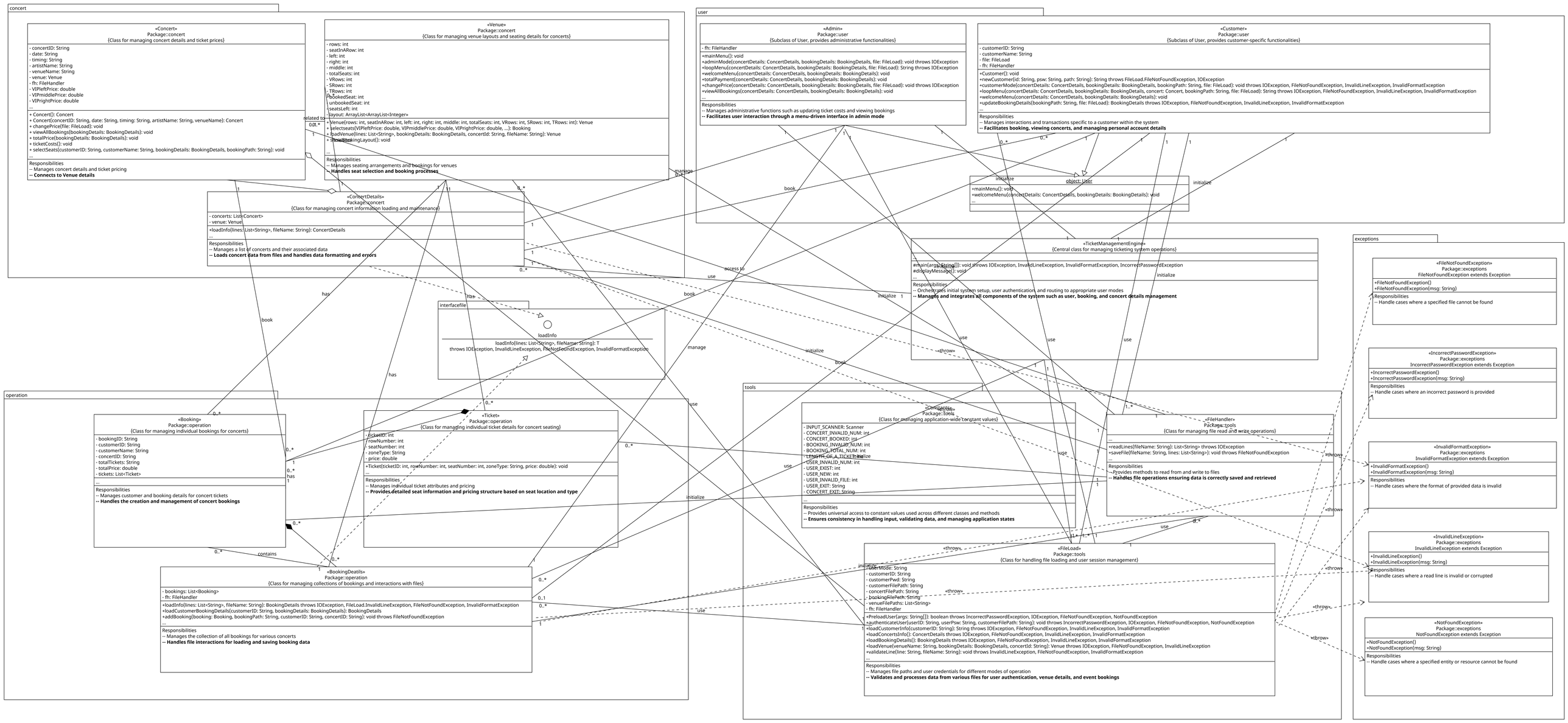
concert
«Concert»
Package::concert
{Class for managing concert details and ticket prices}
- concertID: String
- date: String
- timing: String
- artistName: String
- venueName: String
- venue: Venue
- fh: FileHandler
- VIPleftPrice: double
- VIPmiddlePrice: double
- VIPrightPrice: double
...
+ Concert(): Concert
+ Concert(concertID: String, date: String, timing: String, artistName: String, venueName): Concert
+ changePrice(file: FileLoad): void
+ viewAllBookings(bookingDetails: BookingDetails): void
+ totalPrice(bookingDetails: BookingDetails): void
+ ticketCosts(): void
+ selectSeats(customerID: String, customerName: String, bookingDetails: BookingDetails, bookingPath: String): void
...
