**Architecture hierarchy Tree**

* SRC
  + Main
    - Resources
    - Java.learn.mastery
      * App (main)
        + Models:

Guests

Hosts

Reservations

* + - * + Data:

DataException

HostFileRepository

GuestFileRepository

ReservationFileRepository

HostRepository

GuestRepository

ReservationRepository

* + - * + Domain

GuestService

HostService

ReservationService

Response

Results

* + - * + UI

ConsoleIO

Controller

MainMenuOptions

View

* + Testing
    - Java.learn.mastery
      * Data
        + HostFileRepositoryTest
        + GuestFileRepositoryTest
        + ReservationFileRepositoryTest
        + HostRepositoryDouble
        + GuestRepositoryDouble
        + ReservationRepositoryDouble
      * Domain
        + GuestServiceTest
        + HostServiceTest
        + ReservationServiceTest

**Primary Functions**

* View a selection of reservations
* Add a new upcoming reservation
* Update a reservation to an upcoming date
* Remove an upcoming reservation

**Reservation Data**

* **ID**: Integer used to identify the reservation in file
* **Start Date**: Local Date. The starting date of the reservation.
* **End Date**: Local Date. The end date of the reservation.
* **Guest** – Guest object ID. This is the guest the reservation is associated with
* **Host** – Host Object. Used to retrieve the Big Decimal values to calculate the cost of Stay
* **Cost of Stay**: Total cost of stay. Big Decimal calculated based on the Start and End Dates value and associated host.

**Reservation Validation**

* **Start Date**: Local Date that has yet to occur. May not overlap with another reservation date range
* **End Date**: Local Date that occurs after the start date and has yet to occur. May not overlap with another reservation date range.
* **Guest** – Guest object. Must existing in the guest repository
* **Host** – Host object. Must existing in the host repository
* **Cost of Stay**: Big Decimal value. Must be a positive value.

**Methods by class outline**

* **Models**:
  + Guests
    - String Id (UUID)
    - String Lastname
    - String email
    - String address
    - String city
    - String state
    - String postal\_code
  + Hosts
    - String Id (UUID)
    - String Lastname
    - String email
    - String address
    - String city
    - String state
    - String postal\_code
    - BigDecimal standardRate,
    - BigDecimal weekenRate (fri + sat night)
  + Reservations
    - String/int ID (Id per host)
    - LocalDate Startdate
    - LocalDate Enddate
    - String guest\_id
    - BigDecimal totalCost
* **Data**:
  + DataException (Custom exception)
    - Extend Exceptions
    - Construct(String, throw)
  + HostFileRepository
    - FILEPATH
    - List<> findAll()
      * Finds all Hosts from file
    - Host findById(String)
      * Finds Host with Matching Id field
    - Host FindByEmail(String)
      * Finds Host with matching email field
    - Host deserialize(String[])
      * Converts String into Host from file line
  + GuestFileRepository
    - FILEPATH
    - List<> findAll()
      * Finds all Guests from file
    - Guest findById(int)
      * Finds Guest with Matching Id field
    - Guest FindByEmail(String)
      * Finds Guest with Matching email field
    - Guest deserialize(String[])
      * Converts String into Guest from file line
  + ReservationFileRepository
    - FILEPATH
    - Void setHost(Host)
      * Used to reconfirm repository path
    - Host getHost()
    - String getFilePath()
    - Reservation findById(int)
    - List<Reservation> findAll()
      * Finds all reservations in file
    - Void WriteToFile(List<Reservation>)
      * Updates the file repository with reservations
    - List<Reservation> findAllByGuestId(int)
      * Finds all reservations in file containing the target guest id
    - Reservation add(reservation)
      * Add a reservation
    - Boolean update(Reservation)
      * Update a reservation
    - Boolean delete(int)
      * Deletes a reservation
    - String Serialize(Reservation)
      * Converts a reservation into a string
    - Reservation Deserialize(String[])
      * Converts string into reservation from file line
  + HostRepository (Interface)
    - List<Host> FindAll()
    - Host FindById(String)
    - Host FindByEmail(String)
  + GuestRepository (Interface)
    - List< Guest> FindAll()
    - Guest FindById(int)
    - Guest FindByEmail(String)
  + ReservationRepository (Interface)
    - Void setHost(Host)
    - Host getHost()
    - Reservation findById(intID)
    - List< Reservation > findAllByGuestId(int)
    - List< Reservation > Findall()
    - Reservation Add(Reservation)
    - Boolean Update(Reservation)
    - Boolean delete(int)
* **Domain**:
  + GuestService
    - Result<Guest> ValidateGuest(Guest)
      * General validation
    - Guest FindGuestByEmail(String)
      * pass-through to repository
    - Guest findGuestByGuestID(int)
      * pass-through to repository
    - boolean checkValidStateShorthand(String)
      * General validation
  + HostService
    - Host FindHostByEmail(String)
      * pass-through to repository
    - Host findHostID(String)

