**Week 2 Grade: 89/100 (7.1/8 of the final grade)**

**Review the following:**

1. **Course Content -> Java Review and more ->SeparationOfConcerns**
2. **Course Content -> Java Review and more ->Managing functionality across classes**

|  |  |
| --- | --- |
| Points | Item |
| 10/10 | Class diagram meets the system specification and does not have anything extraneous (You should only have classes, attributes, inheritance, and functional methods in this assignment; **Do not** define yet constructors, get/set methods, exceptions, or other relationships. You will work on these in later weeks. Only define the methods for the functionality you will need in each class. |
| 5/5 | There is a Manager class that User Interface (UI) will interact with and which is storing and managing accounts/reservations (e.g. in Trip example there is similar class called TripOrganizer) |
| 10/10 | There is correct and complete set of classes with correct names and using Java capitalization convention. Classes represent objects (noun) |
| 9/10 | There is correct and complete set of attributes for each class with appropriate and meaningful names and using Java capitalization convention. Attributes represent characteristics or states and have “has a” relationship.  Manager   * account: incorrect attribute - per requirements manager needs to load multiple account objects and not just a single one * path: per Java naming convention constant name should be all uppercase   Account   * noOfAccount: invalid attribute - this class represents a single account object so it cannot store how many accounts there are * lodging: incorrect attribute name - since it is storing multiple objects it should be named something that is plural   Reservation   * missing attribute for number of bedrooms with default of 1 |
| 15/15 | Diagram includes all the necessary inheritance relationships of type “is a”. Attributes and methods reflect the inheritance being used and inheritance relationship is shown using the correct notations in the diagram |
| 20/25 | There is correct and complete set of functional methods for each class with appropriate and meaningful names and Java capitalization convention. Methods represent actions performed by the class (verb). Methods have the appropriate return data type and parameters.  Manager   * addLodging versus updateReservation - you keep swapping between reservation and lodging across and it is confusing - the system is about reservations so you should stick to reservation across and get rid of lodging references * updateReservation, complete\*, cancel\*, calculate: while these methods could search through all accounts to find the matching reservation, UI would know the account id when making these requests so it should pass it to the method * toString: this is the only class that does not need this method because there is no use dumping all accounts data into a single string * createAccount: invalid method - manager cannot create a new account because it does not have data for it and it should not have to know what exact data account needs to be created - review document #1 above * Missing method to save specific account’s data given account id to files in the hardcoded path * Missing method to add account object * createAccountDirectory: you only need account id to create the folder so why pass the whole object?   Account   * Missing methods complete and cancel reservation (review document #1 and #2 above) * missing method to retrieve Reservation object given reservation number * missing methods to update reservation * missing methods to calculate reservation price per night and total reservation   Reservation   * calculatePrice: you need separate methods for calculating price per night and calculating total price * Missing methods to complete and cancel reservation   Reservation children:   * calculatePrice: ambiguous name - which price is it calculating? Total or per night ? |
| 5/5 | Attributes show valid access modifiers (can be all public for now) and data types. I expect you to use upper case names for classes and lower case for attribute names AND your identifiers must be unique and meaningful. Java is case sensitive and so “trip” is different identifier from “Trip”. |
| 5/5 | Classes and methods show valid access modifiers (can be all public for now) and methods have correct return data types |
| 5/10 | There are Java files representing the classes with code for attributes and method signatures and comments briefly explaining attributes and methods.  Attributes and methods are missing brief descriptions |
| 5/5 | Diagram is exported and submitted as an image (in ArgoUML go to File | ExportGraphics and it converts workspace to image file PNG by default) |