Accommodation Reservation Management System

Verson 1.0

Date: 07/15/2025

From: UMGC Inc

For: BestTech Inc

Contract #: ....

Change History

Table of Contents

1. Overview
   1.1 Objectives
   1.2 Scope
   1.3 Background Information
   1.4 Assumptions
   1.5 Document Structure

2. Classes, Attributes, and Methods
   2.1 TestReservationSystem class
   2.2 Manager class
   2.3 Account
   2.4 Abstract Reservation class
   2.5 HotelReservation class
   2.6 CabinReservation class
   2.7 HouseReservation class

3. Design Considerations
   3.1 Data File: JSON vs XML
   3.2 Line Parsers vs Regular Expression (Regex) vs Jackson Documentation
   3.3 Exceptions Methods
   3.4 Database vs Flat Files (JSON, XML, Text Files)

4. Test Results
   4.1 Testing Methodologies (User Interface vs JAVA Testing Methods)
   4.2 Test Case 1
   4.3 Test Case 2
   4.4 Test Case 3
   4.5 Test Case 4

5. Conclusion

1. OVERVIEW

1.1. Objectives

Design and build a prototype stand-alone application tailored to manage user accounts and their associated accommodation reservations. The application will act as a middleware component, interfacing with an external user interface (UI) that handles user input and output, while the application logic and data operations reside in the system being designed. It will implement file-based persistence using structured data files—such as JSON—to save and retrieve information efficiently, without relying on an external database. The application will support modeling of three distinct types of lodging reservations: hotel rooms, cabins, and houses. Each reservation type will inherit shared attributes from a base reservation class, while maintaining its own unique characteristics and constraints. This approach supports code reusability, easier maintenance, and clear differentiation of business logic specific to each lodging type. The objective is to ensure that all data flows—from account creation to reservation management—are logically encapsulated, error-resilient, and scalable for future upgrades.

1.2. Scope

The system will manage account information and lodging reservations stored in flat files. The product is standalone software that must have account and reservation management functionalities. But the application must have three types of reservation such as home, hotel and cabin reservations. It will be used by people who are looking for temporary stays. The main purpose is to facilitate the reservation so that potential customers do not have to call to check and or make a reservation. And the employees of the company that owns the product can efficiently manage the lodge. The company will benefit from collecting important data about customers, reservations, etc. The data will be used to generate information so that the company can make good decisions. The first step is to create the application and then make it available on distribution platforms for download. Beyond the scope of the product, software experts will attempt to create a user interface and install it on a cloud platform.

1.3. Background Information

The system is intended for use by a business that is currently operating without a formalized digital infrastructure for managing accommodation reservations. Initially, the solution will serve as a local file-based prototype to help stakeholders validate the core functionality and workflows of the system. This approach allows for quicker deployment, lower initial costs, and easier debugging and testing during the early stages of development. The prototype will store data in JSON files to maintain human readability and ensure compatibility across platforms. Over time, as the needs of the business grow and demand for scalability increases, the system is designed to migrate to a more robust, server-based architecture utilizing a relational or NoSQL database. This future migration will support enhanced security, concurrent user access, and integration with broader enterprise systems. Thus, the current solution lays the groundwork for a smooth transition into a fully integrated digital reservation management system.

1.4. Assumptions

- UI handles all user interactions and passes valid data.
- Reservation files and account files will follow naming conventions and reside in a hard-coded path.

1.5. Document Structure

This document provides an in-depth outline of the Accommodation Reservation Management System software design. It begins by describing the system's objectives, scope, and assumptions, and offering the necessary background context. The architecture of the software is elaborated, covering each class in detail including attributes and methods, their relationships,

and the inheritance structure for reservation types. File storage methodologies are thoroughly explained, highlighting directory organization, file naming conventions, and parsing approaches including the use of JSON and relevant libraries. Design considerations address platform compatibility, data integrity, and modular coding practices. The document also includes a comprehensive explanation of exception handling, justifying the inclusion of custom exceptions with practical scenarios. Testing strategies are described through a suite of test cases that simulate real-world scenarios, demonstrating the validity and robustness of the system. Finally, it discusses evaluation metrics, challenges encountered, lessons learned and proposed future enhancements.

2. CLASSES, ATTRIBUTES, METHODS

Class Name: Account
Class Description/Purpose: Represents a customer's account in the reservation system. It stores contact and reservation information associated with the user, such as account number, address, phone, email, and reservation numbers.
Class Modifiers: public
Class Inheritance: None
Class Attributes:

- String acctNumber – user account number
- String acctAddress – user address
- String phoneNumber – user phone number
- String email – user email
- ArrayList<String> resvNumbers – list of reservation numbers tied to user account

Exceptions Thrown:

- JsonProcessingException – when fails to convert an object to a string of json format
- IOException – throws when I/O operation has failed

Class Methods:

- public Account() - Default constructor.
  Used to convert from json to object and vice versa

- public Account(String acctNumber, String acctAddress, String phoneNumber, String email, ArrayList<String> resvNumbers) - Parameterized constructor.
  - Assign all the account attributes to the parameter's values

- public void draftReservation(Reservation resv)
  Add draft reservation to the file by using an object of ObjectMapper with writeValue method.

