MARMARA
ÜNİVERSİTESİ

Faculty of Engineering

Department of Computer Science & Engineering

Course: CSE3063(Object Oriented Software Design) **Lecturer**: Murat Can Ganiz

Course Registration Simulator

First Iteration:

Requirement Analysis Document

Members:

Haydar Taha Tunç 150119745

Emir Ege Eren 150119739

Mehmet Emre Şengüler 150119781

Burak Dursun 150119743

Mert Özincegedik 150119643

Emir Devran 150119749

Kerem Kolay 150119773

Osama Mustafa 150119885

Aimen Daddi 150119879

Burak Aslan 150117002

I. Vision

The Project's aim is to simulate the registration process at universities. This System will allow students to register their classes, also allow Advisors to oversee and manage the course registration procedure. It helps the university to update and manage classes and lectures.

II. Scope

In this project we use object oriented software design to create a simulation of class registration, by using this it will make the process of designing and implementation of this project more efficient but with some challenges as object interactions. Throughout this project we will try to respect all rules and obligations of student registration as course limit, prerequisites and credits…

III. Problem Description

Basically, what we need is a software that can simulate a course registration program for students so they can have an easier time while choosing their next classes for that semester. While we do that, we basically need Students, Advisors, Courses lists and the current semester which is either Spring or Fall. All the students will be created randomly also they will be given a year randomly so a student might start from third year, and we will look at that student and generate a random Completed Courses list for that student. We will also make a random FailedCourses generator which will have a percentage chance of failing a randomly generated Completed Courses object. We will then calculate the GPA with Completed Courses list for each student and generate a transcript for each student. We will consider that randomly generated students will be as close to real life as possible. After everything is randomly created or calculated. We will give Available Courses lists for every student so they will have a list of courses they can take for that semester. When the user runs the program it will ask it to choose a student and then it will display its Available Courses.

IV. Requirements

1. Functional Requirements:

1. We need to create random student objects and take into account of their semester and we need to generate a random Completed Courses and FailedCourses objects and

calculate their GPA and also, we need it to give us a transcript from these information.

2. The program needs to give a list of Available Courses for every student so they can choose their desired courses available for selection.
3. We will have advisors which will either allow or deny their courses sent for registration depending on constraints.
4. After randomly generated completed classes, the program needs to check failed classes also because it might randomly generate a course which a student passed but they didn't pass the prerequisite of that.
5. We will have json files for Students, Courses, Advisors which are randomly generated but the rest of them will be calculated like GPA or Available Courses.

2. Non-Functional Requirements :

1. Input files will be in json format.
2. Program can run on any computer capable of running jvm
3. Program will run on command prompt and not a GUI
4. Security issues will be handled by java's own encryption we will not consider much about security issues while writing the code.

V. Use Cases

Actors: System, Student

1. System creates Classes, Students and Advisors.
2. The System assigns Semester and completed Courses also calculates total Credits and GPA to each Student.
3. The System generates available courses for each student.
4. Student selects from available courses.
5. Students sends selected courses for advisor approval.
6. Advisor approves or disapprove courses
7. System creates transcript for each student

   3a. system generates Available courses according to semester and completed courses

