Department of Computer Science
George Mason University

CS 321-002 Software Requirements and Design Modeling

Hotel Reservation System Framework User Guide

3/22/16

Introduction

Each team of students is to implement a subset of the Hotel Reservation System, which manages reservations for a small hotel for the month of January (31 days). In order to assist you with the coding of the framework, each team is provided with a Hotel Reservation System framework as described in this user guide.

The hotel reservation system framework provides:
a) Access to system parameters
b) Functionality for reading instructions from input files
c) Functionality for storing, retrieving, updating, and deleting reservations and customers

System Parameters

The following system parameters are defined as static final integers in the Framework class.

Attribute	Description
NUM_SINGLE_ROOMS	The number of single rooms available in the hotel
NUM_DOUBLE_ROOMS	The number of double rooms available in the hotel
NUM_DAYS	The number of days that the hotel reservation system must support (For January, this value will be set to 31)
SINGLE_RATE	The nightly rate for a single room (in dollars)
DOUBLE_RATE	The nightly rate for a double room (in dollars)
STATUS_RESERVED	The status code for a reserved reservation
STATUS_CHECKED_IN	The status code for a checked in reservation
STATUS_CHECKED_OUT	The status code for a checked out reservation
STATUS_NO_SHOW	The status code for a no show reservation
STATUS_MUST_PAY	The status code for a must pay reservation

Use the values as defined in the framework rather than hard-coding the values into your system. We reserve the right to change the number of rooms and room rates when grading the projects. Your system should support these changes.

Reading Input Files

The hotel reservation system reads in standard input files. The input files contain instructions for the system. Instructions include:

1. Make a reservation
2. Check in
3. Check out
4. Print a management report
5. Day change (signaling that it is now the next day)
6. 6pm alarm (signaling that it is now 6pm)

The framework handles reading input files. When requested, the framework will return the next instruction in the input file as an array of strings.

Instruction Formats (as returned by the framework).

Make Reservation		
String	**Content**	**Example**
0	1	1
1	Name	Gregory C. Stanton
2	Address	4653 Turkey Pen Road, New York, NY 10013
3	Check In Date (1-31)	5
4	Check Out Date (1-31)	10
5	Room Type (1 or 2)	1
6	Number of Occupants (1-4)	2
7	Guaranteed? (0 = No, 1 = Yes)	1
8	Credit Card Info (if 1 above)	MasterCard
9	Credit Card Expiration Date	10/2015
10	Credit Card Number	5412 2765 2503 1552 28

Note: The last 3 strings containing credit card info will not be present if the reservation is not guaranteed. The students should handle this accordingly in their program logic.

Check In		
String	**Content**	**Example**
0	2	2
1	Name	Gregory C. Stanton
2	Updated credit card info	MasterCard
3	Credit Card Expiration Date	10/2015
4	Credit Card Number	5412 2765 2503 1552 28

Note: Updated credit card info will only be present if a credit card was not provided when making the reservation, or if the customer would like to change the credit card on file when checking in. It will not always be provided, so the students should handle this accordingly in their program logic.

Check Out		
String	**Content**	**Example**
0	3	3
1	Name	Gregory C. Stanton

Print Management Report		
String	**Content**	**Example**
0	4	4

Note: If the team is not required to print management reports, they can ignore this instruction.

Day Change Signal		
String	**Content**	**Example**
0	5	5

6pm Signal		
String	**Content**	**Example**
0	6	6

Note: If the team is not required to handle the 6pm timer event, they can ignore this instruction.

Input Instruction Methods Provided by Framework

The framework provides several methods for reading instructions from input files.

Method	Description
void **init**(String *filename*)	Initializes the framework to retrieve instructions from the specified input file. If the file does not exist, or if the file cannot be opened, the framework will throw an IOException.
boolean **hasNextInstruction**()	Returns true if the file contains more instructions, otherwise returns false.
String[] **nextInstruction**()	Returns the next instruction as an array of strings. The length and content of the array depend on the instruction.

Sample code for retrieving instructions from input files:

```
//Initialize the framework to read instructions from the given file
try {
    Framework.init(filename);
}
catch (IOException e) {
   return;
}

//parse the instructions
while (Framework.hasNextInstruction()) {

    //get the instruction data for the next instruction from the framework
    String[] instructionData = Framework.nextInstruction();

    //the first line of every instruction contains the instruction number
    int instruction = Integer.parseInt(instructionData[0]);

    //pass off the handling of the instruction to a different function
    executeInstruction(instruction, instructionData);
}
```

Customers and Reservations

The framework provides functionality for storing, retrieving, modifying and deleting customer and reservation data. This data is passed to and from the framework via Customer and Reservation objects. The specifications for these objects are provided below. You must code these objects according to the specification to guarantee compatibility with the framework.

Customer Fields	
customerID: Integer	A unique identifier for the customer, assigned by the Framework
name: String	The customer's full name
address: String	The customer's address
ccType: String	The type of credit card (Mastercard, Visa, etc)
ccNumber: String	The credit card number
ccExpiration: String	The expiration date of the credit card

Customer Methods	
void **setCustomerID**(int id)	Sets customerID for this object to the value of the parameter "id".
void **setName**(String name)	Sets name for this object to the value of the parameter "name".
void **setAddress**(String address)	Sets the address for this object to "address".