If reservation is a hotel, then display with HotelReservation attributes
If reservation is a cabin, then display with CabinReservation attributes
If reservation is a house, then display with HouseReservation attributes

- public Reservation updateReservation(Reservation resv)
  enter 1 to update lodging address
  enter 2 to update lodging email
  enter 3 to update start date
  enter 4 to update number of nights
  enter 5 to update number of bedrooms
  enter 6 to update number of bed
  enter 7 to update number of bathroom
  enter 8 to update lodging size
  otherwise return the reservation object

- public Reservation updateHotelResv(Reservation resv)
  enter 1 to update lodging address
  enter 2 to update lodging email
  enter 3 to update start date
  enter 4 to update number of nights
  enter 5 to update lodging size
  otherwise return the reservation object

- public String getAcctNumber()
  Return account number

- public void setAcctNumber(String acctNumber)
  Assign account number to acctNumber value
-

- public String getAcctAddress()
  Return user address

- public void setAcctAddress(String acctAddress)
  Assign account address to acctAddress value

- public String getPhoneNumber()
  Return user phone number

- public void setPhoneNumber(String phoneNumber)
  Assign user phone number to phoneNumber value

- public String getEmail()
  Return user email

- public void setEmail(String email)
  Assign user email to email value

- public ArrayList<String> getResvNumbers()
  Returns the list of reservation numbers tied to user account

- public void getResvNumbers(ArrayList<String resvNumbers>)
  Assign resvNumbers which is an arraylist object to resvNumbers

- Public String to String()
  Convert account object to String of json format, then return

Class Name: Manager

Class Description/Purpose: Acts as the controller of the system. Responsible for managing account and reservation data including creation, deletion, loading, and saving. Interfaces with file system directories and handles user input.

Class Modifiers: public

Class Inheritance: None

Class Attributes:

- Account account – user account
- Reservation resv – a user reservation
- static Scanner scan – Shared scanner instance for user input.
- ArrayList<Account> acctList – Stores all account objects.
- ArrayList<Reservation> resvs – Stores all reservations.
- File tempDirectory – Temporary directory for file operations.
- final File myDirectory – Root directory for account data.
- String acctNumber
- static final ObjectMapper objectMapper – Shared object mapper for JSON operations.

Exceptions Thrown:

- IOException – for file operations
- JsonProcessingException – during JSON serialization/deserialization
- DuplicateObjectException – for preventing duplicate entries

- IllegalLoadException – for catching invalid file or data load attempts

Class Methods:

- public Manager()
  initializes directory
  loads existing account list
  Loads existing reservation list

- public void createAccount(String acctNumber)
  Creates a new account if not already existing.
  Creates a new Account with the given account number.
  Adds the new account to acctList and saves it as a JSON file.

- public void getAllAccounts()
  Loads all account .json files from the myDirectory folder into memory (acctList).
  Deserializes each file into an Account object.

- Public void getAccount()
  Prompts the user to enter an account number and attempts to find
  print it from acctList.

- public void getAllReservations()
  Scanning directories for files starting with "C", "H", or "O".
  Reading and converting them to the correct subclass of Reservation.

- public void makeReservation(String resvNumber)
  It first checks if the reservation number already exists in the account's list of reservations.
  If it does, it throws a DuplicateObjectException and stops the process.
  The user is asked to specify the type of accommodation.
  The code accepts only "H", "C", or "O" (case-insensitive) as valid inputs.
  A hotel, cabin, or house reservation object is created by calling a specific helper method (getHotelResvInput, etc.).
  These methods collect all needed info from the user
  The reservation is stored in memory.
  The account object is updated to include this reservation.
  The updated account is written to a .json file for persistence.

- public void updateReservation()

Lists all reservations associated with the current account to help the user choose which one to update.
Prompts the user to input a valid reservation number
Repeats until a valid reservation number from the account is entered.
Finds the corresponding Reservation object in memory using a case-insensitive match.

Throws an IllegalLoadException if the reservation cannot be found.
checks if the reservation is in the future and the status is “draft”
Determines whether to update as a hotel or other reservation.
Calls the appropriate method in the Account class, which handles the specific update logic.
Writes the updated reservation back to disk using ObjectMapper for persistence.
Any errors while fetching the reservation are caught and shown without crashing the app.

- Public void cancelReservation()

  Lists all reservations associated with the current account to help the user choose which one to update.
  Prompts the user to input a valid reservation number
  Repeats until a valid reservation number from the account is entered.
  Finds the corresponding Reservation object in memory using a case-insensitive match.
  Throws an IllegalLoadException if the reservation cannot be found.
  if the reservation is in the future and the status is “draft”
  Update status to “cancel” and update price to current lodge price
  Writes the updated reservation back to disk using ObjectMapper for persistence.
  Any errors while fetching the reservation are caught and shown without crashing the app

- Public void completeReservation()

  Lists all reservations associated with the current account to help the user choose which one to update.
  Prompts the user to input a valid reservation number
  Repeats until a valid reservation number from the account is entered.