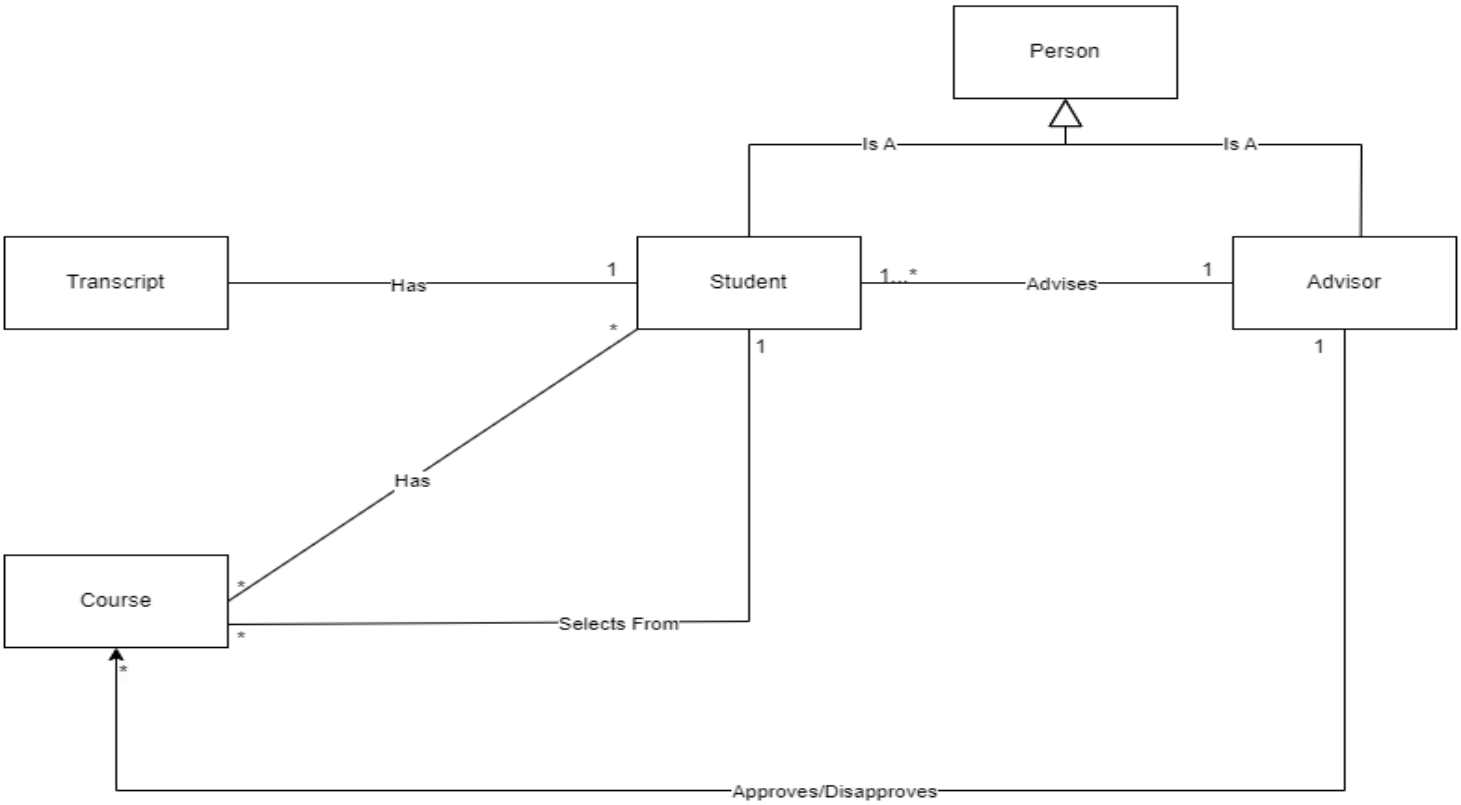
   4a. Student cannot select course that he/she has not passed its prerequisite.

   4b. The selection process prioritizes the selection of Failed Courses.

