
A
Address
- addressID: int
- number: int
- street: String
- city: String
- state: String
- country: String
- postalCode: String
+ toString(): String
Admin
+ toString(): String
Airline
- name: String
- airplanes: ArrayList<Airplane>
+ addAirplane(airplane: Airplane): void
+ removeAirplane(airplane: Airplane): void
+ toString(): String
Airplane
- airplaneID: int
- airline: Airline
- name: String
- status: String
- flightNumber: int
- routesFlying: ArrayList<Route>
+ addRoute(route: Route): void
+ removeRoute(route: Route): void
+ toString(): String
AdminGUI
- currentUser: String
- tabbedPane: JTabbedPane
- flightTableModel: DefaultTableModel
- flightTable: JTable
- addFlightButton: JButton
- editFlightButton: JButton
- removeFlightButton: JButton
- refreshFlightsButton: JButton
- userTableModel: DefaultTableModel
- userTable: JTable
- addUserButton: JButton
- editUserButton: JButton
- removeUserButton: JButton
- systemLogArea: JTextArea
- generateReportButton: JButton
- viewPromotionsButton: JButton
- db: DatabaseManager
- initializeGUI(): void
- createFlightManagementPanel: JPanel
- createUserManagementPanel: JPanel
- createSystemManagementPanel: JPanel
B
Booking
- bookingId: int
- customer: Customer
- flight: Flight
- seatNumber: int
+ toString(): String
BookingController
- db: DatabaseManager
+ cancelBooking(bookingId: int) : void
+ createBooking(customer: Customer, flight: Flight, seatNumber: int): Booking
+ updateBooking(booking: Booking): void
C
Card
- number: String
- holder: String
- expiry: String
- cvv: String
CustomDate
- year : int
- month: int
- date : int
+ toSQLDate(): String
+ StringToDate(date: String): CustomDate
+ toString(): String
Customer
- address: Address
- email: String
- paymentStrategy: PaymentStrategy
+ processPayment(amount: double): String
+ update(promotion: Promotion): void
+ toString(): String
CreditCardPayment
- card: Card
- SUCCESS_RATE: double
+ pay(amount: double): String
CustomerController
- db: DatabaseManager
+ createCustomer(username: String, firstName: String, lastName: String, dob: CustomDate, email: String): Customer
+ updateCustomer(customer: Customer): void
+ deleteCustomer(customerId: int): void
CustomerGUI
- db: DatabaseManager
- currentCustomer: Customer
- flightController: FlightController
- customerController: CustomerController
- bookingController: BookingController
- promotionManager: PromotionManager
- departureField: JTextField
- destinationField: JTextField
- dateField: JTextField
- searchFlightsButton: JButton
- clearSearchButton: JButton
- flightTable: JTable
- flightTableModel: DefaultTableModel
- bookFlightButton: JButton
- bookingsTable: JTable
- bookingsTableModel: DefaultTableModel
- cancelBookingButton: JButton
- refreshBookingsButton: JButton
- profileFirstNameField: JTextField
- profileLastNameField: JTextField
- profileEmailField: JTextField
- profileUsernameField: JTextField
- initializeControllers(): void
- registerForPromotions(): void
- loadCustomerData: void
- initializeGUI(): void
- createFlightSearchPanel: JPanel
- createBookingsPanel: JPanel
- createProfilePanel(): JPanel
- loadProfileData(): void
- refreshBookingsTable(): void
- clearSearchFields(): void
D
DatabaseManager
- connection: Connection
- instance: DatabaseManager
- URL: String
- user: String
- pass: String
- DatabaseManager()
+ connect(): void
+ disconnect(): void
+ getInstance(): DatabaseManager
+ isConnected: boolean
+ insert(tableName: String, columns: String[], values: Object[], expectedGeneratedKey: boolean)
+ insertPerson(firstName: String, lastName: String, dateBorn: String, role: String): int
+ insertPerson(firstName: String, lastName: String, dateBorn: String, password: String, role: String): int
+ insertCustomer(personId: int, email: String): void
+ insertAgent(personId: int): int
+ insertAirline(airlineName: String): int
+ insertAirplane(airlineName: String, name: String, flightNumber: int): int
+ insertAddress(postalCode: String, number: int, street: String, city: String, state: String, country: String): int
+ insertFlight(airplaneId: int, routeId: int, departureDate: CustomDate, arrivalDate: CustomDate, availableSeats: int, flightLength: String, price: float): int
+ insertBooking(customerId: int, flightId: int, seatNumber: int): int
+ insertRoute(origin_id: int, destination_id: int): int
+ updatePerson(person: Person): int
+ updateCustomer(customerId: int, email: String): int
