tatiloradaburada.com

Object Design

1.0

20.12.2019

Ali Haydar Konuk

Yaren Çoşkun

Oğuzhan Aydın

Fatih İdgü

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc436772639)

[1.1. Object Design Trade-offs 1](#_Toc436772640)

[1.2. Interface Documentation Guidelines 1](#_Toc436772641)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc436772642)

[1.4. References 1](#_Toc436772643)

[2. Packages 1](#_Toc436772644)

[3. Class Interfaces 1](#_Toc436772645)

OBJECT DESIGN DOCUMENT

# Introduction

In this document, we define our project structure, our priority in this project, and classification reason of packages and class interfaces. We used our documentation, RAD and SDD, for guideline.

## Object Design Trade-offs

**Buy vs. Build:**

In our project, we designed project structure and we coded ourselves, and we did it in Eclipse Jee. We didn’t use any external codes.

**Memory Space vs. Response Time**

When we build database, we accept some unnecessary data storage to decrease number of tables by mixing some tables.

**Roles in Project**

In our project, number of Registered User is clearly greater than number of other roles. Thus, we provide more functions.

Our project includes a set of time-based boundary cases, and our project obligate user for required information when user register or add hotel.

## Interface Documentation Guidelines

We preferred some patterns when we design project. It provides that what view or occur this class, method, button or tab.

* Our classes divided by roles’ names or their view such as hoHotelsEditBean, PastReservationBean etc.
* Our methods divided by their functions such as AddHotel, EditHotel etc.
* Our buttons or tabs divided by what it does or view such as Past Reservation, Add Hotel.

## Definitions, Acronyms, and Abbreviations

ODD – Object Design Documentation

RAD – Requirement Analysis Documentation

SDD - System Design Document

## References

RAD of tatiloradaburada.com

SDD of tatiloradaburada.com

[**https://www.tatilbudur.com/**](https://www.tatilbudur.com/)

# Packages

# Models

|  |  |
| --- | --- |
| package admin; | Admin.java |
| package hotelOwnerHotels; | hoHotels.java |
| package hotelOwnerInfo; | hoInfo.java |
| package registeredUser; | RegisteredUser.java |
| package reservations; | Reservation.java |
| package user; | User.java |

# Controllers

|  |  |
| --- | --- |
| package HotelReservation\src\admin; | AdminBean.java |
| package HotelReservation\src\hotelOwnerHotels; | hoHotelsBean.java |
| package HotelReservation\src\hotelOwnerInfo; | hoInfoBean.java |
| package HotelReservation\src\registeredUser; | RegisteredUserBean.java  EditInfoBean.java |
| package HotelReservation\src\reservations; | ReservationBean.java  MakeReservationBean.java  PastReservationBean.java |
| package HotelReservation\src\visitorPage; | VisitorPageBean.java |
| package HotelReservation\src\login | LoginBean.java |

**Views**

|  |  |
| --- | --- |
| Admin  HotelReservation\WebContent | VisitorPage.xhtml  login.html  Admin.xhtml |
| Hotel owner  HotelReservation\WebContent | AddNewHotel.xhtml  hotelOwnerMainPage.xhtml  VisitorPage.xhtml  login.xhtml  register.xhtml |
| Registered user  HotelReservation\WebContent | VisitorPage.xhtml  login.xhtml  register.xhtml  editUserInfo.xhtml  makeReservation.xhtml  pay.xhtml  registerUser.xhtml |
| Visitor  HotelReservation\WebContent | VisitorPage.xhtml  login.xhtml  register.xhtml |

# Class Interfaces

**Model Name: User**

**Model Attributes/Explanation/Dependencies:**

* **‘userid’ :** Id of the user which has been given by database.
* **‘email’:** Email address of the user which he/she used for login to system.
* **‘typee’:** Users type depending on his/her role in the system.

This model has dependencies to elements of ‘login’, ‘registeredUser’ and ‘reservations’ packages to complete some of it’s operations properly.