Finds the corresponding Reservation object in memory using a case-insensitive match.
Throws an IllegalLoadException if the reservation cannot be found.
if the reservation is in the future and the status is “draft”
Update status to “complete” and update price to current lodge price
Writes the updated reservation back to disk using ObjectMapper for persistence.
Any errors while fetching the reservation are caught and shown without crashing the app

- Public void pricePerNight()

  Lists all reservations associated with the current account to help the user choose which one to update.
  Prompts the user to input a valid reservation number
  Repeats until a valid reservation number from the account is entered.
  Finds the corresponding Reservation object in memory using a case-insensitive match.
  Throws an IllegalLoadException if the reservation cannot be found.
  if the reservation is in the future and the status is “draft”
  Calculate and display the lodge unit price without storing it in the price
  Any errors while fetching the reservation are caught and shown without crashing the app

- Public void totalReservationPrice()

  Lists all reservations associated with the current account to help the user choose which one to update.
  Prompts the user to input a valid reservation number
  Repeats until a valid reservation number from the account is entered.
  Finds the corresponding Reservation object in memory using a case-insensitive match.
  Throws an IllegalLoadException if the reservation cannot be found.
  if the reservation is in the future and the status is “draft”
  Calculate the lodge unit price multiplied by the number of nights, then display
  Any errors while fetching the reservation are caught and shown without crashing the app

- private Reservation getHotelResvInput(String resvNumber)
  Prompt for input of all the HotelReservation attributes
  Create HotelReservation object with attributes values as arguments

Return the HotelReservation object

- private Reservation getCabinResvInput(String resvNumber)
  Prompt for input of all the CabinReservation attributes
  Create CabinReservation object with attributes values as arguments
  Return the CabinReservation object

- private Reservation getHouseResvInput(String resvNumber)
  Prompt for input of all the HouseReservation attributes
  Create HouseReservation object with attributes values as arguments
  Return the HouseReservation object

- private ArrayList<Account> readInAllAccount()
  Create a list of account folder with DirectoryStream
  Iterate through the list account folders
  Create a list of json file with the DirectoryStream
  Iterate through the list of json files
  If the json file is an account file, then read the file and add it to the arraylist
  Any errors while fetching the account are caught and shown without crashing the app

- Private ArrayList<Account> readInAllReservation()
  Create a list of account folder with DirectoryStream
  Iterate through the list reservation folders
  Create a list of json file with the DirectoryStream
  Iterate through the list of json files
  If the json file is a reservation file, then read the file and add it to the arraylist
  Any errors while fetching the reservation are caught and shown without crashing the app

- Private void createAccountFolder()
  Create account directory
  Concatenate to the current root directory
  Convert the directory into File
  Save the file with the object mapper with writeValue

- Private void saveAccountIntoFile()
  Concatenate the json file name to created folder directory
  Convert the directory into File

- private void saveResvFile()
  Serializes and saves a single Reservation to disk.
  File is named based on the reservation number.

- private boolean dateComparison(String strDate)
  Parse strDate to LocalDate
  Then parse LocalDate object into Date
  Get the current date
  Convert the current local date obtained into Date
  If current is before start date return true
  Otherwise return false

Class Name: Reservation

Class Description/Purpose: A generic base class for different types of lodging reservations. Encapsulates common reservation details.

Class Modifiers: public

Class Inheritance: None. It serves as a superclass for CabinReservation, HotelReservation, and HouseReservation

Class Attributes:

- String acctNum – Associated account number
- String resvNum – Reservation number
- String lodgingAddress – lodging address
- String lodgingEmail – lodging address
- String startDate – start date
- int numNight – number of nights
- int numBed – number of beds
- int numBedroom – number of bedrooms
- int numBathroom – number of bathrooms
- int lodgingSize – Size of the lodging (e.g., in sq. ft.)

- float price - the lodge price
- String status – Reservation status: "draft", "completed", "cancelled"

Exceptions Thrown: None

Class Methods:

- public Reservation()
  Default constructor where account number, status and price are initialized to default values

- public Reservation(String acctNum, String resvNum, String lodgingAddress, String lodgingEmail, String startDate, int numNight, int numBed, int numBedroom, int numBathroom, int lodgingSize, float price, String status)
  Assign all the reservation attributes to the parameter values

- public String getAcctNum()
  Return user account number
- public void setAcctNum(String acctNum)
  Set user account number to acctNum
- public String getResvNum()
  Return reservation number
- public void setResvNum(String resvNum)
  set reservation number to resvNum
- public String getLodgingAddress()
  Return lodging address
- public void setLodgingAddress(String lodgingAddress)
  Set lodging address to lodgingAddress
- public String getLodgingEmail()
  Return lodging email
- public void setLodgingEmail(String lodgingEmail)
  Set lodging email to lodgingEmail
- public String getStartDate()
  Return start date
- public void setStartDate(String startDate)
  Set start date to startDate
- public int getNumNight()
  Return number of nights
- public void setNumNight(int numNight)
  Set number of nights to numNight
- public int getNumBed()
  Return number of beds
- public void setNumBed(int numBed)

