UMUC Inc

Trip Organizer

Software Design Document

Name (s):

Lab Section:

Workstation:

Date: (05/13/2014)

Software Design Document

Table of Contents

[1. INTRODUCTION 3](#_Toc483484949)

[1.1 Purpose 3](#_Toc483484950)

[1.2 Scope 3](#_Toc483484951)

[1.3 Overview 3](#_Toc483484952)

[1.4 Reference Material 3](#_Toc483484953)

[1.5 Definitions and Acronyms 3](#_Toc483484954)

[2. SYSTEM OVERVIEW 3](#_Toc483484955)

[3. SYSTEM ARCHITECTURE 3](#_Toc483484956)

[3.1 Architectural Design 3](#_Toc483484957)

[3.2 Decomposition Description 4](#_Toc483484958)

[3.3 Exception Handling 4](#_Toc483484959)

[3.4 Design Rationale 5](#_Toc483484960)

[4. DATA DESIGN 5](#_Toc483484961)

[4.1 Data Description 5](#_Toc483484962)

[4.2 Data Dictionary 5](#_Toc483484963)

[5. COMPONENT DESIGN 5](#_Toc483484964)

[6. HUMAN INTERFACE DESIGN 7](#_Toc483484965)

[6.1 Overview of User Interface 7](#_Toc483484966)

[6.2 Screen Images 7](#_Toc483484967)

[6.3 Screen Objects and Actions 7](#_Toc483484968)

[7. REQUIREMENTS MATRIX 7](#_Toc483484969)

[8. APPENDICES 7](#_Toc483484970)