Customer Methods	
void **setCCType**(String ccType)	Sets ccType for this object to the value of the parameter “ccType”.
void **setCCNumber**(String ccNumber)	Sets ccNumber for this object to the value of the parameter “ccNumber”.
void **setCCExpiration**(String ccExpiration)	Sets the CCExpiration for this object to the value of the parameter “ccExpiration”.
int **getCustomerID**()	Retrieves the instance variable “customerID” for this object.
String **getName**()	Retrieves the instance variable “name” for this object.
String **getAddress**()	Retrieves the instance variable “address” for this object.
String **getCCType**()	Retrieves the instance variable “ccType” for this object.
String **getCCNumber**()	Retrieves the instance variable “ccNumber” for this object.
String **getCCExpiration**()	Retrieves the instance variable “ccExpiration” for this object.

All fields must be private and methods must be public. There should be a constructor that does not require any arguments that will initialize customerID to -1 and the remaining fields to null. Additional constructors are up to you.

Reservation	
reservationID: Integer	A unique identifier for the reservation, assigned by the Framework
status: Integer	The status of the reservation: 1 for reserved, 2 for checked in, 3 for checked out, 4 for no show, 5 for must pay
startDate: Integer	The start date of the reservation (1-31)
endDate: Integer	The end date of the reservation (1-31)
roomType: Integer	The room type (1 for single, 2 for double)
numOccupants: Integer	The number of occupants in the room
guaranteed: Boolean	True if the reservation is guaranteed with a credit card, false if not guaranteed
roomNumber: Integer	The room number assigned to this reservation
customerID: Integer	The customer ID for the customer who made this reservation

Reservation Methods	
void **setReservationID**(int reservationID)	Sets reservationID for this object to the value of the parameter “id”.
void **setStatus**(int status)	Sets status for this object to the value of the parameter “status”
void **setStartDate**(int startDate)	Sets startDate for this object to the value of parameter “startDate”.
void **setEndDate**(int endDate)	Sets endDate for this object to the value of parameter “endDate”.
void **setRoomType**(int roomType)	Sets roomType for this object to the value of parameter “roomType”.
void **setNumOccupants**(int numOccupants)	Sets numOccupants for this object to the value of parameter “numOccupants”.
void **setGuaranteed**(int guaranteed)	Sets guaranteed for this object to the value of parameter “”.
void **setRoomNumber**(int roomNum)	Sets roomNumber for this object to the value of parameter “roomNum”.
void **setCustomerID**(int customerID)	Sets customerID for this object to the value of parameter “customerID”.
int **getReservationID**()	Retrieves the instance variable “reservationID” for this object.
int **getStatus**()	Retrieves the instance variable “status” for this object.
int **getStartDate**()	Retrieves the instance variable “startDate” for this object.
int **getEndDate**()	Retrieves the instance variable “endDate” for this object.
int **getRoomType**()	Retrieves the instance variable “roomType” for this object.
int **getNumOccupants**()	Retrieves the instance variable “numOccupants” for this object.
int **getGuaranteed**()	Retrieves the instance variable “guaranteed” for this object.
int **getRoomNumber**()	Retrieves the instance variable “roomNumber” for this object.
int **getCustomerID**()	Retrieves the instance variable “customerID” for this object.

All fields must be private and methods must be public. There should be a constructor that does not require any arguments that will initialize all integer fields to -1 and guaranteed to false. Additional constructors are up to you.

Customers and Reservation Methods Provided by Framework

The framework has the following methods for managing the above entities:

Reservations	
Method	**Description**

Reservations	
Method	**Description**
int **storeReservation**(Reservation *res*)	Stores a new reservation with the data contained in the given Reservation object. Assigns a new reservation ID to the reservation and returns the generated ID.
boolean **deleteReservation**(int *reservationID*)	Removes the reservation with the given reservationID from the system. Returns true if successful, or false if no reservation with the given reservationID exists.
Reservation **getReservationByID**(int *reservationID*)	Returns the reservation with the given reservationID. Returns null if no reservation with the given reservationID exists.
Reservation **getReservationByCID**(int *customerID*)	Returns the reservation with the given customerID. Returns null if no reservation with the given customerID exists.
boolean **modifyReservation**(int *reservationID*, Reservation *res*)	Replaces the stored reservation with given reservationID with the reservation data contained in res. Returns true if successful, otherwise returns false if reservation with given reservationID does not exist.

Customers	
Method	**Description**
int **storeCustomer**(Customer *cus*)	Stores a new customer with the data contained in the given Customer object. Assigns a new customer ID to the customer and returns the generated ID.
boolean **deleteCustomer**(int *customerID*)	Removes the customer with the given customerID from the system. Returns true if successful, or false if no customer with the given customerID exists.
Customer **getCustomerByID**(int *customerID*)	Returns the customer with the given customerID. Returns null if no customer with the given customerID exists.
Customer **getCustomerByName**(String *name*)	Returns the customer with the given name. Returns null if no customer with the given name exists.

Customers	
Method	**Description**
boolean **modifyCustomer**(int *customerID*, Customer *cus*)	Replaces the stored customer with the given customerID with the customer data contained in cus. Returns true if successful, otherwise returns false if customer with given customerID does not exist.

Sample code for making a reservation

```
//Create a customer object for the customer (in your code, you will have to
add customer info to the object)
Customer cus = new Customer();

//add customer to framework database
int customerID = Framework.storeCustomer(cus);

//create a reservation object (you will have to add reservation info to the
object)
Reservation res = new Reservation();

//store reservation in framework database and get the assigned resID
int resID = Framework.storeReservation(res);
```

Sample code for retrieving a reservation by the customer's name

```
Customer cus = Framework.getCustomerByName("Alan Turing");
Reservation res = Framework.getReservationByCID(cus.customerID);
```