CSS143 Final Homework

Winter 2019

Original by John Chenault and Lesley Kalmin, revised by Lesley 2019

Purpose

The purpose of this assignment is to give you additional experience in inheritance, generics, sort, and more complex data structures – as well as larger projects.

Overview

You are to write a general-purpose reservation system. You will then extend this system to be able to reserve Tables in a Restaurant, and to reserve Boats with boat rental company.

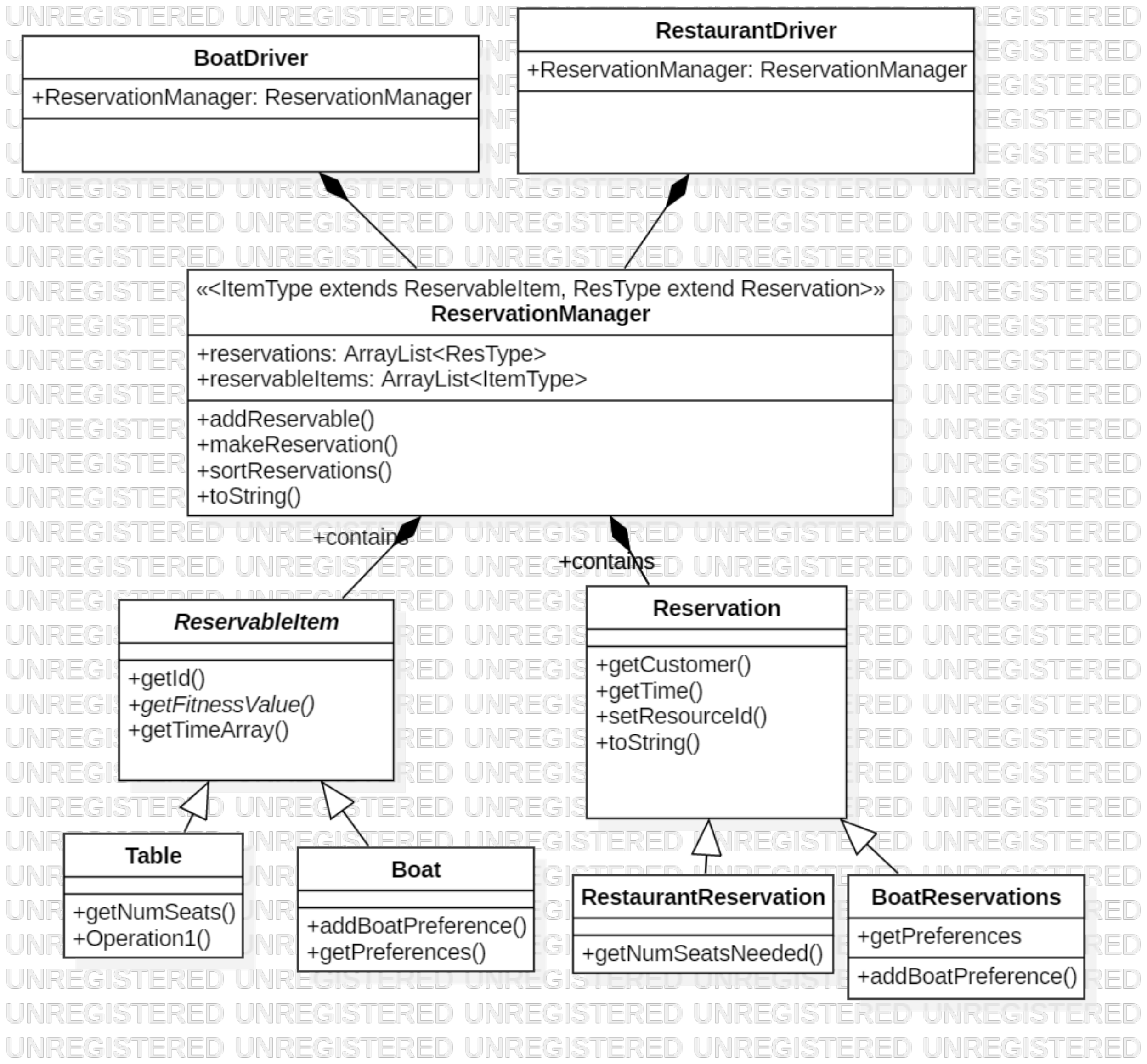
In all three cases (Base system, Restaurant system, and Boat Rental system) you will read in the resources to reserve from provided files: A list of tables and how many people they seat, and a list of boats. I will provide these files, names tables.txt and boats.txt.

