Package:User «abstract» User -firstName: String -lastName: String -email: String -address: String -creditCard: CreditCard -bookedFlights: Set<FlightItinerary> -password: String -isAdmin: boolean -date: DateFormat +User(firstName: String, lastName: String, email: String, address: String, ccNumber: String, expiryDate: String, password: String) +getFlightsFormatted(date: String, origin: String, destination: String, sort: String): ArrayList<String> +getFlights(date: String, origin: String, destination: String, sort: String): List<Flight> +uploadClientInfo(path: String, database: ClientDatabase): void +uploadFlightInfo(path: String, database: FlightDatabase): void +getFirstName(): String +setFirstName(firstName: String): void +getLastName(): String +setLastName(lastName: String): void +getEmail(): String +setEmail(email: String): String +getAddress(): String +setAddress(address: String): String +getExpiryDate(): String +setExpiryDate(expiryDate: Date): void +getbookedFlights(): Set<FlightItinerary> +sortTravel(array: List<Flight>): void +sortCost(array: List<Flight>): void +makeAdmin(): void +isAdmin(): boolean

+getPassword(): String

+setPassword(password: String): void +bookFlight(flight: FlightItinerary): void

Administrator

+Administrator(email: String, password: String)

«enumeration» ClientField

 \triangle

-firstName: String -lastName: String -password: String -address: String -creditCard: String -expiryDate: String

CreditCard

-cardNumber: String -expiryDate: Date

+CreditCard(cardNumber: String,

expiryDate: Date)

+getNumber(): String +getExpiry(): Date

+setExpiry(expiry: Date): void +setNumber(cardNum: String): void

+equals(obj: Object): boolean

Package:Validate ValidateInput -MIN LAYOVER: final long

-MAX LAYOVER: final long

-locale: final Locale -date: final DateFormat

-dateTime: final DateFormat

-time: final DateFormat

+validateDate(date: String): boolean

Package:City City

-name: String -departures: List<Flight>

-arrivals: List<Flight>

+City(name: String) +getName: String

+addDeparture(departure: Flight): void +addArrival(arrival: Flight): void

+getDepartures(): List<Flight> +getArrivals(): List<Flight>

+toString(): String +equals(obj: Object): boolean

ClientDatabase FlightDatabase -Clients: Map<String, User> -flights: Map<String, Flight> -cities: Map<String, City> -date: DateFormat -dateTime: DateFormat +ClientDatabase() -minLayover milli: long +hasClient(email: String): boolean -maxLayover_milli : long +hasClient(client: User): boolean +addClient(firstname: String, lastname: String, email: String, address: String, +FlightDatabase() ccNumber: String, expiryDate: String): void +hasCity(cityName: String): boolean +addClient(newUser: User): void +hasCity(city: City): boolean +searchClientEmail(email: String): User +addCity(cityName: String): void +addCity(newCity: City): void +editClient(client: User, clientField: ClientField, value: Object): void +uploadClientInfo(path: String): void +getCity(city: String): City +uploadClientInfoStream(is InputStream): void +hasFlight(flightNum: String): boolean +equals(obj: Object): boolean +hasFlight(flight: Flight): boolean +addFlight(newFlight: Flight): void FileIO «enumeration» +getFlight(flightNum: String): Flight SortOrder +FileIO() +searchOrigin(city: City): List<Flight> -travelTime +saveFlightDatabase(flightfile: File): void +searchOrigin(cityName: String): List<Flight> -totalCost +saveClientDatabase(clientFile: File): void +searchDestination(city: City): List<Flight> +readFlightDatabase(flightFile: File): void -unsorted +searchDestination(cityName: String): List<Flight> +readClientDatabase(flightFile: File): void +editFlight(flight: Flight, flightfield: FlightFields, value: Object): void Package: flight +editFlight(flightNumber: String, flightField: FlightFields, Exception value: Object): void +uploadFlightInfo(path: String): void +uploadFlightInfoStream(is: InputStream): void +searchItineraries(origin: City, Destination: City, ClientExistsException NoSuchFlightException departureDate: Date):List<FlightItinerary> +NoSuchFlightException() +ClientExistsException() +sort(unsorted: List<FlightItinerary>, sortOrder: SortOrder):List<FlightItinerary> +cloneltinerary(original: Flightltinerary): Flightltinerary NoSuchClientException +addTime(start: Calendar, delta: Duration): Calendar InvalidFileException +NoSuchClientException() +equals(obj: Object): boolean +InvalidFileException() +getItineraries(date: String, origin: String, destination: String) :List<String> +getItinerariesSortedByCost(date: String, origin: String, destination: String) NoSuchCityException InvalidInputException :List<String> +NoSuchCityException() +InvalidInputException() +getItinerariesSortedByTime(date: String, origin: String, destination: String) :List<String> PreexistingCitytException PreexistingFlightException Constants +PreexistingCityException() +PreexistingFlightException() -date: DateFormat -dateTime: DateFormat -flightDatabase: FlightDatabase -clientDatabase: ClientDatabase

-fileManager: FileIO
-displayFormat: String
-minLayover_Mins: int
-minLayover_Hours: int
-maxLayover_Hours: int
-maxLayover_Hours: int

Package:Database «abstract» Database

