**Week 5 Grade: 87/100 (7/8 of the final grade)**

|  |  |
| --- | --- |
| **Points** | **Item** |
| 20/20 | Feedback from previous assignment is incorporated correctly into the class diagram.  Reservation   * I told you incorrect in last feedback - you should only have Reservation(String filename) and not one with File line - (same for children)   Account   * Same as above, you should only have Account(String filename) and not Account(File line) and actually you should be passing directory since Account must load both account data and reservation data from different files |
| 10/10 | Class diagram meets the system specification and does not have anything extraneous (you should only be adding aggregation and/or composition with cardinality and user defined exception classes) |
| 10/10 | Diagram includes all necessary and correct composition or aggregation relationship of type “has a”. Shown aggregation relationship is one where child can exist independently of the parent such as for example: Course (parent) and Student (child). You can have a student without a course. Shown composition relationship is one where child cannot exit independently of the parent such as for example: Building (parent) and Room (child). You cannot have a room without a building |
| 2/10 | Every composition and aggregation relationship has correct cardinality for each class on **each side** of the relationship   * Aggregation Manager to Account: incorrect cardinality on Manager - first of all there is only a single Manager instance in this system but regardless, aggregation/composition is always a one-directional relationship where one class has x to y objects * Aggregation Account to Reservation: missing cardinality on both sides * Aggregation Account to Address: incorrect cardinality on Account (aggregation/composition is always a one-directional relationship where one class has x to y objects); incorrect cardinality on Address - Account can only store a single Address * Aggregation Reservation to Address: incorrect cardinality on Reservation (one-directional relationship); incorrect cardinality on Address - reservation must have 1 but no more than 2 Address objects |
| 10/10 | Relationships and cardinality are using the correct notations in the diagram |
| 5/5 | There are user defined exception classes added with correct names and correct access modifiers as indicated by the requirements document. They inherit from correct builtin parent exception class that it is extending for the functionality indicated in the specification doc |
| 10/10 | Added exception classes have correct attributes (to store data with useful information about the error) with correct access modifier and meaningful name |
| 10/10 | Added exception classes have correct constructors and toString method with appropriate parameters |
| 5/10 | Java files and pseudo code is updated with exceptions code and constructors/methods which need to throw these exceptions are updated to indicate that too  Manager   * addAccount(Account account): if duplicate exists should throw DuplicateObjectException * createAccountDirectory(Integer accountNo): should be private * addReservation(Integer accountNumber, Reservation reservation): if account object with matching id does not exists, should throw IllegalArgumentException (same comment for all methods that must find matching Account object except getAccount(Integer accountno)   Account   * addReservation(Reservation reservation): if duplicate exists should throw DuplicateObjectException * completeReservation(String reservationNo): if reservation object with matching reservation number does not exists, should throw IllegalArgumentException (same comment for all methods that must find matching Reservation object except getReservation(String reservationNo) * Account(String fileName): should be passing directory and not file since must load both account file and its reservations; if loading fails for any reason, must throw IllegalLoadException   Reservation   * Set methods - if status is not draft should fail with IllegalStateException (same for children classes) * Reservation(String filename): if filename does not exist or cannot load object, it should fail with IllegalLoadException (same for children classes) * cancelReservation(),completeReservation(): if cannot do the action it should throw IllegalOperationException |
| 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) |