Additionally, you will be given two basic Driver classes (one for each derived system) that will read in the data files, allow the user to make reservations, and to print out various information about the state of the system.

Reservation are made for a time slot, identified only by its position in the array of time slots. Each type of ReservableItem has different criteria for choosing the best fit. Similarly, the user enters different data as part of the reservation request.

For a table, the user indicates how many people are in the party, and the table must have at least that many seats. For the boat, the user gives a list of boats, ordered by preference, and the best fit is the first on the list that has the requested time slot open.

For example, if I ask for a table to seat 6, I will be given the table available with at least 6 six seats, but as few extra seats as possible. If I give a list of three boat types, in order of preference, I get the first type if it is available, if not I get the second type, and if that is not available, I try for the third.



The Base System:

The base system will contain three classes.

- **ReservableItem** – represents an item that can be reserved
- **Reservation** – represents a request to the system to reserve an item for a specific time slot if the reservation is successful it will know what ReservableItem it matched with
- **ResManager** – a manager that holds a list of Reservable Items, and a list of Reservations.

ReservableItem

This is an abstract class. The Reservabletem class represents an item that can be reserved. Each ReservableItem will have 10 time slots that can be reserved / requested by the user. The user will ask for them as 0 – 9, so you can use the time to index into a sequential structure. I recommend making 10 a constant in your class. It also has a private id, which will be either the table name or boat name.

- Required public methods
  - ReservableItem(Scanner fileIn) // Constructor
    - Reads the next line from the Scanner of an input file. The file has already been opened.
    - Extracts data for new item
    - This constructor is needed in each derived class type, not actually needed in this parent class
  - String getId(); // name of the item, such as table3 or kayak
  - abstract int getFitnessValue(Reservation res)
    - Returns an integer between 0 and 100 which tells the caller how well fit a reservation is for this ReservableItem. 0 means not at all fit. 100 means best fit.
    - Only implemented in subclasses.
  - boolean[] getTimeArray() // Returns the array of time slots
- Suggested implementation / hints
  - We advise you make an array of length 10 of booleans, default false. Each slot in the array represents a time. If it is false, that time is available. When a reservation is made against an item, set the corresponding array entry to true.

Reservation

The Reservation class represents a request by the user to reserve an item and, if successful, represents the reservation for the item. It implements Comparable, for sorting by customer name.

- Required public methods
  - Reservation (String customerName, int timeslot) // Create a reservation request for a name, and slot
  - String getCustomer()
  - int getTime()
  - String toString()
  - void setResourceId(String id) // Set the id of the reservableItem. Ids are read from the files, such as tableN or boat_type
- This class needs to be subclassed for RestaurantReservation and BoatReservation.
  - Each of those will have a constructor that takes the necessary data to make its type of reservation.

Reservation Manager

ResManager is the manager for Reservations. You will only need one implementation of this class, as the generics make it able to handle both Tables and Boats. The ResManager will hold ArrayLists (you can use the java built-in) of Reservations and ReservableItems. The ResManager will be implemented as a Generic class of type types (ItemType and ResType) where ItemType stands for ReservableItem, and ResType stands for Reservation.

- Required public methods
  - Type definition should be class

- `ResManager<ItemType extends ReservableItem, ResType extends Reservation>` where ItemType is a subclass of Reservable and ResType is a Reservation
   - ResManager()   // No argument constructor
   - void addReservable(ItemType item) // Adds an item to the manager
   - Reservation makeReservation(ResType trialRes) //Attempts to take a reservation based on the info in trialRes. If it fails returns null.
   - void sortReservations()   // sorts the list of reservations by customer name. Use your choice of bubble or insertion sort. Remember to use compareTo() for value comparisons.
   - String toString() // creates string listing reservations

```
// Steps to make a reservation:
//     for each ReservableItem
//                check if it is available in that time slot
//                get a fitness value, track the best one
//   if there is any ReservableItem with fitness > 0
//          take the best one
//          mark the item's time slot taken (array entry set to true)
//          put the reservation in the reservation list

//    else print failure message
```

The Restaurant system:

You will create two new classes. Table derived from ReservableItem, and RestaurantReservation derived from Reservation.

The Table class will have an additional integer variable for the number of seats on the table.

The RestaurantReservation will have an additional integer variable for the number of seats needed.

The Table Class is derived from ReservableItem and represents a table in the restaurant. The constructor adds to the ReservableItem class just the number of seats at the table.

