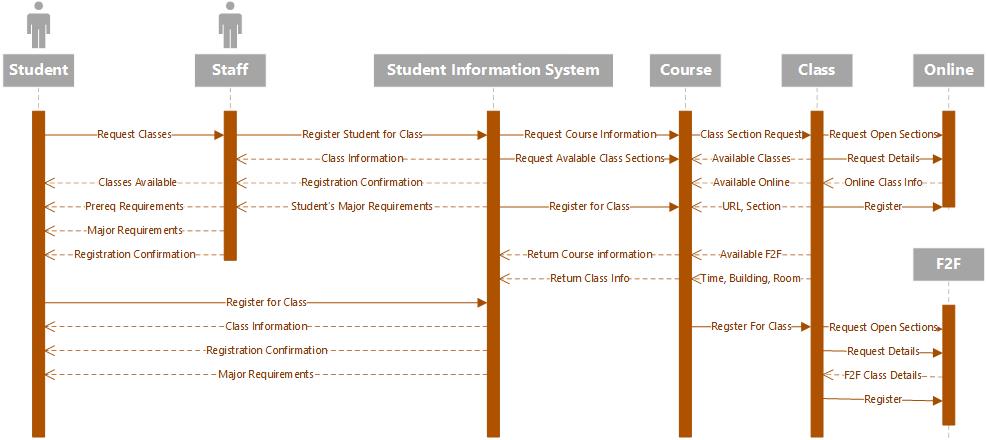
# IT 315 Final Project Part III Solution Submission Template

This template is a guide for you to organize your information. To complete it, **replace the bracketed text with the relevant information.** Some areas may be too large or too small for the information you’re inserting. Adjust the size of the areas as necessary.

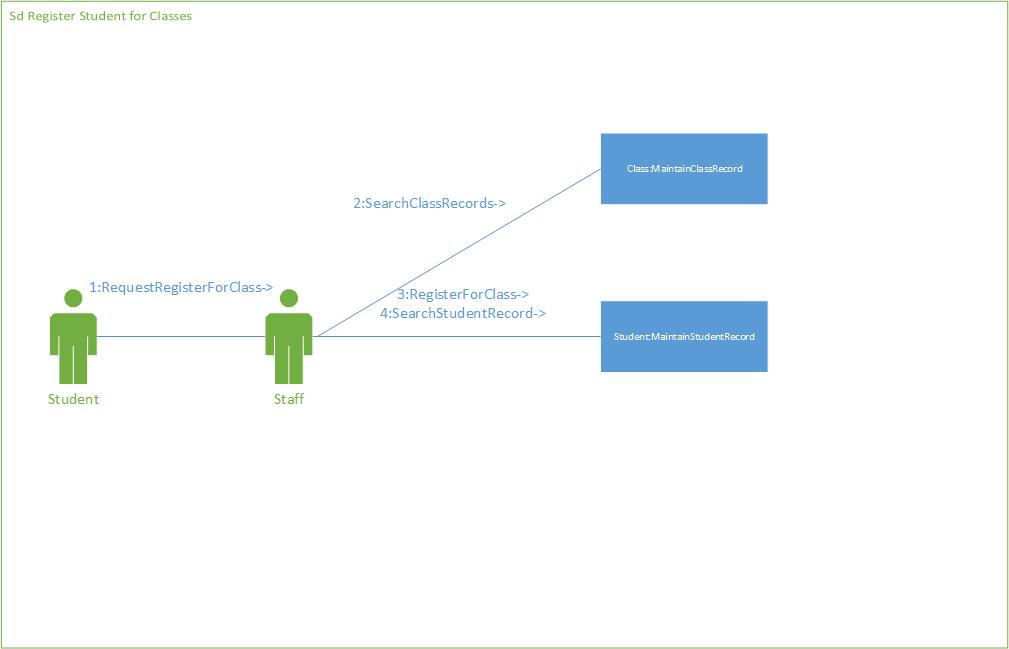
**Name:** Dylan W. Sanderson

**Date:** 2/28/2020

1. Generate your student information system (SIS) sequence diagram for the Register a Student for Classes use case.



Generate your SIS communication diagram for the Register a Student for Classes use case.



**SIS Method Contract 1 template** (refer to textbook pages 306–314):

|  |  |  |
| --- | --- | --- |
| Method Name:  Search Class Record | Class Name:  Maintain Class Records | ID:  1 |
| Clients (Consumers):  Staff  Student | | |
| Associated Use Cases:  Maintain Class Records | | |
| Description of Responsibilities:  Returns all classes available for a particular course. | | |
| Arguments Received:  Course Name | | |
| Type of Value Returned:  Class Objects | | |
| Pre-Conditions:  Course name is known  Course has classes available | | |
| Post-Conditions: | | |

