Krishaun Bartlett

Ryan Reynolds

Part 1 creates an enrollment system using the linked list structures to store the student, faculty, and course information. An array list structure is implemented to interrelate the three linked lists. Three classes-student, faculty, and course- were created. Private classes StudentNode, CourseNode, and FacultyNodes were implemented to store the data fields of a student, a course, and a faculty member respectively. The nodes are singularly linked with the creation of the next node in the Node class. The Integer object was assigned to the object type Student and the StudentNode instead of a generic because the Identification numbers are a 7-digit integer values. Student, Faculty, and Course have respective iterators but they were not implemented in the enrollment methods. Each class has two print functions one that prints the information for all of the nodes in the list and another for an individual list. Furthermore, all three classes have a clear function that sets the list to empty by setting head and tail equal to null. Each have an update method that allows the user to change one data field in the node with the exception of the identification number. Each class also has a remove method that removes the node given a specific ID#. The enrollment class is the driver of the program. Enrollment creates objects of types Student, Course, and Faculty called StudentList, CourseList, and FacultyList. Additionally, Enrollment creates a static ArrayList that stores a one dimensional array that stores Student ID#, Faculty ID#, and Course ID#. EnrollmentList has three methods that operate directly on the list EnrollStudentCourse, DropStudentCourse, and AssignFacultyCourse. EnrollStudentCourse adds the ID#s into the array then stores the array in the list. Enrollstudent also calls the capacity method to make sure that the class is not over capacity before adding the class. DropStudentCourse searches each array in the Array List if it finds the corresponding student and course IDs it sets each value equal to zero in the array. The program will not write this node back to the file upon exiting the program. AssignFacultyCourse searches each array in EnrollmentList and changes the faculty ID stored in the array to the ID passed to the method. DropFacultyCourse asks if the user wants to assign a new faculty to the course. If the user does not want to assign a new faculty to the course the course and students enrolled in the course will be removed from the enrollment list. The main method is where the lists are populated and written back. SecurityCheck is called to ask the user for the system password. If the password is entered incorrectly too many times the program terminates. The program reads in Student.txt, Course.txt, Faculty.txt, and Enrollment.txt using a scanner. Each txt file’s lines are formatted to read in the file line by line and obtain the relevant information via splitting the string. The population of Course, Faculty, and Student lists call their respective LengthCheck methods prior to adding each node. The LengthCheck methods ensure that the data in the file are within the format specified by the project prompt. After Enrollment.txt is read in the administrator can type any manipulation commands they wish. After running these commands the program uses each class’s print function to write the manipulated files back to their respective txt files. The enrollment class contains 3 additional print classes. PrintStudentSchedule prints all the courses a student is taking. printClassesToTeach prints all the courses a professor is required to teach. printStudentsInCourse prints all the students in a course.