Set number of beds to numBed

- public int getNumBedroom()
  Return number of bedroom
- public void setNumBedroom(int numBedroom)
  Set number of bedrooms to numBed
- public int getNumBathroom()
  Return number of bathroom
- public void setNumBathroom(int numBathroom)
  Set number of bathroom to numBathroom
- public int getLodgingSize()
  Return lodging size
- public void setLodgingSize(int lodgingSize)
  Set lodging size to lodgingSize
- public float getPrice()
  return price
- public void setPrice(float price)
- Set price to price
- public String getStatus()
  Return status
- public void setStatus(String status)
  Set status to status
- Public String toString()
  Return dummy string
- Public void updatePrice()
  Calculate base lodge price

Class Name: HotelReservation

Class Description/Purpose: Specialized subclass of Reservation representing a reservation at a hotel. Adds hotel-specific features such as whether a kitchenette is included.

Class Modifiers: public

Class Inheritance: Inherits from Reservation

Class Attributes:

- private boolean kitchenette – Indicates the presence of a kitchenette in the hotel room.

Exceptions Thrown:

None directly; may inherit handling from the Reservation class for JSON and I/O operations.

Class Methods:

- public HotelReservation() – Default constructor.
- public HotelReservation(String acctNumber, String resvNumber, String lodgingAddress, String lodgingEmail, String startDate, int numNight, int numBed, int numBedroom, int numBathroom, int size, float price, String status, boolean kitchenette) – Parameterized constructor that initializes both inherited and new attributes.
- public boolean isKitchenette() – Getter for kitchenette.
- public void setKitchenette(boolean kitchenette) – Setter for kitchenette.

Class Name: CabinReservation
Class Description/Purpose: Represents a specific type of reservation for a cabin. It extends the generic Reservation class with cabin-specific details like the presence of a full kitchen and loft type.
Class Modifiers: public
Class Inheritance: Inherits from Reservation
Class Attributes:

- private boolean fullKitchen – Indicates whether the cabin includes a full kitchen.
- private String loft – Describes the type or availability of a loft in the cabin.

Exceptions Thrown:

None directly, but may inherit or indirectly involve exceptions from the parent Reservation class (e.g., during file or JSON processing).

Class Methods:

- public CabinReservation() – Default constructor.
- public CabinReservation(String acctNumber, String resvNumber, String lodgingAddress, String lodgingEmail, String startDate, int numNight, int numBed, int numBedroom, int numBathroom, int size, float price, String status, boolean fullKitchen, String loft) – Parameterized constructor that also calls the superclass constructor.
- public boolean getFullKitchen() – Getter for fullKitchen.
- public void setFullKitchen(boolean fullKitchen) – Setter for fullKitchen.
- public String getLoft() – Getter for loft.
- public void setLoft(String loft) – Setter for loft.

Class Name: HouseReservation

Class Description/Purpose: Represents a house reservation within the system. This subclass of Reservation includes house-specific features such as the number of floors.
Class Modifiers: public

Class Inheritance: Inherits from Reservation

Class Attributes: int floorNum – Indicates the number of floors in the reserved house.

Exceptions Thrown:
None directly; inherits exception handling (e.g., JSON processing, I/O) from Reservation.

Class Methods:

- public HouseReservation() – Default constructor.
- public HouseReservation(String acctNumber, String resvNumber, String lodgingAddress, String lodgingEmail, String startDate, int numNight, int numBed, int numBedroom, int numBathroom, int size, float price, String status, int floorNum) – Parameterized constructor that initializes inherited and subclass-specific fields.
- public int getFloorNum() – Getter for floorNum.
- public void setFloorNum(int floorNum) – Setter for floorNum.

Class Name: TestReservationSystem

Class Description/Purpose: Acts as the driver class for running and testing the reservation system functionalities. Provides a user interface for selecting and executing various tasks such as creating accounts, retrieving data, and managing reservations.

Class Modifiers: public

Class Inheritance: None

Class Attributes: static Scanner scan – Scanner object for reading user input from the console.

Exceptions Thrown: IOException – For operations that involve file handling in coordination with Manager.

Class Methods:

- public static String numberGenerator()
  Generates a random 9-digit number
  Prefixes it with "A" to format it like an Account ID.
  Printed for reference and returned as a string.

- public static void options(int choice, Manager manager)
  Takes a user's menu selection (choice) and routes it to a specific operation.
  Uses Switch statement to select an option in the menu

- public static void main(String[] args)
  The entry point for the program.
  Likely:
  Initializes a Manager object
  Shows a menu to the user via console
  Calls options() with the selected task number
  Uses Scanner to take input and loop through user actions.

3. DESIGN CONSIDERATIONS

3.1 Data File: JSON vs XML

JSON was chosen for simplicity and integration with most modern parsers. All files are human-readable. The program is set to be built on Windows platform.

