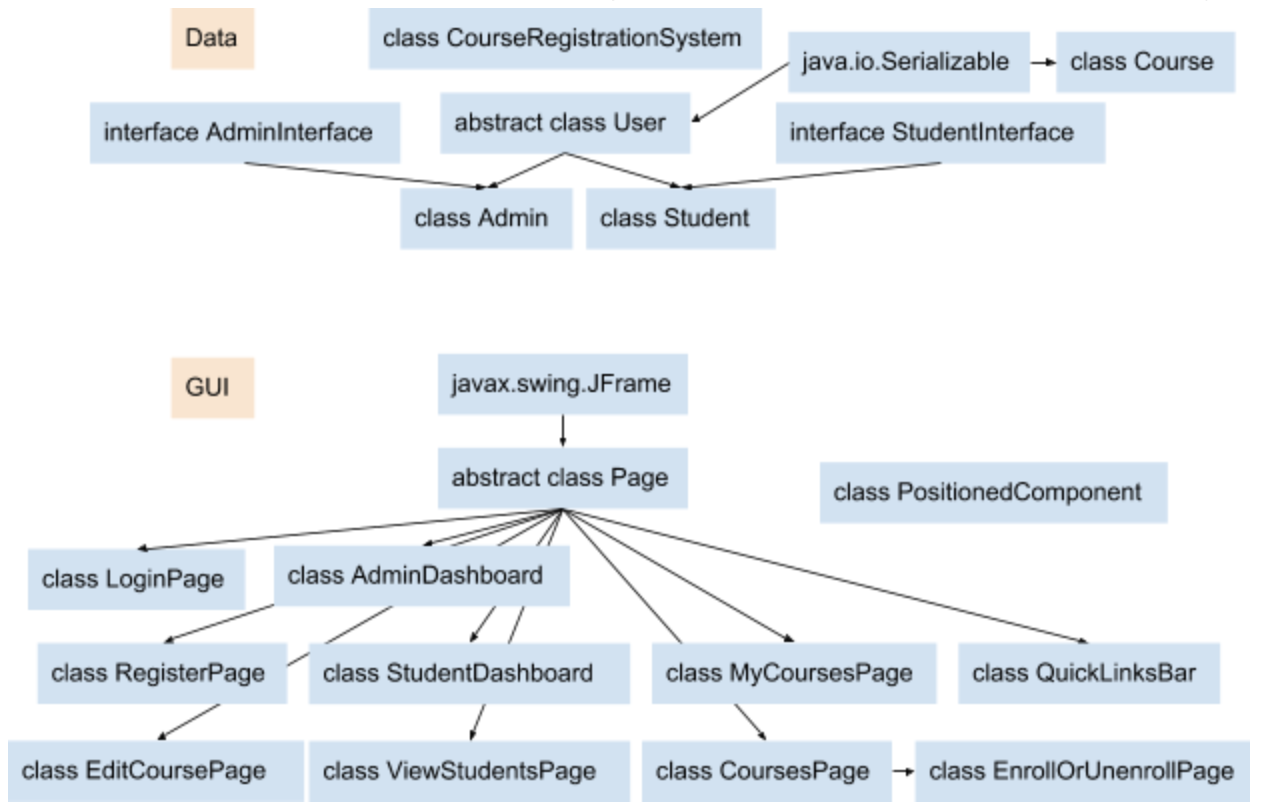
**Author:** Garrett Lu

**Description:**

This is a simple course registration system created with the Eclipse IDE using the Java language, while following the object oriented programming paradigm. The graphical user interface was created using Swing.

Relationships between classes and interfaces (an arrow indicates an inheritance relationship):

**How to run:**

Import project folder into Eclipse, double click *edu.nyu.cs.gbl254.course\_registration\_system.data\_management.CourseRegistrationSystem*, and click the green arrow.

