Assignment 1

The src file comprises of Assignment1.java which contains the main method. The other files contain the provided interfaces and classes which implement them. Various classes that are implemented are as follows:

1. **Course implemented from Entity_:**
   - Course name and course title are maintained as strings within a course object and are passed to the constructor when a new object is created. THere's also a linked list named stud which maintains all the students(Student_objects) taking that course.
   - The method addStud(Student s) adds a student object to the list of courses.
   - The method name() returns the name of the course which is maintained within the object.
   - The method title() returns the course title which is already being maintained within the object.
   - The method studentList() returns an iterator of the type Iterator<Student_> for the linked list stud.

2. **Hostel implemented from Entity_:**
   - Hostel name is maintained as a string in the variable name which is passed to the constructor while creating a new object. It also maintains a linked list of students(Student_ object) named stud which do live in that hostel.
   - The method name() returns the name of the hostel which is stored as a string.
   - The method addStud(Student_ s) adds a new student object to the linked list stud.
   - The method studentList() returns an iterator of the type Iterator<Student_> for the linked list stud.

3. **Dept implemented from Entity_:**
   - Dept name is maintained as a string in the variable name which is passed to the constructor while creating a new object. It also maintains a linked list of students(Student_ objects) named stud.
   - The method name() returns the name of the dept which is already maintained as a string.
   - The method addStud(Student_s) adds a new Student_ object s to the linked list stud.
   - The method studentList() returns an iterator of the type Iterator<Student_> for the linked list stud.

4. **Student implemented from Student_:**
   - The variables name, hostel, dept, and entry number are passed to the constructor as strings and are also maintained within the object as strings. The other variables maintained are completedCredits (int) , totalCredits (int), cumulative (int) and also a linked list of CourseGrade_ objects for the courses taken by the student named as courses .
   - The method name(), entryNo(), department() and hostel() return the corresponding strings which are already stored.

Param Khakhar
2018CS10362

- The method completedCredits() returns the string of the int variable completedCredits. The method getLL() returns the Linked List courses and the method cgpa() cgpa by dividing cumulative by totalCredits and rounding the float upto two places.
- The method addCourse(CourseGrade_ c) adds the CourseGrade_ object c to the Linked List courses . If the grade of the student for that course isn't 'I', then the totalCredits is incremented by 3. The cumulative is incremented by 3*gradepoint where gradepoint is integer value for the corresponding grade. Further if the grade of the student isn't 'E' or 'F', then the completedCredits are incremented by 3 as well.
- The method courseList returns an iterator of the type Iterator<CourseGrade_> for the linked list courses .

5. CourseGrade implemented from CourseGrade_:

- The variables course name, course num, and grade are strings which maintain the info of a particular student for that course. These are passed as parameters to the constructor of the class.
- The method coursetitle() returns the already stored string coursetitle.
- The method coursenum() returns the already stored string coursenum.
- The method grade(), forms a new GradeInfo_ object from the string grade and returns it.

6. GradeInfo implemented from GradeInfo_ :

- The variables store(String) and l(GradeInfo_.LetterGrade) are maintained. The grade in string is passed to the constructor and corresponding GradeInfo_.LetterGrade object is created.
- The method grade() overrides the signature in the GradeInfo_ interface and it returns the variable l(GradeInfo_.LetterGrade) which is already maintained.
- The method value() returns the integer which is the gradepoint of a particular grade. It calls the method GradeInfo_.gradepoint() for the GradeInfo_.LetterGrade object l and returns the result.

7. Position<T> implemented from Position_<T>:

- It is a generic class and maintains a variable val of type T and next of type Position<T> . A variable of type T is passed to the constructor and the val is set to corresponding value. The variable next is initialized as null.
- The method value() returns the variable val of type T .
- The method after() returns the variable next which is maintained in the class.
- The method setNext(Position<T> pos) sets the value of the object variable next as pos.

8. LList<T> implemented from LinkedList_<T>:

A. Class PosIterator implemented from Iterator<Position_<T>>:
   - The variables curr and last of type <Position<T>> are maintained within the object. A Position<T> object node is passed to the constructor and the value of curr is assigned to node and the value last is initialized to null .

Param Khakhar
2018CS10362

- The method hasNext() returns a boolean true if the attribute next of the Position<T> object curr isn't null and it would return false if it is null.
- The method next() returns returns the Position<T> object which is assigned to the next of the Position<T> object curr. It updates the value of the attribute curr of the PosIterator object.

B. Class ValIterator implemented from Iterator<T>

- A new PosIterator object is created in this class and an object node of type Position<T> is passed to the constructor. The PosIterator object is maintained by a variable.
- The method hasNext() calls the hasNext() method of the PosIterator object and it returns a boolean.
- The method next() also calls the next() of the PosIterator object, but it returns the value of the object stored in Position<T> object which is currently referred by the variable curr of the PosIterator object.