Responsibilities
-- Manages concert details and ticket pricing
-- Connects to Venue details
«Venue»
Package::concert
{Class for managing venue layouts and seating details for concerts}
- rows: int
- seatInARow: int
- left: int
- right: int
- middle: int
- totalSeats: int
- VRows: int
- SRows: int
- TRows: int
- bookedSeat: int
- unbookedSeat: int
- seatsLeft: int
- layout: ArrayList<ArrayList<Integer>
+ Venue(rows: int, seatInARow: int, left: int, right: int, middle: int, totalSeats: int, VRows: int, SRows: int, TRows: int): Venue
+ selectseats(VIPleftPrice: double, VIPmiddlePrice: double, VIPrightPrice: double, ...): Booking
+ loadVenue(lines: List<String>, bookingDetails: BookingDetails, concertId: String, fileName: String): Venue
+ showBookingLayout(): void
...
Responsibilities
-- Manages seating arrangements and bookings for venues
-- Handles seat selection and booking processes
«ConcertDetails»
Package::concert
{Class for managing concert information loading and maintenance}
- concerts: List<Concert>
- venue: Venue
+loadInfo(lines:List<String>, fileName: String): ConcertDetails
...
Responsibilities
-- Manages a list of concerts and their associated data
-- Loads concert data from files and handles data formatting and errors
interfacefile
loadInfo
loadInfo(lines: List<String>, fileName: String): T
throws IOException, InvalidLineException, FileNotFoundException, InvalidFormatException
user
«Admin»
Package::user
{Subclass of User, provides administrative functionalities}
- fh: FileHandler
+mainMenu(): void
+adminMode(concertDetails: ConcertDetails, bookingDetails: BookingDetails, file: FileLoad): void throws IOException
+loopMenu(concertDetails: ConcertDetails, bookingDetails: BookingDetails, file: FileLoad): String throws IOException
+welcomeMenu(concertDetails: ConcertDetails, bookingDetails: BookingDetails): void
+totalPayment(concertDetails: ConcertDetails, bookingDetails: BookingDetails): void
+changePrice(concertDetails: ConcertDetails, bookingDetails: BookingDetails, file: FileLoad): void throws IOException
+viewAllBookings(concertDetails: ConcertDetails, bookingDetails: BookingDetails): void
...
Responsibilities
-- Manages administrative functions such as updating ticket costs and viewing bookings
-- Facilitates user interaction through a menu-driven interface in admin mode
«Customer»
Package::user
{Subclass of User, provides customer-specific functionalities}
- customerID: String
- customerName: String
- file: FileLoad
- fh: FileHandler
+Customer(): void
+newCustomer(id: String, psw: String, path: String): String throws FileLoad.FileNotFoundException, IOException
+customerMode(concertDetails: ConcertDetails, bookingDetails: BookingDetails, bookingPath: String, file: FileLoad): void throws IOException, FileNotFoundException, InvalidLineException, InvalidFormatException
+loopMenu(concertDetails: ConcertDetails, bookingDetails: BookingDetails, concert: Concert, bookingPath: String, file: FileLoad): String throws IOException, FileNotFoundException, InvalidLineException, InvalidFormatException
+welcomeMenu(concertDetails: ConcertDetails, bookingDetails: BookingDetails): void
+updateBookingDetails(bookingPath: String, file: FileLoad): BookingDetails throws IOException, FileNotFoundException, InvalidLineException, InvalidFormatException
...
Responsibilities
-- Manages interactions and transactions specific to a customer within the system
-- Facilitates booking, viewing concerts, and managing personal account details
initialize
object: User
+mainMenu(): void
+welcomeMenu(concertDetails: ConcertDetails, bookingDetails: BookingDetails): void
...
«TicketManagementEngine»
{Central class for managing ticketing system operations}
...
#main(args: String[]): void throws IOException, InvalidLineException, InvalidFormatException, IncorrectPasswordException
#displayMessage(): void
...
