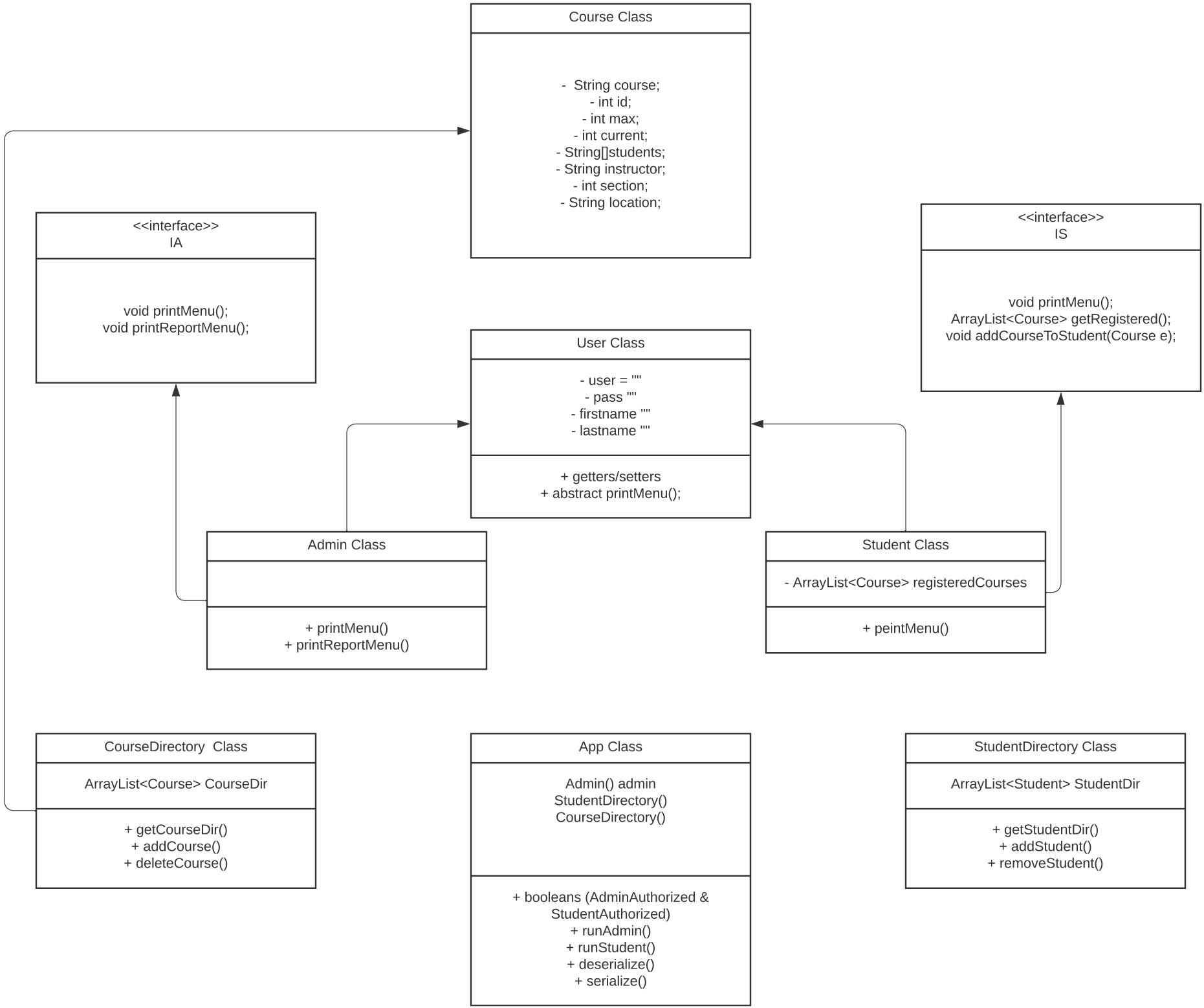
Course Class
- String course;
- int id;
- int max;
- int current;
- String[]students;
- String instructor;
- int section;
- String location;
<<interface>>
IA
void printMenu();
void printReportMenu();
<<interface>>
IS
void printMenu();
ArrayList<Course> getRegistered();
void addCourseToStudent(Course e);
User Class
- user = ""
- pass ""
- firstname ""
- lastname ""
+ getters/setters
+ abstract printMenu();
Admin Class
+ printMenu()
+ printReportMenu()
Student Class
- ArrayList<Course> registeredCourses
+ peintMenu()
CourseDirectory Class
ArrayList<Course> CourseDir
+ getCourseDir()
+ addCourse()
+ deleteCourse()
App Class
Admin() admin
StudentDirectory()
CourseDirectory()
+ booleans (AdminAuthorized &
StudentAuthorized)
+ runAdmin()
+ runStudent()
+ deserialize()
+ serialize()
StudentDirectory Class
ArrayList<Student> StudentDir
+ getStudentDir()
+ addStudent()
+ removeStudent()

Fatima Villena
CS102
Homework 1

Method Overloading:

The find() method in both Directory classes is seen twice, once with an empty parameter and once with an int id parameter. This allows me to accommodate the different searches I will need during the running of my program.

Method Overriding:

There is an overriding of the abstract method “printMenu()” in the User class by the Student and Admin class. There is also an overriding of all interface methods in IA and IS in Admin and Student, respectively such as the IS interface’s method “void addCourseToStudent(Course e);” which is provided a body in the Student class.

Abstract Class:

I have one abstract class “User” which is the parent class of both Student and Admin. They both inherit the constructors from User, but the User class is abstract because it is not instantiated in the program. The abstract method is printMenu() as both student and admin utilize it, but print out different menus.

Inheritance:

The only instances of the class User are made through inheritance by the Admin and Student class. Although multiple class inheritance in Java is not supported, Java allows multiple inheritance using interfaces, hence why “class Student extends User implements Serializable, IS”.

Polymorphism:

The ArrayLists made in my program are great examples of polymorphism. For instance, studentList is an ArrayList of Student objects, but it can hold instances of any class that is a subclass of Student or implements the Student interface. Polymorphism allows me to access the Student objects in this array uniformly.

Encapsulation:

There are multiple private data members throughout my program such as the private parameters for the constructors in User. These are only accessible in the class and through publicly accessible methods such as getId();. In this way, encapsulation maintains the integrity of the program and stores data away.

ADT:

ArrayList is an ADT due to its abstracted methods such as size(), add(), remove() that are defined inside the data type. The StudentDirectory and CourseDirectory can be considered to be Abstract Data Types due to their abstracted away methods and operations that act on their instances as objects in the main program.