- The variables first and curr of type Position<T> and a variable count of type int are maintained within a linked list object. No arguments are passed to the constructor and the object variable curr is initialized to value null.
- The method add(T e) adds the object of type T by initially creating a new Position<T> object and accordingly the next of the field curr is updated. Along with this the field count is incremented by 1.
- The method positions() returns an iterator object of type Iterator<Position_<T>> which is done by creating a new PosIterator object .
- The method positions1() returns an iterator object of type Iterator<T> which is done by creating a new ValIterator object.
- The method count() returns the variable count which is the size of the linked list.

9.Public class Assignment1

- The main method maintains 4 linked lists(LList objects) allHostels, allStudents, allCourses,and allDept of types Hostel,Student, Course, and Dept respectively. These are passed as arguments to the private methods getData and answerQueries.
- The private method getData() takes two strings which are the files containing data(students.txt and courses.txt) and 4 linked lists s1(LList<Student>), c1(LList<Course>), h1(LList<Hostel>) and d1(LList<Dept>). These are stored within the method as allStudents(LList<Student>), allCourses(LList<Course>), allHostel(LList<Hostel>) and allDept(LList<Dept>).

A. Buffered reader is used to read inputs from the file which reads a line from a particular file and split function is used on the resultant string which returns an array containing the strings which are separated by the space. Since the first file contains information about the dept,

hostel, entrynumber and name for all the students, a new student object is created for each line read and then the corresponding object is added to the linked list allStudents. The linked lists allHostels and allDept are updated as well. If the hostel(Hostel) of the particular student(Student object) is already present in the linked list allHostels, then the student object is added to the linked list of students contained by the hostel object; otherwise, a new hostel object is created and the student object is added to the linked list of the students contained in the hostel object. The above procedure is again repeated while updating the linked list allDept as well.

B. The second file namely courses.txt contains the information(coursename, coursenum, grade) about the different courses taken by the student(identified by entry number). Buffered reader is used again which reads each line of the file as String. The split() method returns an array of strings which contains the different substrings of the original string which were originally separated by spaces. The linked list allStudents is updated by adding new CourseGrade object to the linked list courses of a particular student (student identified by entry number). The reference of that particular student is stored.The linked list allCourses is also updated. If the course(Course object) with the particular name is already present in the linked list allCourses, then the student object stored by the reference is added to the linked list stud of the course object. If the course isn't found in the linked list allCourses then a new Course object is created and the student object(stored) is then added to the linked list stud of the newly created course. Thus all the data is stored in the linked lists allStudents, allCourses, allDept, and allHostels.

- The method answerQueries() takes in the string which is the file containing queries, the linked lists allHostels(LList<Hostel>), allCourses(LList<Course>), allStudents(LList<Student>) and allDept(LList<Dept>). It maintains the above arguments as references h1(LList<Hostel>), c1(LList<Course>), s1(LList<Student>) and d1(LList<Dept>). It also maintains a linked list results(LList<String>). All the results of the queries are stored in linked list in different position objects. Buffered reader is again used to read lines from the file query.txt. The split() method is again used on the string and the different queries INFO, SHARE and COURSETITLE are treated separately.

  A. For the query COURSETITLE, the iterator itr1 iterates through each course contained in the linked list allCourses and the corresponding course's name is added to the linked list results if the course is present in the linked list otherwise the string "Course Unavailable" is added to the linked list results.

  B. For the query INFO, the iterator itr2 iterates over the linked list allStudents until the required student is found(checking entrynumber and name). A reference to the required

student(Student object) is stored. A string result is maintained which contains the information about the student. Student's entry number, name, hostel, department, and cgpa are added by calling the respective methods of the student object. Then an array of size equal to the size of the linked list courses of the student is created which contains different CourseGrade_ objects. The array is then sorted by bubblesort by lexicographically comparing the course title of different CourseGrade_ objects. The sorted ordering is then stored in the string result along with the respective grades of the courses. Finally the string result is added to the linked list results. If a non existent student's information would is seeked, then the output would be "Student Unavailable".

C. For the query SHARE, the entity is searched in the linked lists allCourses, allHostels, and allDept. A string result is maintained which contains the entry number of all the students sharing the given identity. Since each entity(hostel, dept, and course) contains a linked list of student objects, that linked list is then iterated to store the entry numbers of the student in the string result. The split() method returns an array of substrings which were separated by a space in the string result. The array is then sorted lexicographically(entry numbers) by using bubblesort and then the entrynumbers in sorted order are stored in a string which is subsequently added to the linked list result.

- An array of size equal to the size of linked list result is created and the all the strings are stored. The array is then iterated in reverse order and all the strings are printed to the console.

All the classes are getting compiled and without any errors and the code runs perfectly without any errors. The output of the code matches the required output.

References:

- The concept of two iterators referred from the textbook Data Structures and Algorithms in Java by Goodrich and Tamassia.
- Implementation of iterator and enum from www.geeksforgeeks.com.

Param Khakhar
2018CS10362