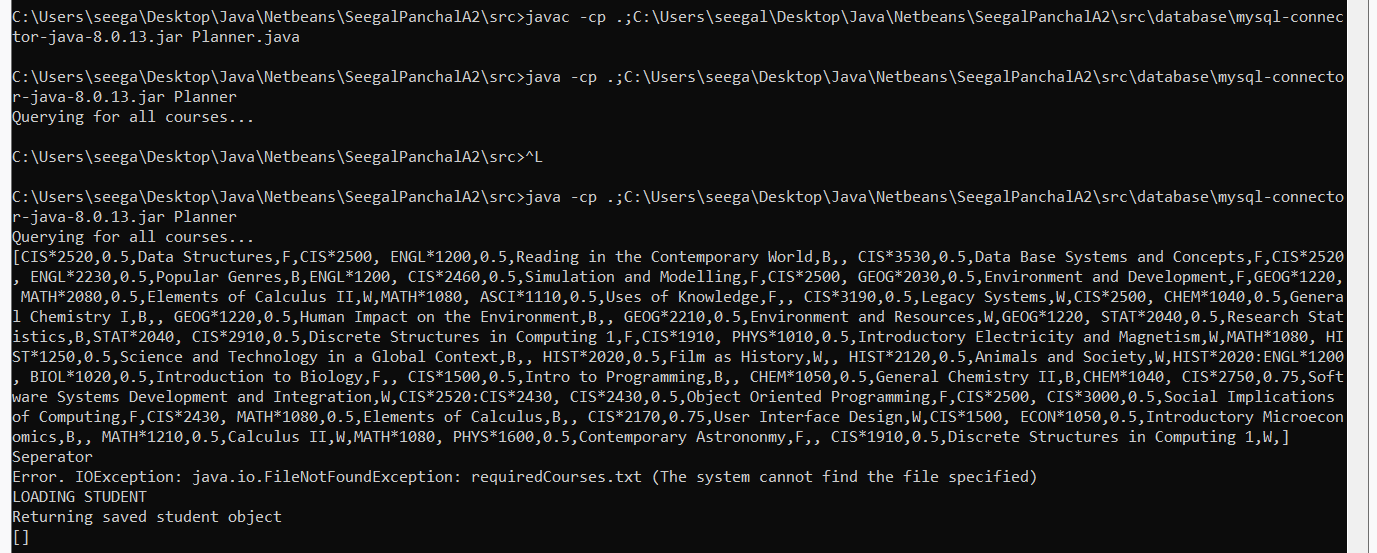
CIS 2430 A2 README/Checklist

|  |  |
| --- | --- |
| Name: | Seegal Panchal |
| UoG Email Address: | panchals@uoguelph.ca |
| Partner’s name (if applicable) |  |
| Percentage of your submission that is taken from the starter code provided (approximately) | I modelled a few getters/setters after the starter code, but I haven’t copied or used any starter code. I wrote all the code myself. If you want proof just look through the code and see that my code is better (hers doesn’t account for things properly. The only good code she has is getters/setters). |
| How to run your application: provide the exact string to type to compile it, and run it. Must be runnable on the command line. | **Either**:  javac Planner.java  java Planner  OR, if the sql driver doesn’t work, compile/run with absolute path. (With your own absolute path obviously)  OR, use the below relative path (this works all the time for me)  javac -cp .;.\database\mysql-connector-java-8.0.13.jar \*.java  java -cp .;.\database\mysql-connector-java-8.0.13.jar Planner |
| Notes for TAs (anything special we should know when grading your assignment) | I have an image of me executing the program from the terminal down below.  NOTE\*: I’ve run the same command twice, the first time it worked and if I run it with the same command again, sometimes it says “sql Driver ClassNotFoundException.” It works 100% of the time on Netbeans. I suggest trying multiple times.  How to reach each User Story in my program:   1. Degree drop down menu present on the screen when you log in, the selected degree is your current degree 2. You can ONLY select a major (you cannot have something like a General Degree without being a part of the BComp major for example) 3. There are buttons to add/remove courses, just input the Course Code / Semester Taken and click Update/Add to plan 4. Same as Plan, but you can also update grade by clicking update grade 5. Save button saves the entire student, including the Plan and Transcript 6. “Required Not in Plan/Transcript” button 7. “Required Not in Transcript” button 8. “Check Credits” button, last outputted line 9. I made this broader, you input a course and it outputs a list of its prerequisite courses if it can find them 10. “Prerequisites Required” button 11. “Check Credits” button, first outputted line 12. “Check Credits” button, second outputted line 13. “Check Transcript” button, checks if you meet requirements to graduate (from completed courses was my assumption) 14. “View Plan” and “View Transcript” button are sorted by the Semester Taken 15. “Check GPA” accounts for all 3 of the last user stories   How to reach each Admin Story in my program:   1. Click “File” on the menu bar, and select “Administrator Mode” and choose to Edit Courses/Edit Degrees 2. Choose “Edit Degrees”, this is a fake representation of what it would look like if there was a working way to do this 3. Choose “Edit Courses”, you can repopulate the courses to the database from the button “repopulate” and any changes you make you can save to the database |

