# IT 315 Final Project Part II 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:** 2/16/2020

1. Using this class responsibility collaboration (CRC) card template, document each class you identified from your student information system (SIS) functional model.

**CRC Card 1:**

|  |  |  |  |
| --- | --- | --- | --- |
| Front: | | | |
| Class Name:  Student | ID:  1 | | Type:  Concrete, Domain |
| Description:  An individual attending the University who has/needs their records stored in the SIS. | | | Associated Use Cases:  4 |
| Responsibilities:  Ensure information up-to-date  Register for classes | | Collaborators:  Class | |
| Back: | | | |
| Attributes:  Name (text)  Date of Birth (text)  Student ID (int)  Major (text)  GPA (double)  Registered Classes (array) | | | |
| Relationships   * Generalization (a-kind-of):   Person   * Aggregation (has-parts):   Name  Date of Birth  Student ID  Major  GPA   * Other Associations:   Classes - Registration | | | |

**CRC Card 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| Front: | | | |
| Class Name:  Course | ID:  2 | | Type:  Concrete |
| Description:  Record of a course listing which will subsequently be tied to separate classes. | | | Associated Use Cases:  2 |
| Responsibilities:  Hold data about a course. | | Collaborators:  Class  Student  Staff | |
| Back: | | | |
| Attributes:  Name (text)  Course ID (int)  Credit Hours (double)  Description (text)  Prerequisites (text) | | | |
| Relationships   * Generalization (a-kind-of):   Record   * Aggregation (has-parts):   Name  Course ID  Credit Hours  Description  Prerequisites   * Other Associations:   Classes - Referenced By | | | |
|  | | | |

**CRC Card 3:**

|  |  |  |  |
| --- | --- | --- | --- |
| Front: | | | |
| Class Name:  Class | ID:  3 | | Type:  Concrete |
| Description:  A record of specific offerings of a course. | | | Associated Use Cases:  2 |
| Responsibilities:  Hold records of when and how a specific course offering is accessible | | Collaborators:  Course  Student | |
| Back: | | | |
| Attributes:  Course Reference ID (int)  Start/End dates (text or date/time)  online/in person status (text)  URL (text)  Browser(text)  Building (text)  Room (int) | | | |
| Relationships   * Generalization (a-kind-of):   Record   * Aggregation (has-parts):   Course Reference ID  Start/End dates  If online:  URL  Browser  If in person:  Building  Room   * Other Associations:   Course - References ID  Student - Referenced for Schedule | | | |

**CRC Card 4:**

|  |  |  |  |
| --- | --- | --- | --- |
| Front: | | | |
| Class Name:  Maintain Student Records | ID:  4 | | Type:  Functional |
| Description:  The class responsible for editing the Student object. | | | Associated Use Cases:  Maintain Student Records |
| Responsibilities:  CRUD Student objects | | Collaborators:  Student  Class | |
| Back: | | | |
| Attributes:  Create student object  Read information from student object  Alter (update) information in student object  Delete student object | | | |
| Relationships:  Generalization (a-kind-of):  Set of Functions  Aggregation (has-parts):  Crud functions  Other Associations:  Class - register for classes | | | |

