**Research and Analysis:**

The code was constructed among two different classes, the Student Management System (SMS) and the Course Management System (CMS). The Student\_employee file was a subclass under the SMS. I declared the variable int management.Option; and used management = input.nextInt() after the introduction statement. Then I used the following method: if (managementOption == 1) menuLoop = true; …. } else if (managementOption == 2) menuLoop = true for the two main systems, SMS and CMS. This helps display only one interface at a time. The Student\_employee subclass had a method that was able to display all the students with an assigned on-campus job. An additional section included in the SMS, was assigning an on-campus job to a student after asking for the student’s ID. For assigning a specific student to an on-campus job, I decided to create a new array for Student\_employee. The Student\_employee [] obj2 = new Student\_employee [size]; creates a java class from the SMS. The int EmployeeArrayOrder = 0, add a new Student\_employee or return/remove them from the system. Throughout this case, I used System.out.println(); String … input.nextInt and System.out.println(); and String … input.nextLine; method therefore the user can enter the student ID, job, and job type. The obj.length was used to get the number of objects that are present in the array. To display all the students with an on-campus job, I used the same method as ‘case 3: displaying all students’, except I used obj2.length instead of counter +1. Additionally, I used the “if, else” method, therefore if a student is not assigned an on-campus job, then they will not appear when the user presses ‘6’. To exit the SMS, I used the menuLoop = false; break; and the system should return to the original introduction. For the CMS, I constructed a code that confirms a new code, assigns a student to a course, and can display all students with an assigned course. The Course [] catalog = new course [10]; creates a new java class from the CMS, where the user can insert up to ten different courses. The int courseCounter = 0, adds a new course ID to the CMS. I used the System.out.println(); String … input.nextInt to enter the course ID and System.out.println(); String … input.nextLine to enter the course name. The catalog[courseCounter] = newCourse; adds the course information into the catalog. To assign a student to a new course, I used the same method from SMS (when assigning a student to a new on-campus job); however, I used catalog.length instead of obj.length. For displaying all members with assigned courses, I used the same method under Student\_employee, however I changed the object values. To exit the CMS, I used the menuLoop=false; break; (which was also used for the SMS). At the end of the code, there is a} else if (managementOption == 0) {break;} else {System.out.println (“Option not recognized. Please pick 0, 1, or 2.” );, therefore if a user inserts the number that’s not ‘0,1,2’ then the system will remind them to pick 0,1, or 2.

**Design:**

My program is designed with two interfaces (SMS and CMS) under the Driver/main class. There are multiple objects and arrays for the two different classes. I added the new methods and important values that are going to be used throughout the program in the beginning. After identifying those key principles, I entered a comment of “STUDENT MANAGEMENT SYSTEM” next to the else if (managementOption == 1) to indicate that the first section is the SMS and entered the information for case 5 “Assigning on-campus job” in the SMS. In this case, I inserted the necessary codes that allows the user to assign a student to an on-campus job. I added a System.out.print(“has been assigned to” + JobType + “ “ + “job”); to give a written sentence of the student’s first name and last name with their specific job type and employment type. Furthermore, I provided a comment of “Displaying all students with on-campus job” for case 6. In this case, I included the code that would only display students with an on-campus job, instead of all the students in the system. I labeled case 0 as “Exiting the SMS” and used the menuLoop = false; break; method. Moving forward, I entered a comment of “COURSE MANAGEMENT SYSTEM” next to the else if (managementOption == 2) to indicate that the second section is the CMS. Once I finished writing the CMS description, I wrote a comment of “Adding a new course” for case 1 and collected the codes that would display the new course ID and course name for the CMS. The System.out.println(“Confirmation: New Course “+ newCourse.CourseID + “has been added.”); will give a sentence of the new course that’s been added to the CMS. For case 2, I had a comment that says, “Assigning student to a new course” and constructed a code that allows the user to assign a student to a course. I added a System.out.println(“has been assigned course” + courseSearchID); and this should give an output of the student’s first name, last name, and the new course ID. In case 3, I labeled this section as “Displaying all students with assigned course” and entered the code that would only display students with an assigned course, rather than all students in the system. For case 0, I commented “Exiting the CMS” and provided the same method of menuLoop = false; break;. Lastly, I inputted the} else if (managementOption == 0) {break;} else {System.out.println (“Option not recognized. Please pick 0, 1, or 2.”) to remind the user to only choose 0,1,2 when navigating the code.

**Testing:**

I tested the entire code by inputting all the values for both management systems to see if the computer generated the correct output. This helped me see if all implementations worked perfectly, and each system (SMS or CMS) only appeared on console when entering the values “0,1, or 2”. For example, if I enter ‘1’ then only SMS will appear, and when I enter “0” the computer will exit the SMS and take me back to the original introduction. Thus, when I enter ‘2’ then only CMS will appear. I would test each case individually before moving forward in the code once I had completed all the cases that needed to be created. This allowed me to understand if the new objects, arrays, and methods worked perfectly. If the console displayed an error, I would attempt to fix the mistake. As a result of this, I was able to comprehend the problems, and construct the code correctly.