• pass-through to repositor

* + - Result<Host> ValidateHost(Host)
      * General validation
    - boolean checkValidStateShorthand(String
      * General validation
  + ReservationService
    - List<Reservation> findAllReservationsForHost(Host)
      * pass-through to repository
    - Reservation findReservationById(List<Reservation>, int)
      * pass-through to repository
    - List<Reservation> filterUpcomingReservations(List<Reservation>)
      * pass-through to repository, filters results
    - List<Reservation> filterReservationsByGuest(List<Reservation>, Guest)
      * pass-through to repository, filters results
    - Result<Reservation> addReservationToFile(Reservation)
      * validate, then add via repository
    - Result<Reservation> updateReservation(Reservation)
      * validate, then update via repository
    - Result<Reservation> deleteReservation(Reservation)
      * validate, then delete via repository
    - Result<Reservation> checkReservation(Reservation)
    - Result<Reservation> conflictingDateRange(Reservation)
      * General validation
    - boolean areDatesOverlapping(Reservation, Reservation)
      * General validation
    - Result<Reservation> validate(Reservation)
      * General validation
  + Response
    - ArrayList<Strings> errorMessages
      * Holds error messages
    - AddError(String)
      * Adds error to error list
    - List<string> getErrors()
      * Returns list of errors
    - Boolean isSuccess()
      * Checks if any errors are in list
  + Results
    - Extend response
    - T payload
    - T getPayload()
    - setPayload(T)
* **UI**:
  + ConsoleIO
    - String readString(String) allow null
    - String RequireString(String) prevent null
    - String PromptForString(String Prompt)
    - Int readInt(String)
    - LocalTime readLocalTime(String)
    - BigDecimal readBigDecimal(String) //min max variant
  + Controller
    - Void Run()
      * Initiates program
    - Void RunLoop()
      * Primary menu loop
    - Void viewReservations ()
      * coordinates between service and view to print a selection of reservations
    - Void addReservation ()
      * coordinates between service and view to add a new reservation
    - Void editReservations ()
      * coordinates between service and view to update an existing reservation
    - Void deleteReservations ()
      * coordinates between service and view to delete an existing reservation
    - List<Reservation> identifyGuestsInReservations(List<Reservation>)
      * Helper, used to link guest data to selection of reservations
  + MainMenuOptions (Enum)
    - String Message
    - Int value
    - MenuOption getOptionFromValue(Int)
    - Int getValue()
    - String getMessage()
  + View
    - MenuOption selectOption()
      * Prompts user for a selection, used to control main menu loop
    - Void displayHeader(String)
      * Prints text divider in console
    - void displayStatus(boolean, String)
      * Prints pass / fail state of a result
    - void displayStatus(boolean, List<String>)
      * Prints a list of pass / fail states of a result
    - void displayException(Exception)
    - void printReservations(List<Reservation>)
    - String getHostEmail()
      * Prompts user for string input
    - String getGuestEmail()
      * Prompts user for string input
    - void printHostResults(Host)
      * Prints out a selection of reservations to the console
    - Reservation createNewReservation(Host, Guest)
      * Prompts the user for reservation dates
    - Result<Reservation> confirmReservation(Reservation)
      * Prints a summary of a reservation and asks the user to keep it
    - int getReservationID()
      * Prompts user for integer input
    - Reservation updateReservationDates(Reservation, Host)
      * Prompts the user for reservation dates

**Controller Perspectives**

Void Run()

Void RunLoop()

View Reservations

1. Collect Host Email
2. Validate host from repository via service
3. Display Host reservations via view

Add a Reservation

1. Collect Host Email
2. Collect Guest Email
3. Validate host from repository via service
4. Validate guest from repository via service
5. Display Host upcoming reservations via view
6. Collect stay duration of reservation from view
7. Print summary of reservation from view
8. Add reservation via service
9. Display success of fail from view

Update a Reservations

1. Collect Host Email
2. Collect Guest Email
3. Validate host from repository via service
4. Validate guest from repository via service
5. Display Host reservations from guest via view
6. Collect updated stay duration of reservation from view
7. Print summary of updated reservation from view
8. Update and save reservation via service
9. Display success of fail from view

Cancel a Reservation

1. Collect Host Email
2. Collect Guest Email
3. Validate host from repository via service
4. Validate guest from repository via service
5. Display Host upcoming reservations from guest via view
6. Collect Target reservation ID from user
7. Fetch target reservation by ID via service
8. Remove reservation via service
9. Display success of fail from view

**Schedule Outline**

Monday

* Reviewed provided project outline and instructions
* Prepared local file directory and new project
* Began creation of project outline
* Created new branch on Git

Tuesday

* Created directory file tree
* Added Class files for each layer
* Implemented Model classes
* Implemented data layer classes
* Added commonly used class files
  + ConsoleIO, Result, Response
* Added Simple outline UI layer methods
  + Print menu

Wednesday

* Added seeded data file for test purposes
* Adjusted Model layer classes
* Fixed bugs in data layer
* Added data layer test files
* Begin implementation of domain layer
  + Implementing bottom to top

Thursday

* Began implementation of UI
* Began to Implement top to bottom in-order to better trace the flow of methods

Friday

* Finish Implementation of Domain classes
* Finish Implementation of View classes

Saturday

* Added Spring Injection
* Remove Test code from classes in main
* Test Input
* Ensure method comments
* Clean up code block structures
* Clean up output console
* Started to write missing domain layer test files

Sunday

* Finish remaining domain layer test files
* Review and Update Project Outline documentation
* **Menu Option Breakdown**
  + View Reservations for Host (email)
    - Print by date
    - Error if no Reservations
  + Make reservation
    - Ask Host Email, Guest Email,
    - Select host
    - Select guest
    - Show future reservations for host
    - Get Start and End (no overlap)
    - Print Summary
    - Start, End, Cost
    - Confirm
  + Edit reservation
    - Ask Host Email, Guest Email,
    - Show host-guest reservations
    - EDIT START/END, confirm
  + Cancel reservations
    - Ask Host Email, Guest Email,
    - Show future host-guest reservations
    - Show success