|  |  |
| --- | --- |
| **Learning Outcomes** | **3 examples from your code. File name, line number** |
| refactor and restructure class design for improved encapsulation, modularity, cohesion and coupling | Student.java, Course.java, Attempt.java  I moved POS to Student, created Course/Attempt, made Student useful (login system with Student) |
| demonstrate use of inheritance through super/sub classes as well as through the use of interfaces | Degree.java   * I had inheritance in my degree class, moved my meetsRequirements and other functions down a level so I can accommodate a new class such as Bachelor of Arts (changed hierarchy from A1)   I used the Serializable interface to read/write required courses for Degree   * I also created a DegreeInterface that shows that each Degree needs methods to test the requirements for the degree |
| demonstrate clear understanding event driven programming through well designed listeners and gui components | Planner.java:  The menu bar Lambda for Admin -> (ln 79- 111)  Lambda for Save 🡪 ln : 123  Lambda for Exit -> ln: 131   * I also wrote LE3 and got 15 so I know how to use Listeners |
| demonstrate service-based error handling through a rich set of exception classes that communicate specific errors to client classes | Student.java  Line 233, setter method throws exception  Line 239, setter method throws exception  Planner.java:  IOException: line142 try/catch  NumberFormatE: Line 245 try/catch  NumberFormatE: Line 1369 try/catch  I had quite a few try/catches interspersed throughout my program |
| create a repeatable testing suite and justify the choice of test cases | Just work your way through the program testing edge cases. Each element can be tested immediately (adding/removing/changing anything, putting in negatives, etc..) |
| design and create a graphical user interface that is learnable and usable | Planner.java, literally the whole file |
| use inner classes, anonymous classes, and/or lambdas effectively | The menu bar Lambda for Admin -> (ln 79- 111)  Lamda for Save 🡪 ln : 123  Lamda for Exit -> ln: 131  I actually learned you can do lambdas like this:  (event) -> { more than 1 line of code here } |

|  |  |
| --- | --- |
| **Required elements** | **Examples from your code (File name, line number) – more than one example preferred** |
| Exceptions and try/catch loops | I looped a try catch right at the beginning where you if you try to log in you have to input a positive number between 1000 and 9999 [ LINE: 243 Planner.java ]  Student.java  Line 233, setter method throws exception  Line 239, setter method throws exception |
| Error prevention/handling (might also be try/catch or might be input checking) | IOException: line142 try/catch  NumberFormatE: Line 245 try/catch  NumberFormatE: Line 1369 try/catch  Error prevention is literally EVERYWHERE, GPA can’t be divided by 0, cant input duplicate planned courses, or duplicate transcripts, etc… |
| Two different layout managers | menuBar is CardLayout()  GUI is GroupLayout() |
| Separate window/panel for administration | Planner extends JFrame (so this)   * I made a mistake and should have made the frame inside the Univ package, but it works and I do not have time to change this   Admin Frame LINE: 294 Planner.java  Degree Frame LINE: 323 Planner.java |
| Listeners | The menu bar Lambda for Admin -> (ln 79- 111)  Lamda for Save 🡪 ln : 123  Lamda for Exit -> ln: 131  Lambdas are anonymous listener classes. I did not make any named Listeners and just used lambdas, as well as GUI-builded code (which has a bunch of listeners for Button events but I didn’t write them). |
| Course class refactored and immutable | Course class was moved to the univ package where it is immutable. The setters are not accessible from planner. Course only has relevant course information, such as:   * Course Code * Course Title * Semester Offered * Prerequisites |
| Attempt class created | Attempt.java, it contains information about a students attempt   * Attempt Grade (P,F,INC,MNR, or not set) * Semester Taken (also the semester planned to be taken) * Course Taken (The course itself) |
| Classes in package | univ package   * Course.java * DegreeInterface.java * Degree.java * BCG.java * CS.java * SEng.java   Database package:   * Database stuff   Other: (outside any package)   * Planner.java * Attempt.java * Student.java |
| Refactor Plan of Study (include how/where you provided the functionality if you eliminated POS) | I deleted POS and moved adding/removing courses from plan to Student at lines:  Add: 64(transcript)/82(plan)  Remove: 99(transcript)/105(plan) |
| Database usage | I connected to the database and it has a table of courses and students. You can save your data on the database. The administrator can repopulate the courses from the courseList as well.  CourseCatalog.java, Line 19, Line 50, Line 137, Line 167, etc…  Student.java Line 18, Line 64, Line 144, etc… |
| Javadoc comments (the most complete examples) | Student.java has the best Javadoc  Best ones:  Line 359, Student.java  Line 376, Student.java  Line 397, Student.java |
| Evidence of testing | I followed the way I showed above in ‘Notes for TAs’ to test each user case story.   * I would follow each user case story and test edge cases, for example, dividing by 0, adding duplicates, editing courses, etc… |

Other Notes:



Database connection is working, you just have to make sure the sql connector is in the class path.