**Model Operations:**

* **LoginBean.SetValues():**Checks if a user with given input values exist in database. If exist creates a ‘User’ and . If does not exist gives a warning.
* **LoginBean.userPage():**Controls ‘typee’ of ‘User’ and returns his/her main page address.
* **EditInfoBean.init():**Checks database with ‘email’ belong to ‘User’ and gets his/her informations.
* **RegisteredUserBean.RegisteredUserBean():**Checks database with ‘email’ belong to ‘User’ and gets his/her informations.
* **RegisteredUserBean.SendDeleteRequest():**Checks database with ‘userid’ belong to ‘User’, sends a account delete request to admin.
* **MakeReservationBean.getBalaanceDB():**Checks database with ‘email’ belong to ‘User’ and gets his/her balance information.
* **MakeReservationBean.createReservation():**Creates a ‘Reservation’ with inputs and ‘userid’ of ‘User’ and add that reservation to database.
* **MakeReservationBean.payment():**Updates balance of user in database with the changes ‘balance’ on system and uses ‘userid’ of ‘User’ to update.
* **PastReservationBean.PastReservationBean():**Checks database with ‘email’ belong
* **ReservationBean.ReservationBean():**Checks database with ‘email’ belong to ‘User’ and gets his/her future reservations.
* **ReservationBean.delete():**Deletes choosen reservation, gets users balance from database with ‘email’ of ‘User’ ,add canceled reservations payment to balance and updates balance on database.

**Model Name:** Hotel

**Model Attributes/Explanation/Dependencies:**

* **hid**: ID of the hotel
* **hownerid**: ID of the hotel’s owner
* **name**: Name of the hotel
* **location**: Location of the hotel
* **quality**: Quality of the hotel. It keeps number between 1 and 5.
* **costs**: Price of the Standart Room
* **vrooms**: Number of the vacant room for Standart Room
* **coste**: Price of the Executive Room
* **vroome**: Number of the vacant room for Executive Room
* **costp**: Price of the Presidential Room
* **vroomp**: Number of the vacant room for Presidential Room
* **isactive**: It keeps hotel is whether active
* **isaccepted**: It keeps hotel is whether accepted by Admin

**Model Operations:**

**This model does not contain any operations at any controller.**

**Model Name:** HotelOwner

**Model Attributes/Explanation/Dependencies:**

* **hoid**: ID of the hotel owner
* **name**: Name of the hotel owner
* **lastname**: Lastname of the hotel owner
* **pnumber**: Phone number of the hotel owner
* **gender**: Gender of the hotel owner
* **bdate**: Birthdate of the hotel owner
* **issendrequest**: It keeps hotel owner whether send ‘Delete My Account’ request to Admin.

**Model Operations:**

* **hoInfoBean.SendDeleteRequest():** Send a delete request to Admin for Hotel Owner’s account
* **editHoInfoBean.EditInfo():** Edit Hotel Owner’s information
* **hoHotelsAddBean.AddHotel():** Add new Hotel to Hotel Owner’s hotel list
* **hoHotelsEditBean.EditHotel():** Edit Hotel’s information
* **hoHotelsBean.ActiveHotel():** Making Hotel Active
* **hoHotelsBean.DeactiveHotel():** Making Hotel Deactive

**Model Name:** Registered User

**Model Attributes/Explanation/Dependencies:**

* **Id:** ID which is given by the system for database
* **Name:** name of the registered user
* **lastName:** last of the registered user
* **email:** Email of the registered user that is used in login session.
* **pnumber:** Phone number of registered user
* **birthDate:** birth date of registered user
* **gender:** gender of registered user
* **isSendRequest:** Statement that determines whether the user has sent a request to admin for delete his/her account
* **balance**: the balance for the registered user to pay the hotel fee.

This model has dependencies to elements of ‘Admin’ package to complete some of it’s operations properly.