## 1. INTRODUCTION

### 1.1 Purpose

### 1.2 Scope

### 1.3 Overview

### 1.4 Reference Material

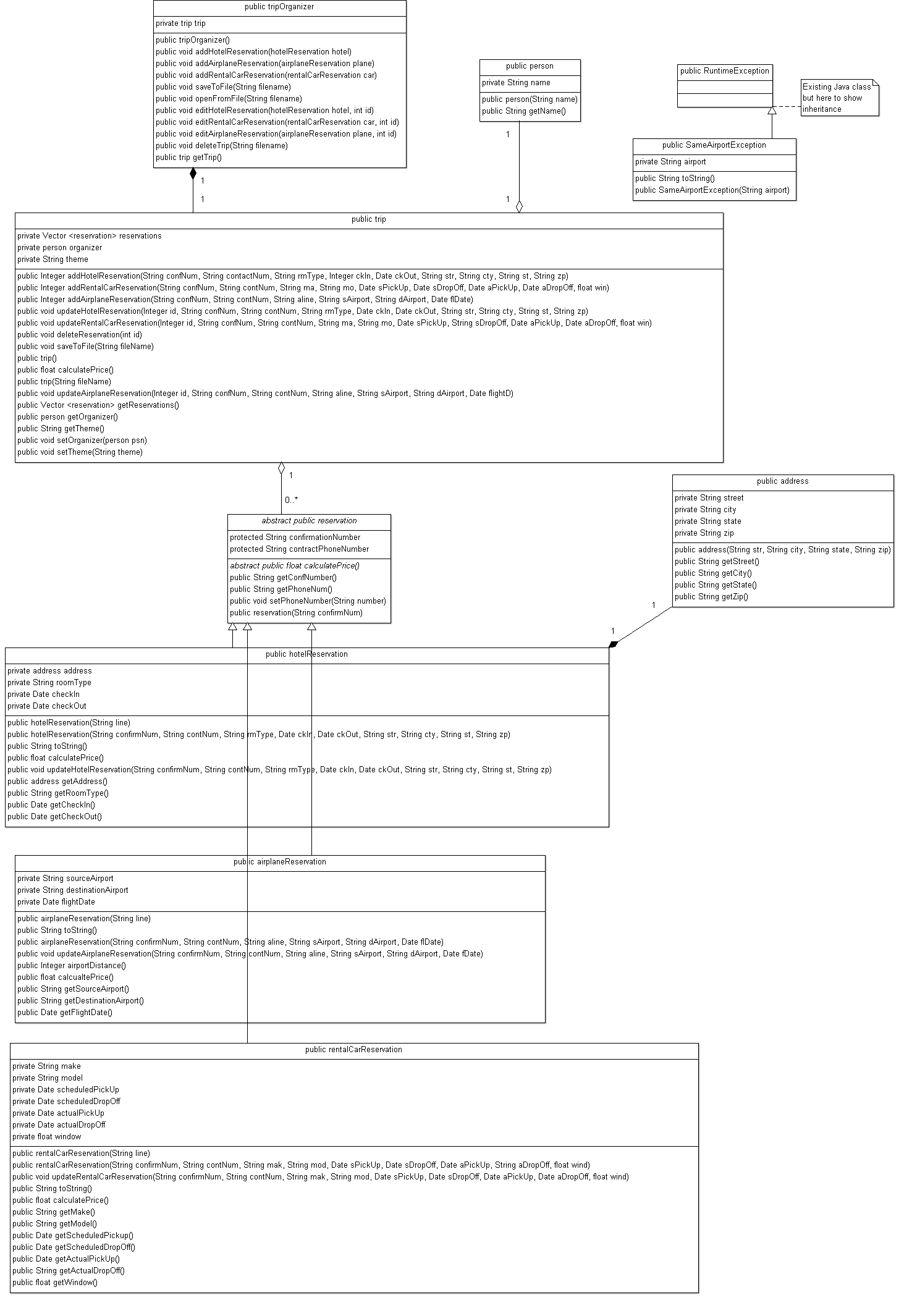
### 1.5 Definitions and Acronyms

## 2. SYSTEM OVERVIEW

## 3. SYSTEM ARCHITECTURE

### 3.1 Architectural Design

The following is the class diagram of the Trip Organizer software architecture.



### 3.2 Decomposition Description

### 3.3 Exception Handling

System will have a new exception called SameAirportException as a child of RuntimeException (unchecked exception). The exception will take as parameter and save the name of an airport and it will set the exception message to “The source and destination airports have the same value XXX” where XXX is the name of the airport. The method toString() will return string with the name of the exception and the same message. The exception will be thrown when airport reservation is created or updated.

### 3.4 Design Rationale

## 4. DATA DESIGN

### 4.1 Data Description

### 4.2 Data Dictionary

## 5. COMPONENT DESIGN

**Class Name:** trip

**Class Description/Purpose:** The trip class is the highest-level class for a single trip. It contains the complete interface for the User Interface to interact with.

**Class Modifiers:** public

**Class Inheritance:** None

**Class Attributes:**

Reservations – Vector – this contains a list of reservations for the trip.

Organizer – Person - This is an object of type person that is the organizer of the trip.

Theme – String – This is the theme of the trip.

**Exceptions Thrown:** None

**Class Methods:**

Constructors: there are two ways to create a new trip object. If the user selects to create a new trip, then constructor trip() is called. If the user opens a trip from a file, then constructor trip(String fileName) is called.

Trip()

Create a new Vector object for attribute reservations

Create a new Person object for attribute organizer

Create a new String object for attribute theme

Trip (String fileName)

Create a new Vector object for attribute reservations

Create a new Person object for attribute organizer

Create a new String object for attribute theme

Read the first line of the file filename and load the trip and organizer into the associated attributes