- Each folder created based on the account number: ***accounts/A123456789/***
- Account file: ***accounts/A123456789/acc-A123456789.json***
- Reservation files: ***accounts/A123456789/res-HOT12345678.json***

3.2. Line Parsers vs Regular Expression (Regex) vs Jackson Documentation

Jackson Library is used as the documentation to parse and write JSON files.

- **ReadValue** method is used to read data from JSON files and parse them into java objects
- **WriteValue** method is used to convert java objects to JSON format and store them into JSON files
- **WriteValueAsString** method is used to display java object to JSON format

3.3. Exception Methods

Custom Exceptions

- IllegalLoadException(String message)
- DuplicateObjectException(String message)
- IllegalOperationException(String message)

Java built-in exception

- IllegalArgumentException
- IllegalStateException

3.4. Database vs Flat Files (JSON, XML, Text Files)

In this prototype, JSON flat files are used for data storage. This decision was based on the project's early development phase, where quick prototyping, ease of access, and human-readable data formats were prioritized. JSON offers a lightweight and structured way to store account and reservation data, simplifying the development and debugging process. File-based storage also eliminates the need for a complex database setup, reducing system dependencies and enabling the application to run on any machine without additional infrastructure.

4. Test Results

4.1 Testing Methodologies (User Interface vs JAVA Testing Methods)

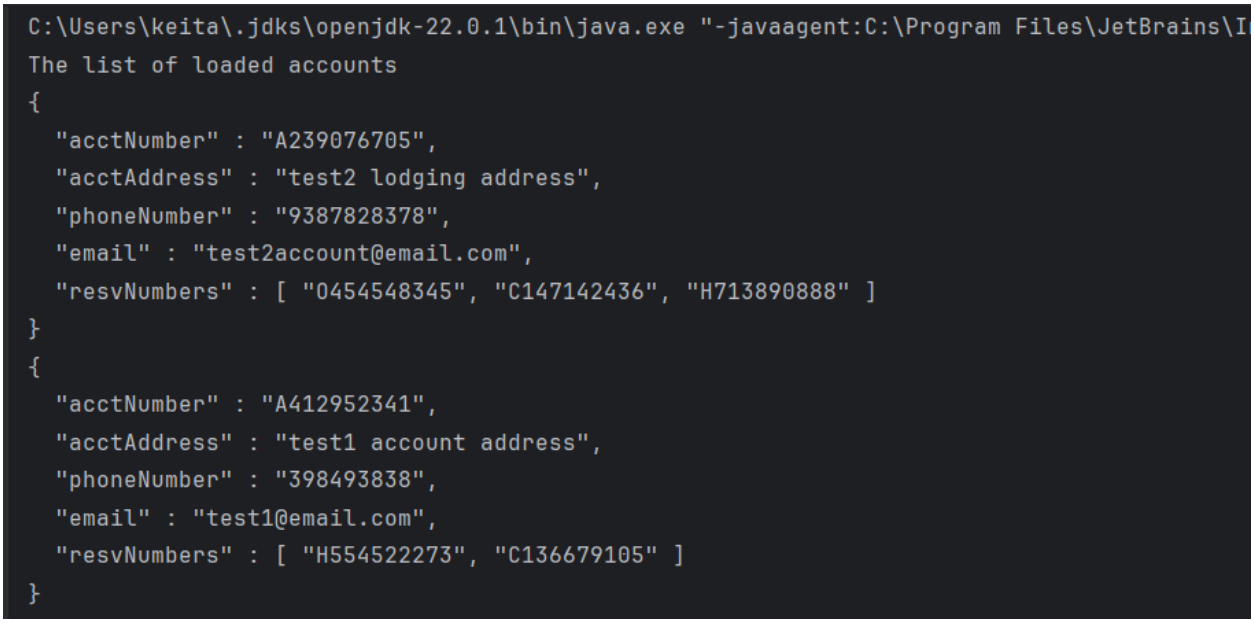
Each test is a separate method and is documented. Tests include:

1. Get the list of loaded accounts from Manager
2. Retrieve a loaded account object that matches a specific account number
3. Add new account object to the list managed by Manager (if account object already exists on add action with the same account number, it is considered an error)
4. Request that Manager updates specific account’s files with data stored in memory
5. Add draft lodging reservation to an account (if reservation object already exists with the same reservation number, it is considered an error)
6. Complete reservation that is associated with an account
7. Cancel reservation that is associated with an account

8. Change reservation values that can be changed (if reservation is cancelled, completed, or for past date, it is considered an error)
9. Request for price per night to be calculated and returned for a specific reservation
10. Request for total reservation price to be calculated and returned for a specific reservation

4.2 Test Case 1: get a list of loaded accounts from manager

- **Input:** When the program, it displays a list of user accounts
- **Output:** list of account displayed on the console
- **Screenshot:**



4.3 Test Case 2: retrieve a loaded account object that matches a specific account number

- **Input**: Enter “2” as selection input to the user prompt
  Enter “A239076705” as account number to the user prompt to get account number
- **Output**: user account retrieved and displayed in JSON format