**Program flow:**

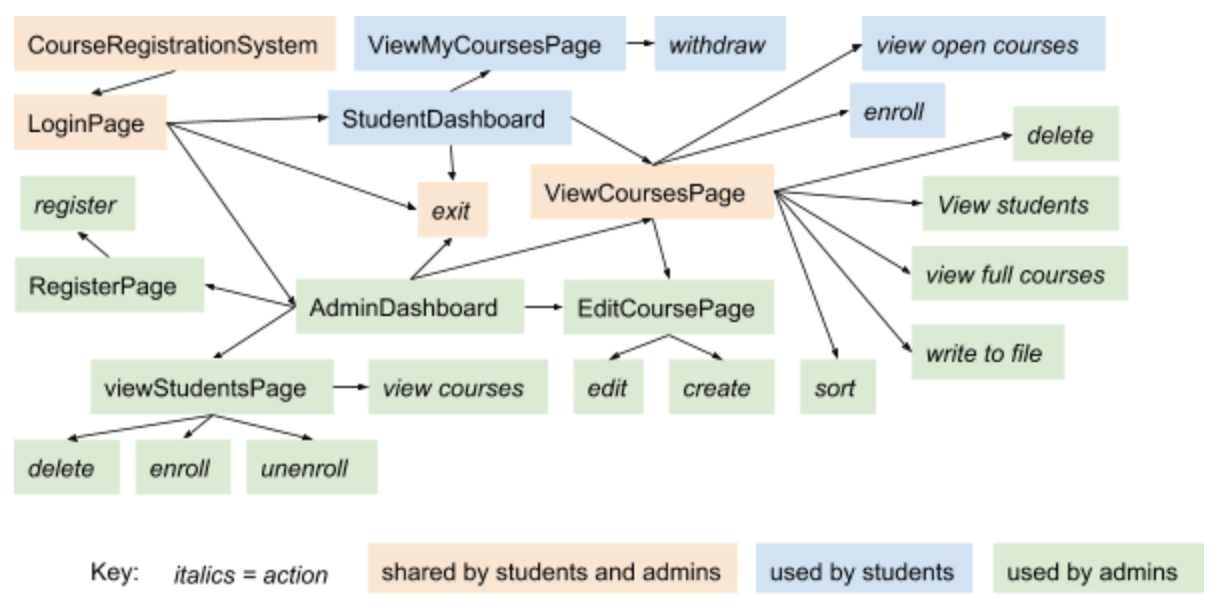
1. The main method is located within the CourseRegistrationSystem class. The program will first try to read a serialized file storing the ArrayLists of courses and students. If unsuccessful, it will read from a CSV containing course data.
2. After reading the data, a LoginPage is instantiated, allowing the user to log in.
3. After logging in, a StudentDashboard or AdminDashboard is instantiated, depending on user account type. Both dashboards have an “exit” button, which causes the system to terminate.

**If the user is a student:**

1. If the current user is a student and it is his/her first time logging in, he/she will be prompted to set a new username and password. After that is done, the student dashboard will show.
2. From the dashboard, he/she can click the “browse courses” button to view all courses that he/she is not enrolled in. From there, the student can filter courses by showing only courses that are not full, enroll in a course, and return to the dashboard by clicking the back button.
3. From the dashboard, the student can view all courses he/she is enrolled in, by clicking the “my courses” button. From there, the student can withdraw from a course, or go back to the dashboard.

**If the user is an admin:**

1. If the current user is an admin, the admin dashboard will show after logging in.
2. From the dashboard, he/she can click the “view courses” button to view all courses in the system. From there, the admin can filter courses by showing only courses that are full, sort the displayed courses by enrollment number, edit a course, delete a course, view a course’s enrolled students (where he/she can also remove students from the course), and write the displayed courses to a file.
3. From the dashboard, the admin can create a course by clicking the “create a course” button. He/she will be asked to enter the necessary course information.
4. From the dashboard, the admin can click the “view students” button to view student information. From there, the admin can view a student’s courses, enroll a student in a course, un-enroll a student from a course, and delete a student.
5. From the dashboard, the admin can click the “register a student” button to register a student with a first name, last name, username, and password.

Flow diagram:

**OOP concepts demonstrated:**

**Method overloading:** The User class has a method that allows users to see all courses in the system. The Admin class has a method of the same name that takes different parameters, which allows the admin to see all courses from a specific subset of courses (such as all courses in a student’s list of courses).

**Method overriding:** The User class has a method defined that allows the user to view all courses, but its behavior needs to be different for the Student class. A method of the same name and parameters is defined in the Student class, which overrides the one in the User class, the parent class.

**Abstract class:** The program contains two abstract classes, User and Page. User is abstract because the only types of users the system allows are Admin and Student. Therefore, User should never be instantiated as the actual type of an object; It contains properties and methods common to Admin and Student, which inherit from it.

**Inheritance:** The Admin and Student classes inherit from the User class, allowing them to share common attributes, such as first name, last name, username, and password. Multiple GUI pages inherit from the Page abstract class, which itself inherits from the javax.swing.JFrame class.

**Polymorphism:** The currentUser field in the CourseRegistrationSystem class is declared as type User, but can be an instance of Admin or Student depending on who is logged into the system.

**Encapsulation:** The various classes in the program wrap its properties and methods in their respective definitions. For example, a user’s first name, last name, username, and password is always stored in an instance of a User, and a course’s name, ID, and other information is always stored in an instance of a Course.

**Abstract Data Types:** List is an abstract data type that has multiple implementations, such as ArrayList and LinkedList. This program uses multiple ArrayLists to store course and user data.