Assignment 1 CS20B (Questions Portion)

Jonathan Reznik

Student ID#1654919

**EDIT: I made some changes to problems 1 and 5. Questions 2, 3, 4 and 6 are unchanged.**

1. *Two examples of the object and class relationship are* a (a) chair object from the class for Furniture, and (b) a dog object from the class for Animal. *The differences are many between objects and classes including,* methods of objects are instance methods usually while static methods are applicable to the static members of the class. Objects unlike classes are generated at runtime, added to a heap in memory, and referenced or aliased, cloned or copied, etc. but a class is simply stored in a .java file providing the organizational view of the code. Naming conventions differ among them as well.
2. *Graph Structures*
   1. Flights
      1. Node: Destination
      2. Edge: Travel time or distance
   2. Borders
      1. Node: Single Country Name,
      2. Edge: Pairs of countries e.g. ‘US/Mexico border’, or ‘South US with North Mexico’.
   3. Research Collection
      1. Node: Title or Author
      2. Edge: Mention of certain words, common references or citations
   4. Actors
      1. Node: Actors names
      2. Edge: Movies in common with other actors
   5. Computer Network
      1. Node: Host
      2. Edge: Connections or Services (e.g. shared files, printing or transfer of data)
   6. Labyrinth
      1. Node: Intersections of paths
      2. Edge: Connections to other intersections, with distance or direction in weighted values
   7. WWW
      1. Node: Hosts
      2. Edge: Ports or Addresses or weighted values of traffic over either one
   8. Social Media
      1. Node: Individuals account
      2. Edge: Number of friends or shares (likes)
3. ArrayList inherits from AbstractList, which inherits from AbstractCollection. The ArrayList class has 3 direct subclasses: AttributeList, RoleList, and RoleUnresolvedList. (Not sure about implement methods, but the implementation Classes are List, RandomAccess, Cloneable, and Serializable). A count of all methods include 3 constructor methods, 31 instance/concrete methods, and another 16 methods inherited from the three inherited classes. And toString() is inherited from AbstractCollection in ArrayList.
4. Complexity in **Big-O**
   1. O(N) or linear complexity is derived from the N + 1 count of operations with 1 variable declaration and N linear processing operations inside a for loop.
   2. O(N2) or time-squared complexity is derived from N2 + 1 operations, with 1 variable declaration and N2 or N \* N from single nested operation inside of the two nested for loops.
   3. O(N) or linear complexity is derived from 2N + 1 operations, with 1 for variable declaration, and 2N or N + N in each subsequent for loop, which makes N + N + 1 or 2N + 1.
   4. O(N2) or time-squared complexity is the same as it was in part (b) despite counting the number in quarter overall as derived by N2/4 +1 count of operations the complexity O(N2).
   5. O(N) or linear time complexity is derived via N + 4 or N + 3 depending on if counting the return statement.
5. Design Problem
   1. Banking – Modeling customer activities with a parent class CustomerAction, and children classes DepositTo, WithdrawalFrom or BalanceInquiry. The benefit of such a design would be the ability to reuse the code for establishing customer related information, such as a name, date of birth, and account #’s.
   2. Gaming – Many models could exist for inheritance to be used in a simulated game setting. I will use the example of overlaying sounds into an existing virtual environment or game simulation, with parent class Sound extended to children subclass for various actions like SndNewGame, SndMakeMove, and SndQuitGame. In this case the reusability extends beyond variable members into shared method calls are inherited from superclass each time sounds play in the game, e.g. playSound() or findSoundFile().
   3. Travel – Modeling it from the *perspective of* the travel business offering various modes of transportation as options to its customers would be easily accomplished with a parent class for ModeOfTransport and children subclasses ByPlane, ByTrain, ByBus. In reality this inheritance hierarchy may work better as an interface with other classes for Fare, or TravelTime, and TravelDestination, which might also include subclasses DestinationLoc and DepartureLoc. In the first example (ModeOfTransport) the subclass for each type may provide some specifics such as the exact reason and amount of trip delays, while the superclass could then be used for more shared attributes like each object (or trip) has a data member associated with it called totalTravelTimeForTrip in the superclass.
6. Complexity – NOTE: I tried to make some explanations that involved more than identifying the highest order of polynomial. Please correct faulty logic wherever possible...
   1. N2+3N = O(N2), because evaluating each term with x2 gives 4x2 which is complexity O(N2).
   2. 3N2+N = O(N2) basically for exact same reason is 4x2 which is O(N2).
   3. N\*(N-1)/2 = O(N2) simplifying mathematically 0.5N2 – 0.5N, looking at the highest order to determine this is of complexity O(N2).