- **Screenshots:**

```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 2
Enter the account number (A followed by 9 digits) -> : A239076705
{
  "acctNumber" : "A239076705",
  "acctAddress" : "test2 lodging address",
  "phoneNumber" : "9387828378",
  "email" : "test2account@email.com",
  "resvNumbers" : [ "0454548345", "C147142436", "H713890888" ]
}
```

4.4 Test Case 3: create and add a new account to the list of accounts

- **Input**: Enter “3” as selection input to the user prompt
  Enter “test 3 address” as user address input
  Enter “9387438343” as user phone number input
  Enter test@email.com as user email input

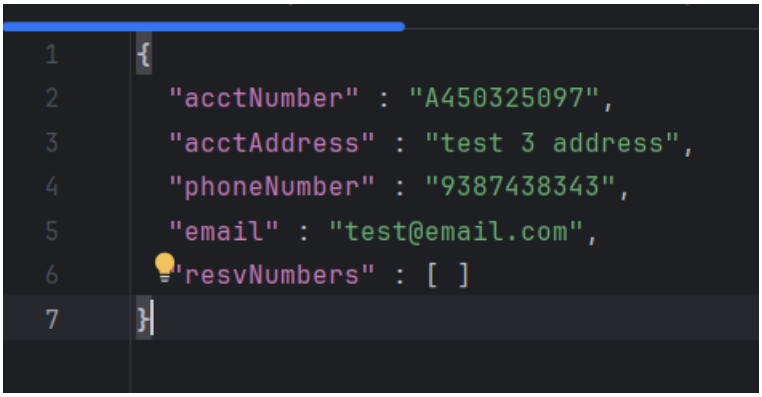
- **Output**: new account created and added to the local storage. JSON file of the account created

- **Screenshots:**

```
Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 3
Generated 9-digit number: 450325097
Enter the address that is associated with the account number:
test 3 address
Enter phone number (should be 10 digits):
9387438343
Enter the email address associated with the account number:
test@email.com
File exists: C:\Users\keita\Projects\LodgingReservation\data\accounts\A450325097\acc-A450325097.json
The account with account number#A450325097 is successfully created
{
  "acctNumber" : "A450325097",
  "acctAddress" : "test 3 address",
  "phoneNumber" : "9387438343",
  "email" : "test@email.com",
  "resvNumbers" : [ ]
}
```

(the user account number is automatically generated before user responds to the prompt)



4.5 Test Case 4

- **Input**: Enter “4” as selection input to the user prompt
  Enter “A450325097” as the account number input for the account number to be updated
  Enter “1” to updated address: test3 address updated
  Enter “3” to update email: test3@email.com updated

- **Output**: updated account displayed on console and JSON file of the account updated

- **Screenshots**

```
Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 4
Enter the account number (A followed by 9 digits) of the account to be updated : A450325097

Please enter a number for the corresponding account data you would like to update or change

Enter a number that corresponds to the account field to be updated
1 to update address
2 to update phone number
3 to update email
Otherwise enter 0, any other number than 1, 2, 3 to end the update
1
Update address --> : test3 address updated
```

```
Update address --> : test3 address updated

Enter a number that corresponds to the account field to be updated
1 to update address
2 to update phone number
3 to update email
Otherwise enter 0, any other number than 1, 2, 3 to end the update
3
Update email --> : test3@email.com updated

Enter a number that corresponds to the account field to be updated
1 to update address
2 to update phone number
3 to update email
Otherwise enter 0, any other number than 1, 2, 3 to end the update
0
The account with account number#A450325097 is successfully updated
{
  "acctNumber" : "A450325097",
  "acctAddress" : "test3 address updated",
  "phoneNumber" : "9387438343",
  "email" : "test3@email.com updated",
  "resvNumbers" : [ ]
}
```

4.6 Test 5: create a draft reservation and add it to the list of reservation

- **Input:** Enter “5” as selection input to the user prompt

- **Output:** created reservation displayed on the console and reservation JSON file
- **Screenshots**

