer to the age parameter, expects "AgeLessThanZeroException" to be thrown

void emailNotAvailable() creates a 2 people with the same id (email) and expects "EmailNotAvailableException" to be thrown fro the second person

void createFlight() creates sample flight with flight number TK505 and checks If the flights HashMap is not empty, if it contains the key TK505. Then checks if the getters of this flight are equal to the parameters passed to the constructor

void businessCapacityMoreThanCapacity() creates a flight with business capacity more than the total capacity, and expets "BusinessCapacityMoreThanCapacityExeption" to be thrown, because the total capaccity already includes the business capacity

void invalidDate() creates the flight with invalid date out of bound and expects the "DateTimeException" to be thrown

flightNumberAlreadyExist() creates 2 flights with the same flight number and expects "FlightNumberAlreadyExistException" to be thrown for the second flight

void buyTicket() creates sample flight and person, buys him a ticket, then checks if the flight has the person as a passenger and if the person has this flight

void cancelTicket() creates 2 sample flights, a person and buy him the tickets for both flights on different seats. Then checks if the flight's have the person's id (email)

void isValidEmailAndFlightNum() inputs person's email that does not exist and expects "NoPersonWithEmailException" to be thrown. Inputs flight's flight number that does not exist and expects "NoFlightWithThisFlightNumberException" to be thrown

All getters () checks if the returned value equals to the samples' parameters

Total work activity: 150hours

Modeling tool: WhiteStar UML, UML Generator plugin

Other tools:

IntelliJIDEA – Educational edition

References:

https://docs.oracle.com/javase/tutorial/

https://www.stackoverflow.com/

https://docs.oracle.com/javase/tutorial/

https://www.w3schools.com/

https://www.javatpoint.com/java-hashmap

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

https://www.youtube.com/watch?v=sAReaaTxNGU