**SIS Method Contract 2 template:**

|  |  |  |
| --- | --- | --- |
| Method Name:  Search Student Record | Class Name:  Maintain Student Record | ID:  2 |
| Clients (Consumers):  Staff  Student | | |
| Associated Use Cases:  Maintain Student Records | | |
| Description of Responsibilities:  Returns all information about a particular Student object | | |
| Arguments Received:  Student ID number | | |
| Type of Value Returned:  Student object | | |
| Pre-Conditions:  Student record exists | | |
| Post-Conditions: | | |

**SIS Method Specification 1 template** (refer to textbook pages 314–318):

|  |  |  |
| --- | --- | --- |
| Method Name:  Search Class Record | Class Name:  Maintain Class Record | ID:  1 |
| Contract ID:  Sample | Programmer:  Sample | Date Due:  Sample |
| Programming Language:  Java | | |
| Triggers/Events:  Someone accesses the SIS to request available class sections. | | |

| **Arguments Received:**  **Data Type:** | **Notes:** |
| --- | --- |
| Course Name | Course name the class would be associated with |

| **Messages Sent & Arguments Passed:**  **ClassName.MethodName:** | **Argument Data Type:** | **Notes:** |
| --- | --- | --- |
| Course.name | String |  |
|  |  |
|  |  |

| **Argument Returned:**  **Data Type:** | **Notes:** |
| --- | --- |
| Vector of Class Objects | All of the classes that match the given course |
| Algorithm Specification:  Sample | |
| Misc. Notes:  None | |

**SIS Method Specification 2 template:**

|  |  |  |
| --- | --- | --- |
| Method Name:  Search Student Record | Class Name:  Maintain Student Record | ID:  2 |
| Contract ID:  Sample | Programmer:  Sample | Date Due:  Sample |
| Programming Language:  Java | | |
| Triggers/Events:  Someone accesses the SIS and requests the information on a student. | | |

| **Arguments Received:**  **Data Type:** | **Notes:** |
| --- | --- |
| Student ID | The ID number for the student in question |

| **Messages Sent & Arguments Passed:**  **ClassName.MethodName:** | **Argument Data Type:** | **Notes:** |
| --- | --- | --- |
| Student.ID | int |  |
|  |  |
|  |  |

| **Argument Returned:**  **Data Type:** | **Notes:** |
| --- | --- |
| Student | The Student’s file |
| Algorithm Specification:  Sample | |
| Misc. Notes:  None | |

1. Verify and validate your sequence diagram and communication diagram against your SIS functional model and structural model.

These diagrams demonstrate what is needed for the action of registering a student. The sequence diagram depict the three players involved, namely, the student, staff, and SIS. They also show that the student may have the option to interact directly with the SIS, rather than having the staff member do it on their behalf. The communication diagram shows the specifics of what methods would be used, and the two methods outlined in the contract and method specifications show what input and output would be expected of the methods.

As explained, we added to the sequence diagram the subgroups within the SIS, namely, course, class, online, and F2F. These, along with their messages, now populate the diagram. They include obtaining specific information from the classes, from their subgroups, and open availability and registration, all functions formerly described in the CRC cards and previous diagrams. The messages sent in-between are supported by the functions described in the other diagramming efforts.

1. Explain your approach to the problem, the decisions you made to arrive at your solution, and how you completed it.

There were a number of unique problems to tackle here. The first was effectively demonstrating that the student had the option to interact with the SIS. This was demonstrated in the first diagram. We achieved this by extending the length of the bar below the SIS and shortening the length of the bar below the staff member to accommodate the message lines. Unfortunately, a number of the message lines were repeated, but there isn’t much we can do about that in this particular setup, as we needed to show that both the staff and student could interact similarly, and the staff could act on the student’s behalf given the proper messages from the student.

The communication diagram was a bit more taxing, but allowed us the chance to start to examine specific methods that would exist within the code. We also were able to outline the classes those methods would reside beneath, which led into the next part of the project. Lastly, we had to name two methods for the student records object, as we would need to both access the student records and register the student for a particular class.

Lastly, the cards were fascinating in that they allowed us to specify input and ouyput for the methods, which brought some unique questions to light. The first was how student records would be classified, and I determined that this would be through an int ID number, as names can be repeated, but each student would be assigned a unique number. However, we stuck with a string for course name input, as no two courses would be titled the same, and it would likely improve ease-of-use (especially if the search algorithm does not require an exact match). The student method would return the student object, likely in a digestible, readable form for viewing, while the class method would need to return multiple class objects based on the course chosen and availability. To this end, we defined that they would be returned in a vector, so they could be easily scrolled through and interacted with.

1. Reflect on this experience and the lessons you learned from it.

This particular experience was enjoyable for me, as I have spent the majority of my computer science education in dealing with OOP, and tend to think in such a way. Being able to outline input and output objects for methods seems very logical to me, and allowed me to put some previous experience to use. This also allowed me to figure out how to tackle some unique problems, such as, how would we handle it if the requested course has multiple class sections?

The diagramming gave me the chance to consider the methods and who would utilize them a bit more in-depth than we have insofar. Overall, this project got us closer to thinking of the actual coding side of the SIS, rather than just focusing on the abstracts of what we would like it to do. Here, we got to look a the practicality of actually creating a functional product.