- Required public methods
   - Constructor – Table (Scanner inFile)
      - Fills in String id, int numSeats
   - Int getNumSeats()
- Methods to override
   - findFitnessValue(Reservatin res) Finds the appropriate fitness value so tables will be assigned appropriately. By this we mean that you will look for an available table which has the fewest number of extra seats. This will be assigned the highest fitness value. Also note that when this method is called it will be with a RestRes, but the signature of the method is of the base class – IE a Reservation. You will need to cast appropriately.

The RestaurantReservation class is derived from Reservation and represents request for a table at a specific time slot.

- Required public methods
    - Constructor RestaurantReservation (String name, int startTime, int numSeatsNeeded);
    - Int getNumSeatsNeeded()

Input

The Driver class will load in the Tables from a provided data file. It will allow the user to make reservations against tables, and view various reports.

For example, for Tables the output would look like:

```
Unsuccessful reservation: Reservation name: TooBig time: 4 id:  customer
seats: 11
Unsuccessful reservation: Reservation name: party6 time: 4 id:  customer
seats: 10
Unsuccessful reservation: Reservation name: overflow time: 4 id:  customer
seats: 10
Reservation name: Chen family time: 3 id: table2 customer seats: 2
Reservation name: Singh party time: 4 id: table5 customer seats: 8
Reservation name: Kal time: 4 id: table6 customer seats: 8
Reservation name: business1 time: 4 id: table7 customer seats: 8
Reservation name: Newmans time: 2 id: table2 customer seats: 2
Reservation name: party5 time: 4 id: table8 customer seats: 10

Sorted reservations
Reservation name: Chen family time: 3 id: table2 customer seats: 2
Reservation name: Kal time: 4 id: table6 customer seats: 8
Reservation name: Newmans time: 2 id: table2 customer seats: 2
Reservation name: Singh party time: 4 id: table5 customer seats: 8
Reservation name: business1 time: 4 id: table7 customer seats: 8
Reservation name: party5 time: 4 id: table8 customer seats: 10
```

The Boat Rental system:

The Boat Rental system tries to let the customer pick a boat type for a ride. You will have two new classes – Boat (derived from ReservableItem) and BoatReservation (Derived from Reservation)

The ID of the ReservableItem is the type of the boat. Boats are also reserved in the 10 time slots.

The main difference from Restaurant is that the user can give both a time and a list (1 or more) of boats in order of preference. The fit algorithm takes the boat available with the best preference.

The Boat class is derived from ReservableItem. It does not need any new fields.

- Methods to override
    - getFitnessValue(Reservation res) Finds the appropriate fitness value for the boat. The request contains a list of possible boats, so you will need to find a way to compute a fitness function that ranks them appropriately.

The BoatReservation class is derived from Reservation.

- Public Methods
  - BoatReservation(int timeslot, String customerName) // constructor
  - void addBoatPreference(String boatName) // Adds a boat name to an internal list of preferences in the reservation
  - ArrayList<String> getPreferences(); // returns list of customer's boat preferences

The Driver class will load in the Boats from a provided data file. It will allow the user to make reservations for boats, and view various reports.

For example, for Boat Rental a session output look like:

```
Adding Boat type: kayak
Adding Boat type: rowboat
Adding Boat type: submarine
Adding Boat type: yacht
Adding Boat type: speedboat
Adding Boat type: aircraft carrier
Adding Boat type: pedalboat
Adding Boat type: canoe
Adding Boat type: zodiak
Adding Boat type: dinghy

Unsuccessful reservation: Reservation name: Unlucky party time:
3 id:

reservations before sort by customer
Reservation name: Chen family time: 2 id: kayak
Reservation name: Singh party time: 8 id: speedboat
Reservation name: Kal time: 8 id: dinghy
Reservation name: Party9 time: 8 id: zodiak
Reservation name: Newmans time: 2 id: speedboat
Reservation name: Party2 time: 3 id: speedboat
Reservation name: Party5 time: 9 id: speedboat
Reservation name: Party6 time: 1 id: aircraft carrier
Reservation name: Party3 time: 3 id: zodiak


reservations after sort by customer
Reservation name: Chen family time: 2 id: kayak
Reservation name: Kal time: 8 id: dinghy
Reservation name: Newmans time: 2 id: speedboat
Reservation name: Party2 time: 3 id: speedboat
Reservation name: Party3 time: 3 id: zodiak
Reservation name: Party5 time: 9 id: speedboat
Reservation name: Party6 time: 1 id: aircraft carrier
Reservation name: Party9 time: 8 id: zodiak
Reservation name: Singh party time: 8 id: speedboat
```