**Model Operations:**

* **EditInfoBean.EditInfo():** Retrieves new information about user from the registered user and updates the database.
* **RegisterUserBean.SendDeleteRequest ():** sendsa request to admin from registered user for delete his/her account

**Model Name:** Reservation

**Model Attributes/Explanation/Dependencies:**

* **İd:** Id which is given by the system for database.
* **userid:** Id of registered user who made the reservation.
* **hotelid:** Id of the hotel the user has booked.
* **iscancelld:** Shows whether the reservation is canceled or not.
* **startdate:** Start date of the reservation.
* **Enddate:** End date of the reservation.
* **numberofroom:** number of rooms booked.
* **roomtype:** Type of room booked.
* **hotelname:** Name of the hotel booked.
* **Cost:** reservation fee.

This model has dependencies to elements of ‘Registered User’ package to complete some of it’s operations properly.

**Model Operations:**

* **MakeReservationBean.getBalaanceDB ():** Retrieves balance of registered user from database.
* **MakeReservationBean.takeHotelinfoDate ():** Retrieveshotelid, hotelname, startdate, enddate informations from visitor page for using on make reservation page.
* **MakeReservationBean.takeReservationInfo():** Retrieves numberofroom, hotelid, , startdate, enddate, roomtype informations from make reservation page for creating reservation.
* **MakeReservationBean.calculatecost():** Retrieves roomtype information and calculates the reservation fee.
* **MakeReservationBean.createReservation():** Creates reservations based on user's selections.
* **MakeReservationBean.payment ():** Reduces the fee from the user's balance.
* **MakeReservationBean.changeRoomNumber ():** Changesthe number of rooms in a room type of a hotel after reservation is done.
* **ReservationBean.delete ():**Cancels a reservations made by the user , refunds to the registered user.
* **ReservationBean.changeRoomNumber():** Changesthe number of rooms in a room type of a hotel after reservation is canceled.

**Model Name:** Comment

**Model Attributes/Explanation/Dependencies:**

* **id:** Id which is given by the system for database.
* **hotelid:** Id of the hotel the registered user has commented on.
* **Text:** Comment about hotel
* **userid:** Id of the registered user who wrote the comment.
* **isdeleted:** Indicates whether the comment has been deleted or not.
* **rname:** Registered user’s name
* **hname:** Name of the hotel

This model has dependencies to elements of ‘Registered User’ and ‘hotelOwnerHotels package to complete some of it’s operations properly.

**Model Operations:**

* **CommentBean.takeHotelId():** Retrieves name of the hotel and hotel id.
* **CommentBean.writecomments():** Writes comment into the hotel

**Model Name:** Admin

**Model Attributes/Explanation/Dependencies:**

* **userid** : id of admin
* **email** : email of admin which is using for login
* **password** : password of admin whic is using for login
* **type** : type for admin is ‘a’, after login system checks user’s type and opens related page with user type.
* **isdeleted** : is ‘0’ for admin, checks users are deleted from system or not.

This model has no dependencies with other packages.

**Model Operations:**

* **AdminBean**(); Brings list of hotels owners, registered users which are sent request for deleting thier account, list of hotels which are waiting for approval and lists of comments from database.
* **AdminBean.delete();** Marks as deleted for registered users and hotel owners, if any current reservation available for related with this hotel owner’s hotels, system cancel all of them and pay back their money. If registered user has any reservation which are continuing, registered user’s reservations cancels by system and increase vacant rooms which are related hotels.
* **AdminBean.deleteComment();** deletes comments.
* **AdminBean.accept();** Accept hotels.
* **AdminBean.decline();** Decline hotels.
* **AdminBean.getRegUser();** Get registered users list which are sent request for deleting their account.
* **AdminBean.getComments();** Get list of comments.
* **AdminBean.getHotelss();** Get list of hotels for approval.
* **AdminBean.getHoOwner();** Get hotel owners list which are sent request for deleting their account.