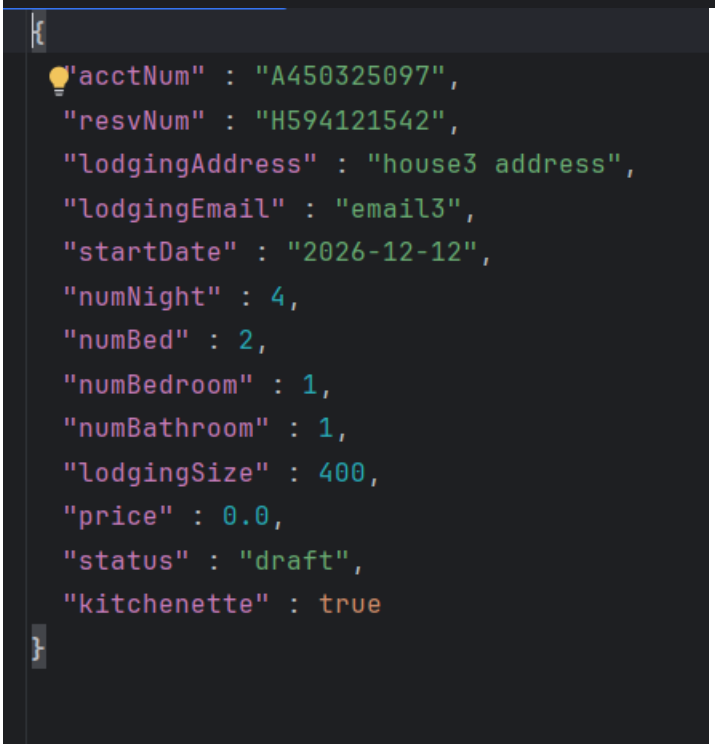
```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 5
Generated 9-digit number: 594121542
There are three types of accommodations: hotel room reservation, cabin reservation, and house reservation
Enter: H/h for hotel - C/c for cabin - Enter O/o for house
h
Enter lodging address --> : house3 address
Enter lodging email --> : email3
Enter lodging start date (yyyy-mm-dd) --> : 2026-12-12
Enter number of night --> : 4
Enter number lodging size --> : 400
Does the lodge have kitchenette? (true/false) --> :true
```

```
Enter lodging address --> : house3 address
Enter lodging email --> : email3
Enter lodging start date (yyyy-mm-dd) --> : 2026-12-12
Enter number of night --> : 4
Enter number lodging size --> : 400
Does the lodge have kitchenette? (true/false) --> :true

Reservation with the account #H594121542 is created successfully
{
  "acctNum" : "A450325097",
  "resvNum" : "H594121542",
  "lodgingAddress" : "house3 address",
  "lodgingEmail" : "email3",
  "startDate" : "2026-12-12",
  "numNight" : 4,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 400,
  "price" : 0.0,
  "status" : "draft",
  "kitchenette" : true
}
```



4.7 Test 6: complete a reservation that is associated with an account

- **Input:** Enter “6” to respond to the prompt for a reservation number
- **Output:** completed reservation displayed on the console and reservation JSON file
- **Screenshots**

```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 6
This account has following reservation numbers: [H594121542]
Enter a reservation number of the list displayed --> : H594121542
Reservation with the account #H594121542 is updated successfully

Choose an option --> : 6
This account has following reservation numbers: [H594121542]
Enter a reservation number of the list displayed --> : H594121542
Reservation with the account #H594121542 is updated successfully
{
  "acctNum" : "A450325097",
  "resvNum" : "H594121542",
  "lodgingAddress" : "house3 address",
  "lodgingEmail" : "email3",
  "startDate" : "2026-12-12",
  "numNight" : 4,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 400,
  "price" : 180.0,
  "status" : "completed",
  "kitchenette" : true
}
```

```
{
  "acctNum" : "A450325097",
  "resvNum" : "H594121542",
  "lodgingAddress" : "house3 address",
  "lodgingEmail" : "email3",
  "startDate" : "2026-12-12",
  "numNight" : 4,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 400,
  "price" : 180.0,
  "status" : "completed",
  "kitchenette" : true
}
```


4.8 Test 7: cancel a reservation that is associated with an account

- **Input:** Enter 2 as input selection to prompt for another account number.
  Enter “A412952341” to retrieve or login to an account number
  Enter 7 to prompt for a reservation number.
  Enter “C136679105” the reservation number to cancel the reservation.

- **Output:** cancelled reservation displayed on the console and reservation JSON file
- **Screenshots**

```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 2
Enter the account number (A followed by 9 digits) -> : A412952341
{
  "acctNumber" : "A412952341",
  "acctAddress" : "test1 account address",
  "phoneNumber" : "398493838",
  "email" : "test1@email.com",
  "resvNumbers" : [ "H554522273", "C136679105" ]
}

Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!


 Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

 Please enter:
 1 to get the list of loaded accounts from Manager
 2 to retrieve a loaded account object that matches a specific account number
 3 to add new account object to the list managed by Manager
 4 to request that Manager updates specific account's files with data stored in memory
 5 to add draft lodging reservation to an account
 6 to complete reservation that is associated with an account
 7 to cancel reservation that is associated with an account
 8 to change reservation values that can be changed
 9 to request for price per night to be calculated and returned for a specific reservation
 10 to request for total reservation price to be calculated and returned for a specific reservation
 Otherwise enter 0 or any non-matching number or character to exit

 Choose an option --> : 7
 This account has following reservation numbers: [H554522273, C136679105]
 Enter a reservation number of the list displayed to cancel--> : C136679105
 Reservation with the account #C136679105 is updated successfully
 {
```

```
Choose an option --> : 7
This account has following reservation numbers: [H554522273, C136679105]
Enter a reservation number of the list displayed to cancel--> : C136679105
Reservation with the account #C136679105 is updated successfully
{
  "acctNum" : "A412952341",
  "resvNum" : "C136679105",
  "lodgingAddress" : "123 cabin1 lodging address",
  "lodgingEmail" : "cabin1@email.com",
  "startDate" : "2025-12-12",
  "numNight" : 9,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 850,
  "price" : 0.0,
  "status" : "cancelled",
  "fullKitchen" : false,
  "loft" : "Y"
}
```


```
{
  "acctNum" : "A412952341",
  "resvNum" : "C136679105",
  "lodgingAddress" : "123 cabin1 lodging address",
  "lodgingEmail" : "cabin1@email.com",
  "startDate" : "2025-12-12",
  "numNight" : 9,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 850,
  "price" : 0.0,
  "status" : "cancelled",
  "fullKitchen" : false,
  "loft" : "Y"
}
```