+ updateFlight(flightId: int, airplaneId: int, routeId: int, departureDate: CustomDate, arrivalDate: CustomDate, availableSeats: int, flightLength: String, price: float): int
+ updateAirplane(airplaneId: int, airlineName: String, name: String, flightNumber: int): int
+ updateAddress(addressId: int, postalCode: String, number: int, street: String, city: String, state: String, country: String): int
+ updateRoute(routeId: int, originId: int, destinationId: int): int
+ updateBooking(bookingId: int, customerId: int, flightId: int, seatNumber: int): int
+ update(tableName: String, columns: String[], values: Object[], whereClause: String, whereValues: Object[]): int
+ delete(tableName: String, whereClause: String, whereValues: Object[]): int
+ deleteFlight(flightId: int): int
+ deleteBooking(bookingId: int): int
+ deletePerson(personId: int): int
+ deleteCustomer(customerId: int): int
+ deleteAgent(agentId: int): int
+ deleteAirline(airlineName: String): int
+ deleteAirplane(airplaneId: int): void
+ deleteAddress(addressId: int): int
+ deleteRoute(routeId: int): int
+ deleteBookingByCustomer(customerId: int): int
+ deleteBookingByFlight(flightId: int): int
+ deleteAgentCustomerRelationship(agentId: int, customerId: int): int
F
Flight
- flightId: int
- airplane: Airplane
- route: Route
- departureCustomDate: CustomDate
- arrivalCustomDate: CustomDate
- availableSeats: int
- flightTime: String
- price: float
+ toString(): String
FlightAgent
- clients: ArrayList<Customer>
+ addClient(customer: Customer): void
+ removeClient(customer: Customer): void
+ toString(): String
FlightController
- db: DatabaseManager
+ getFlight(flightId: int): Flight
+ createFlight(airplane: Airplane, route: Route, departureCustomDate: CustomDate, arrivalCustomDate: CustomDate, availableSeats: int, flightTime: String, price: float)
FlightAgentGUI
- db: DatabaseManager
- currentUser: String
- tabbedPane: JTabbedPane
- flightController: FlightController
- customerController: CustomerController
- bookingController: BookingController
- flightSearchField: JTextField
- flightSearchType: JComboBox<String>
- searchFlightButton: JButton
- refreshFlightsButton: JButton
- flightTable: JTable
- flightTableModel: DefaultTableModel
- customerSearchField: JTextField
- searchCustomerButton: JButton
- refreshCustomersButton: JButton
- customerTable: JTable
- customerTableModel: DefaultTableModel
- addCustomerButton: JButton
- editCustomerButton: JButton
- removeCustomerButton: JButton
- bookingTable: JTable
- bookingTableModel: DefaultTableModel
- createBookingButton: JButton
- editBookingButton: JButton
- cancelBookingButton: JButton
- refreshBookingsButton: JButton
- initializeControllers(): void
- initializeGUI(): void
- createFlightManagementPanel: JPanel
- createCustomerManagementPanel(): JPanel
- createBookingManagementPanel(): JPanel
- loadInitialData(): void
- refreshFlightsTable(): void
- refreshCustomersTable(): void
- refreshBookingsTable(): void
L
LoginGUI
- usernameField: JTextField
- passwordField: JTextField
- loginButton: JButton
- cancelButton: JButton
- DB_URL: String
- initializeGUI(): void
- authenticateUser(username: String, password: String): String
- getRoleDisplayName(role: String): String
- openMainApplication(role: String, username: String): void
P
<<interface>>
PaymentStrategy
+ pay(amount: double): String
PayPalPayment
- card: Card
- SUCCESS_RATE: double
+ pay(amount: double): String
<<abstract>>
Person
- id: int
- firstName: String
- lastName: String
- DoB: Date
+ toString: String
Promotion
- promoCode: String
- discountRate: double
- description: String
- startDate: CustomDate
+ toString(): String
PromotionManager
- instance: PromotionManager
- observers: List<PromotionObserver>
- activePromotions: List<Promotion>
+ getInstance(): PromotionManager
+ registerObserver(observer: PromotionObserver): void
+ removeObserver(observer: PromotionObserver): void
+ notifyObservers(promotion: Promotion): void
+ addPromotion(promotion: Promotion): void
+ removePromotion(promoCode: String): void
+ getPromotionByCode(promoCode: String): Promotion
+ getActivePromotions(): List<Promotion>
+ loadPromotionsFromDatabase(): void
<<interface>>
PromotionObserver
+ update(promotion: Promotion): void
<<interface>>
PromotionSubject
+ registerObserver(observer: PromotionObserver): void
+ removeObserver(observer: PromotionObserver): void
+ notifyObservers(promotion: Promotion): void
R
Route
- departureLocation: Address
- arrivalLocation: Address
+ method(type): type