Responsibilities
-- Orchestrates initial system setup, user authentication, and routing to appropriate user modes
-- Manages and integrates all components of the system such as user, booking, and concert details management
«throw»
exceptions
«FileNotFoundException»
Package::exceptions
FileNotFoundException extends Exception
+FileNotFoundException()
+FileNotFoundException(msg: String)
Responsibilities
-- Handle cases where a specified file cannot be found
«IncorrectPasswordException»
Package::exceptions
IncorrectPasswordException extends Exception
+IncorrectPasswordException()
+IncorrectPasswordException(msg: String)
Responsibilities
-- Handle cases where an incorrect password is provided
«InvalidFormatException»
Package::exceptions
InvalidFormatException extends Exception
+InvalidFormatException()
+InvalidFormatException(msg: String)
Responsibilities
-- Handle cases where the format of provided data is invalid
«InvalidLineException»
Package::exceptions
InvalidLineException extends Exception
+InvalidLineException()
+InvalidLineException(msg: String)
Responsibilities
-- Handle cases where a read line is invalid or corrupted
«NotFoundException»
Package::exceptions
NotFoundException extends Exception
+NotFoundException()
+NotFoundException(msg: String)
Responsibilities
-- Handle cases where a specified entity or resource cannot be found
operation
«Booking»
Package::operation
{Class for managing individual bookings for concerts}
- bookingID: String
- customerID: String
- customerName: String
- concertID: String
- totalTickets: String
- totalPrice: double
- tickets: List<Ticket>
...
Responsibilities
-- Manages customer and booking details for concert tickets
-- Handles the creation and management of concert bookings
«Ticket»
Package::operation
{Class for managing individual ticket details for concert seating}
- ticketID: int
- rowNumber: int
- seatNumber: int
- zoneType: String
- price: double
+Ticket(ticketID: int, rowNumber: int, seatNumber: int, zoneType: String, price: double): void
...
Responsibilities
-- Manages individual ticket attributes and pricing
-- Provides detailed seat information and pricing structure based on seat location and type
«BookingDeatils»
Package::operation
{Class for managing collections of bookings and interactions with files}
- bookings: List<Booking>
- fh: FileHandler
+loadInfo(lines: List<String>, fileName: String): BookingDetails throws IOException, FileLoad.InvalidLineException, FileNotFoundException, InvalidFormatException
+loadCustomerBookingDetails(customerID: String, bookingDetails: BookingDetails): BookingDetails
+addBooking(booking: Booking, bookingPath: String, customerID: String, concertID: String): void throws FileNotFoundException
...
Responsibilities
-- Manages the collection of all bookings for various concerts
-- Handles file interactions for loading and saving booking data
tools
«Constants»
Package::tools
{Class for managing application-wide constant values}
- INPUT_SCANNER: Scanner
- CONCERT_INVALID_NUM: int
- CONCERT_BOOKED: int
- BOOKING_INVALID_NUM: int
- BOOKING_TOTAL_NUM: int
- LENGTH_OF_A_TICKET: int
- USER_INVALID_NUM: int
- USER_EXIST: int
- USER_NEW: int
- USER_INVALID_FILE: int
- USER_EXIT: String
- CONCERT_EXIT: String
...
Responsibilities
-- Provides universal access to constant values used across different classes and methods
-- Ensures consistency in handling input, validating data, and managing application states
«FileHandler»
Package::tools
{Class for managing file read and write operations}
...
+readLines(fileName: String): List<String> throws IOException
+saveFile(fileName: String, lines: List<String>): void throws FileNotFoundException
...
Responsibilities
-- Provides methods to read from and write to files
-- Handles file operations ensuring data is correctly saved and retrieved
«FileLoad»
Package::tools
{Class for handling file loading and user session management}
- userMode: String
- customerID: String
- customerPwd: String
- customerFilePath: String
- concertFilePath: String
- bookingFilePath: String
- venueFilePaths: List<String>
- fh: FileHandler
+PreloadUser(args: String[]): boolean throws IncorrectPasswordException, IOException, FileNotFoundException, NotFoundException
+authenticateUser(userID: String, userPsw: String, customerFilePath: String): void throws IncorrectPasswordException, IOException, FileNotFoundException, NotFoundException
+loadCustomerInfo(customerID: String): String throws IOException, FileNotFoundException, InvalidLineException, InvalidFormatException
+loadConcertsInfo(): ConcertDetails throws IOException, FileNotFoundException, InvalidLineException, InvalidFormatException
+loadBookingDetails(): BookingDetails throws IOException, FileNotFoundException, InvalidLineException, InvalidFormatException
+loadVenue(venueName: String, bookingDetails: BookingDetails, concertId: String): Venue throws IOException, FileNotFoundException, InvalidLineException
+validateLine(line: String, fileName: String): void throws InvalidLineException, FileNotFoundException, InvalidFormatException
...
Responsibilities
-- Manages file paths and user credentials for different modes of operation
-- Validates and processes data from various files for user authentication, venue details, and event bookings
related to
manage
book
access to
use
has
initialize
«throw»