4.9 Test 8: change reservation value(s) that can be changed

- **Input:** Enter 8 to prompt for another reservation number associated with account # A412952341.
  Enter “H554522273” the reservation number to change some values.
  Enter “1” to update the reservation lodge address.
  Enter “123 house1 lodging address updated” which is the updated address
  Enter “5” to update the reservation lodging size
  Enter “700” as the new lodging size value
  Enter “0” to end the update prompt

- **Output:** changed reservation values displayed on the console and reservation JSON file

- **Screenshots:**

```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 8
This account has following reservation numbers: [H554522273, C136679105]
Enter a reservation number of the list displayed --> : H554522273
Please enter a number for the corresponding account data you would like to update or change

1 for lodging address
2 for lodging email
3 for start date
4 for number of night
5 for lodging size
Press 0 to exit or other to finish

Enter a number that corresponds to the account field to be updated
Otherwise enter 0, any other number or character to end the update
```

```
Otherwise enter 0, any other number or character to end the update
1
Update lodging address --> :
123 house1 lodging address updated

Enter a number that corresponds to the account field to be updated
Otherwise enter 0, any other number or character to end the update
5
Update lodging size --> :
700

Enter a number that corresponds to the account field to be updated
Otherwise enter 0, any other number or character to end the update
0
Reservation with the account #H554522273 is updated successfully
{
  "acctNum" : "A412952341",
  "resvNum" : "H554522273",
  "lodgingAddress" : "123 house1 lodging address updated",
  "lodgingEmail" : "house1@email.com",
  "startDate" : "2026-12-12",
  "numNight" : 8,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 700,
  "price" : 0.0,
  "status" : "draft",
  "kitchenette" : true
}
```

```
{
  "acctNum" : "A412952341",
  "resvNum" : "H554522273",
  "lodgingAddress" : "123 house1 lodging address updated",
  "lodgingEmail" : "house1@email.com",
  "startDate" : "2026-12-12",
  "numNight" : 8,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 700,
  "price" : 0.0,
  "status" : "draft",
  "kitchenette" : true
}
```


4.10 Test 9: request the price per night of a reservation

- **Input:** Enter 9 to prompt for a reservation number associated with account # A412952341.
  Enter “H554522273” as the reservation number to request the price per night.

- **Output:** price per night of the reservation displayed on the console and reservation JSON file

- **Screenshots:**

```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 9
This account has following reservation numbers: [H554522273, C136679105]
Enter a reservation number of the list displayed --> : H554522273
The reservation with identifiable number #H554522273 has price per night : $180.0

Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!
```


```
{
  "acctNum" : "A412952341",
  "resvNum" : "H554522273",
  "lodgingAddress" : "123 house1 lodging address updated",
  "lodgingEmail" : "house1@email.com",
  "startDate" : "2026-12-12",
  "numNight" : 8,
  "numBed" : 2,
  "numBedroom" : 1,
  "numBathroom" : 1,
  "lodgingSize" : 700,
  "price" : 0.0,
  "status" : "draft",
  "kitchenette" : true
}
```


(The reservation price does not get updated. The price is just request based on the lodging size)

4.11 Test 10: request the total reservation price of a reservation

- **Input:** Enter 10 to prompt for a reservation number associated with account # A412952341.
  Enter “H554522273” as the reservation number to request the total reservation price.

- **Output:** total reservation price of the reservation displayed on the console and reservation JSON file

- **Screenshots:**

```
Welcome to DONALD KEITA's Rental Reservation System for Hotel, House, and Cabin!

Please enter:
1 to get the list of loaded accounts from Manager
2 to retrieve a loaded account object that matches a specific account number
3 to add new account object to the list managed by Manager
4 to request that Manager updates specific account's files with data stored in memory
5 to add draft lodging reservation to an account
6 to complete reservation that is associated with an account
7 to cancel reservation that is associated with an account
8 to change reservation values that can be changed
9 to request for price per night to be calculated and returned for a specific reservation
10 to request for total reservation price to be calculated and returned for a specific reservation
Otherwise enter 0 or any non-matching number or character to exit

Choose an option --> : 10
This account has following reservation numbers: [H554522273, C136679105]
Enter a reservation number of the list displayed --> : H554522273
The reservation with identifiable number #H554522273 has total reservation price : $1440.0
```

(The reservation price does not get updated. The total reservation price is just request based on the lodging size and the number of nights)

Conclusion

Lessons learned

- Early validation of input and file structure avoids runtime issues
- Specific messaging for an exception helps debugging

Challenges

- Managing reservation rules across subtype classes
- Designing consistent file read/write logic

Improvements

- Integrate path configurability instead of hardcoding