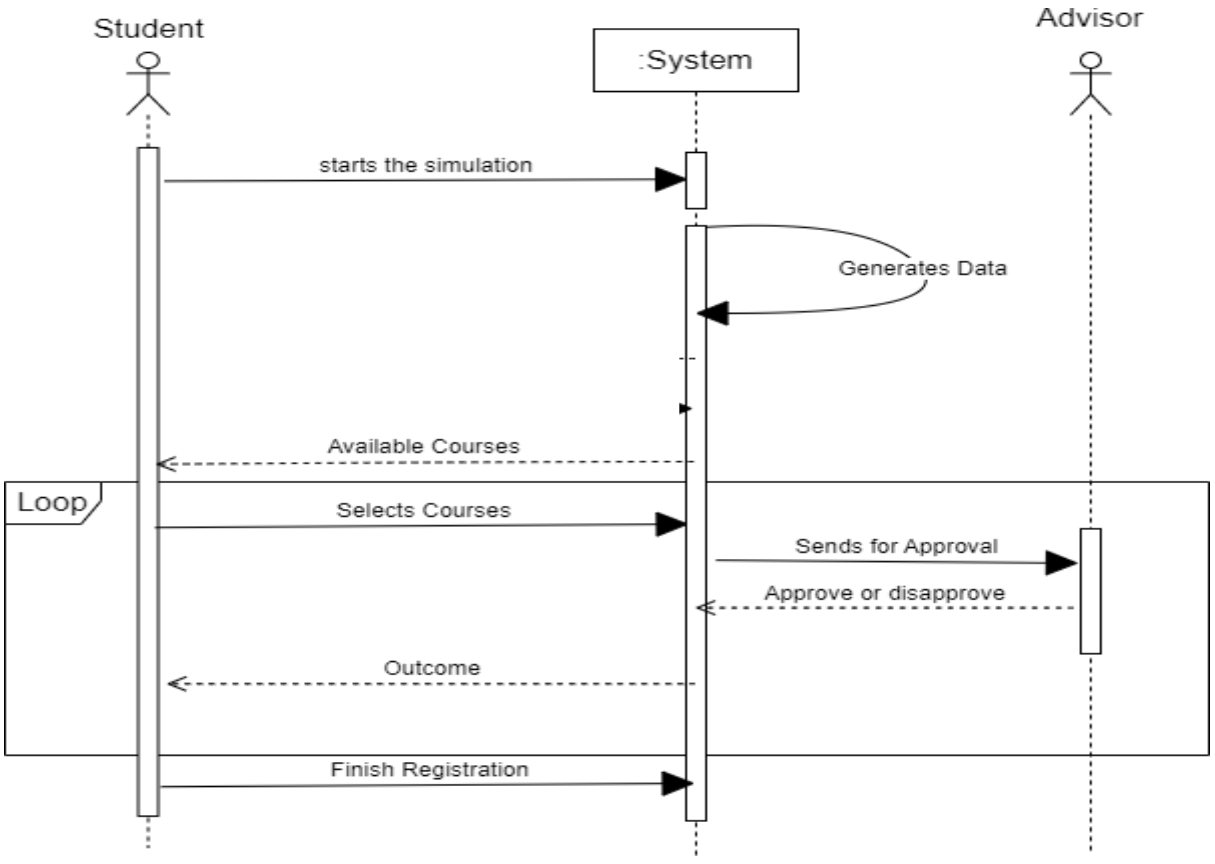
VI. Main Success Flow

1. Read from pre prepared Advisors.json file and create Advisors objects
2. Read from pre prepared Courses.json file and create Course objects
3. Random first names, last names and student id's and semester are generated and put into students.json file
4. Randomly generated completedCourses and also randomly generated grades need to be put into students.json file while taking into account of every students semester and the prerequisite.(a student with only 3 semester cannot have all the courses passed for example)
5. Randomly generated failedCourses need to be created for every class there might be %5 chance for a student to get FF from that class and put all these steps into students.json file
6. Read from students.json file and create student objects list
7. Calculate GPA of every student from failed and completed courses and their grades.
8. Student will have an availableCourses object which will be a list of available courses that is generated from completed/failed courses so that the student may register.
9. Student will select courses desired from availableCourses and that will be put in to selected courses list
10. Selected courses list will be sent to the students advisor
11. Advisor will approve/reject any course(s)
12. Old transcript gets deleted
13. New transcript gets generated.

VII. Domain Model



VIII. System Sequence Diagram(SSD)



IX. Glossary

- **Software**: A program
- **JVM**: Java Virtual Machine
- **Json**: JavaScript Object Notation, a lightweigth format for storing and transferring data.
- **GPA**:Grade Point Average, every student has a gpa which is calculated from their average passed and failed courses if they failed with FF
- **CompletedCourses**: A subclass object for the Student class so we can store the course name and course grade for every student
- **FailedCourses**: A subclass object for the Student class so we can store the course name and course grade for every student
- **AvailableCourses**: A subclass object for the Student class so we can store the course name to show it to students for their semester course registrations.