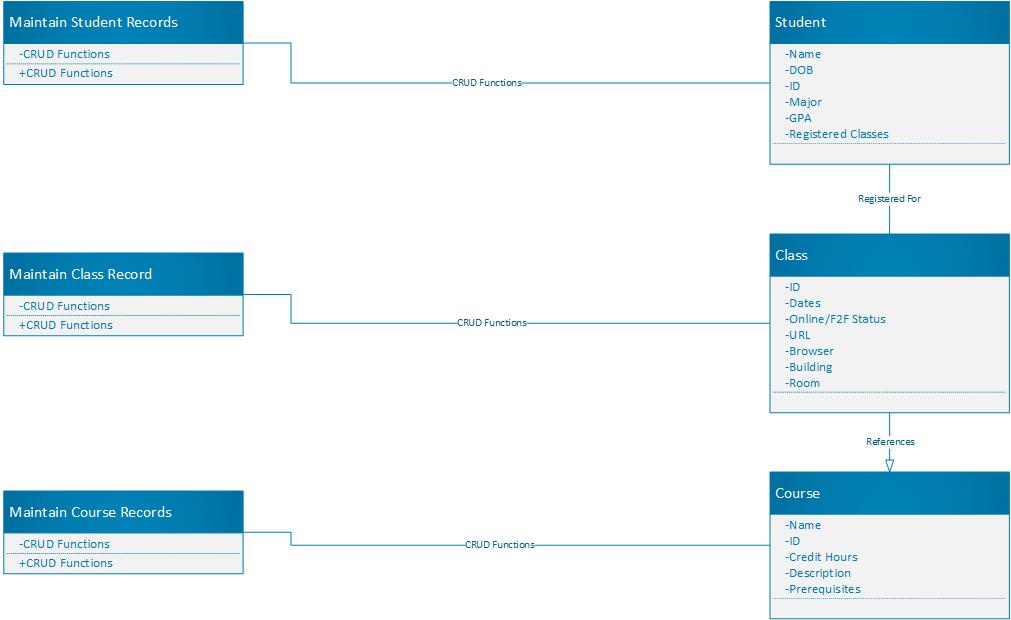
**CRC Card 5:**

|  |  |  |  |
| --- | --- | --- | --- |
| Front: | | | |
| Class Name:  Maintain Course Record | ID:  5 | | Type:  Functional |
| Description:  The class responsible for editing the course record objects. | | | Associated Use Cases:  Maintain Course Record |
| Responsibilities:  CRUD Course objects | | Collaborators:  Class | |
| Back: | | | |
| Attributes:  Create course object  Read information from course object  Alter (update) information in course object  Delete course object | | | |
| Relationships:  Generalization (a-kind-of):  Set of Functions  Aggregation (has-parts):  CRUD functions  Other Associations: | | | |

**CRC Card 6:**

|  |  |  |  |
| --- | --- | --- | --- |
| Front: | | | |
| Class Name:  Maintain Class Records | ID:  6 | | Type:  Functional |
| Description:  This class is responsible for any alterations to the class objects | | | Associated Use Cases:  Maintain Class Records |
| Responsibilities:  CRUD Class objects | | Collaborators:  Course  Student | |
| Back: | | | |
| Attributes:  Create class object  Read information from class object  Alter (update) information in class object  Delete class object | | | |
| Relationships:  Generalization (a-kind-of):  Set of Functions  Aggregation (has-parts):  CRUD functions  Other Associations:  Course - references ID | | | |

Generate your SIS class diagram:



1. Verify and validate your CRC cards and class diagram against your SIS functional model:

The functional model essentially detailed the need for six classes - three governing objects, and three governing the CRUD functionality therein. As such, while there might be some inter-referencing between the various class objects, only three pairs truly directly interact with each other, which is why the diagram looks as it does above. While I’m sure this is known, I should state that CRUD functions encompass Create, Read, Update, and Remove, the four basic modifiers on any object.

This made the diagramming a bit odd for the maintain records classes, but we will detail that more in the next section. Rather, now we need to focus on how these diagrams and cards fulfilled the SIS requirements. As there are three basic record types (student, course, and class), we would need a separate class for each, detailing the specifics that would be contained within the options. The CRC cards also detail whatever inter-referencing occurs, for instance, when a class references the governing course number, or the array of registered classes within a student record.

But this is only half of the equation. The other half is how those records are actually altered. In previous object-oriented programs I have created or interacted with, there tends to be a particular individual class for every object, and then another class encompassing the various functions that interact with the objects. In this case, we are dealing with the basic CRUD functions, which would likely be broken down into various sub-functions based on what particular data point was being edited.

We also added a couple of relationships between various objects with the class object. First, since the class object is a type of a course, referencing the course itself, we put a relationship there. Additionally, since the student object can be registered for classes, we added another relationship to class there.

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

The CRC cards were not a terrible burden, as we were approaching that part with previous experience. My largest struggle was with the overall class diagram, in which we had to show the interrelationships between the classes. In this case, despite having some references between classes, they were not vital for any other part to exist. Rather, the only concrete association was between the object classes and the associated function sets.

Additionally, we had to describe the attributes and operations within each class. For the object classes, they didn’t have any operations, as those were handled in separate classes. However, the attributes were concretely listed in the SIS document, and reflected in the CRC cards. As such, those were copied into the diagram. But this leads to the question, why are the operator functions described as such?

The following is my thought process: The attributes of a class are what it contains. For an object class, it contains the various facets of the object. For an alteration class, it contains whatever functions are used to alter it’s associated object. As such, in this case, they each contain the CRUD functions, detailed in greater detail on the CRC cards. I elected to list said functions as the operators as well, as that are the functions the class performs. Since they are how the alteration class affects the object class, they are also listed on the association relationship. This may seem redundant, but it is also accurate.

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

Perhaps the largest lesson here is how much I have yet to learn. I have learned that there are multiple ways to visualize, present, and discuss the various planning phases, none of which encompass every possible situation without working together. I am not terribly effective at class diagrams, as the planning and diagramming stages are not my strong suit, like pseudocode. I would much rather cut teeth into a program than plan it for days.

However, that does not mean these planning stages are valueless or ineffective, rather, my understanding of them may be. But this is why we are in class, to learn.

I learned that both CRC and Use Case cards are very different (despite the outwardly similar appearance), but also depend on each other for effective management and planning. Being able to effectively construct both of these items can increase the efficacy of any team. As such, we will continue moving forwards in the pursuit of knowledge.