Loop through all of the other lines of the file

For each line check the beginning

If the beginning is <hotel>

Create a new hotelReservation object, pass in the current line from the file

Add the hotelReservation object to the reservations Vector

If the beginning is <rentalcar>

Create a new rentalCarReservation object, pass in the current line from the file

Add the rentalCarReservation object to the reservations Vector

If the beginning is <airplane>

Create a new airplaneReservation object, pass in the current line from the file

Add the airplaneReservation object to the reservations Vector

addHotelReservation (see class diagram for parameters and order)

returns int which is the number that this new hotel reservation object is on the

Vector

create new hotelReservation object

load method parameters to object attributes

put hotelReservation object on list of reservations

return number of hotelReservation on list

addRentalCarReservation (see class diagram for parameters and order)

returns int which is the number that this new rental car reservation object is on the Vector

create new rentalCar object

load method parameters to object attributes

put rentalCar object on list of reservations

return number of rentalCar on list

addAirplaneReservation (see class diagram for parameters and order)

returns int which is the number that this new airplane reservation object is on the Vector

create new airplaneReservation object

load method parameters to object attributes

put airplaneReservation object on list of reservations

return number of airplaneReservation on list

**Class Name:** airplaneReservation

**Class Description/Purpose:** This class creates object for reservation of a flight

**Class Modifiers:** public

**Class Inheritance:** inherits from reservation class

**Class Attributes:**

String airline – name of airline

String sourceAirport – name of airport from where we fly

String destinationAirport – name of destination airport

Date flightDate – date of the flight

**Exceptions Thrown:**

SameAirportException when source/destination are given or modified and both values are for the same airport

**Class Methods:**

Constructors: there are two ways to create a new airplaneReservation object. If the caller selects to create a new airplaneReservation, then constructor airplaneReservation with parameters for attributes is called. If the caller wants to load existing reservation information from a String, then constructor airplaneReservation (String line) is called.

airplaneReservation(String c, String cn, String al, String sa, String da, Date flightD)

check if sa and da are the same value and if so, throw SameAirportException

load method parameters to object attributes

airplaneReservation(String line)

Check for substring <sourceairport> and assign value to tmpSource attribute

Check for substring <destinationairport> and assign value to tmpDestination attribute

Check if values are the same and if so, throw SameAirportException

Assign tmp values to attributes sourceAirport and destinationAirport

Check for substring <confirmationNum> and assign value to confirmationNum attribute

Check for substring <contactNum> and assign value to contactNum attribute

Check for substring <airline> and assign value to airline attribute

Check for substring <flightdate> and assign value to flightdate attribute

public void updateAirplaneReservation(String c, String cn,

String al, String sa, String da, Date flightD)

check if sa and da are the same value and if so, throw SameAirportException

load method parameters to object attributes

public String toString()

return a string with tags and values for each attribute that has the following values and format

"<airplane>

<confirmationNum>confirmationNum</confirmationNum>

<contactNum>contactNum</contactNum>

<airline>airline</airline>

<sourceairport>sourceAirport</sourceairport>

<destinationairport>destinationAirport</destinationairport>

<flightdate>flightDate</flightdate>

</airplane>

private int airportDistance()

if airports are ‘IAD’ and ‘ORL’ return 723

if airports are ‘IAD’ and ‘BWI’ return 100

if airports are ‘IAD’ and ‘NYC’ return 273

if airports are ‘ORL’ and ‘BWI’ return 776

if airports are ‘ORL’ and ‘NYC’ return 842

if airports are ‘BWI’ and ‘NYC’ return 221

else return 0

public float calculatePrice()

return airportDistance() times 2

**Class Name:** Name

**Class Description/Purpose:** Description

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

**Exceptions Thrown:**

**Class Methods:**

**Class Name:** Name

**Class Description/Purpose:** Description

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

**Exceptions Thrown:**

**Class Methods:**

**Class Name:** Name

**Class Description/Purpose:** Description

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

**Exceptions Thrown:**

**Class Methods:**

**Class Name:** Name

**Class Description/Purpose:** Description

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

**Exceptions Thrown:**

**Class Methods:**

**Class Name:** Name

**Class Description/Purpose:** Description

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

**Exceptions Thrown:**

**Class Methods:**

## 6. HUMAN INTERFACE DESIGN

### 6.1 Overview of User Interface

### 6.2 Screen Images

### 6.3 Screen Objects and Actions

## 7. REQUIREMENTS MATRIX

## 8. APPENDICES