Brian Lee
CS202 - Fant
Program 4/5 Design Write up

For program 4/5 the following classes will be implemented: City, Listing, Apt., Room, House, Reservation, Review and Application.

A Balanced tree (implemented as a AVL tree) will store all the available Cities that a user can select from. For each City, there will also be a AVL tree of all the available Listings for rent in that City. Each Listing will be of type Apartment, House, or Room and each Listing will have a linked listed of Reviews, organized by the most recent Review. Since the Listing class will serve as the abstract base class for the Apt., Room, and House classes, dynamic binding will occur between the abstract Listing class and the derived Apt., Room, and House classes (see the differences between abstract and derived classes below).

While all Listing have common items such as price, number of guests, number of rooms, a title, a description and a range of available dates; The specific types of Listings (Apartment, House, Room) will have some unique data such as any taxes or fees that apply, the minimum number of days a reservation must be made for (while some types may not have a minimum), experiences offered by the host, and accommodations.

Users will be able to read Reviews for each Listing found in a City (provided there are reviews). The Reviews will include the length of the reviewer's stay, a rating (1-10), and comments.

If the user decides they would like to stay in a location, that Listing will be added to the user's Reservations. The Reservations will be stored as an array of Listing for a given City that the user is staying in. From the Reservation class the user will be able to view their Listings in sequence from the beginning of their trip to the end. After each stay at a Listing, the user will be prompted to leave a Review.

The Application class will act as a container for a major classes (the Reservation and City/ Listing classes). From the Application class a user could host their place and look up places to stay in, view all cities, etc.

Prior to allowing the user to interact with the Application, data for the Review, Listing, and City classes can be read in from an external text file. In this case, that will be part of the implementation so that the program will have data ready to work with.

For classes like the City and Listing (which are both AVL trees) methods to remove, insert and find are necessary as well as methods to obtain the height of the tree and the balance factor for a given node (to assist with rotations). For the Listing class specifically a method to add a given Listing to a Reservation object's linked list of Listings would be handy. Additionally, for the Listing class, since there are many data fields (when instantiated as subclass objects Apt., Home, Room) methods to search within a price range, by number of guests allowed, and number of rooms would be good to have. Implementation would simply be a matter of searching the tree with the requested constraints. Ideally, there would be references to different arrangements of trees (for ex. organized by price, number of guests, rooms etc.).

For the Reservation class, methods to add and remove a Listing are essential. Additionally, before a user can add a Listing to their Reservations, the Reservation class will check to ensure that the user has not violated any rules for the type of Listing (ex. reserving the Listing for less than the minimum days, or indicating that the user plans to have more guests stay in a Listing than are allowed).

See UML digram below: