Criterion C - Development

I've used several techniques that I won't be expanding on due to word count constraints:
● Data Hiding ● Files to save, store -- .txt ● Traversals: if, while, for loops ● Composition ● Encapsulation ● Validation and Verification ● Java Built-in classes ● Sorting algorithm -- in GUI/Java Swing (shown in video)

Serialization and Deserialization

Success Criteria: ● System's login page should store/save login details ● Information can be stored on persistent storage
Source: Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat "Serialization and Deserialization in Java with Example." GeeksforGeeks, 13 Jan. 2016 "Serialization in Java - Javatpoint." Www.Javatpoint.Com

Justification: Serialization allows users to reopen system without having to re-enter details, like login details. Data is saved & retained, further increasing usability & convenience. It allows an easy method to store aggregated objects in case of reusage.

Explanation: Serialization is built-in Java process for persistent storage. It converts entire object(and all its instance variables, even if instance variables are objects) and converts into a byte-stream and saves it to.txt file. It can then be retrieved and pre-entered data can be used without setting up another account.

Code Example:

```
public class AcademicAdvisor extends Person implements Serializable{


    private ArrayList<Student> studentList = new ArrayList<>();
    private ArrayList<Message> messages = new ArrayList<>();
    //variable to ensure serialization doesnt break
    private static final long serialVersionUID = 1093916303206104521;
```

Encryption Algorithm (SHA-256)

Success Criteria: ● System will feature secure login page
Source: "SHA-256 Hash in Java." GeeksforGeeks, Geeksforgeeks, 7 Aug. 2019

Justification: Encryption using hash function (SHA-256) will provide secure storage for login details in .txt file. No passwords can be extracted and decoded, therefore secure for important information like universities.

Explanation: Cryptographic hash is kind of 'signature' for text or data file. SHA-256 generates an almost-unique 256-bit (32-byte) signature for text. Creating this signature allows for absolute security as trying to retrieve password is computationally impractical. When password is stored, it goes through this process and then is stored in .txt file. If this file is intercepted, the hacker/cracker cannot understand. Adapted from geeksforgeeks.com, SHA hash allows no information to be 'stolen' or 'decrypted', which is something the client wants and needs when storing important information like university applications and documents.

To calculate cryptographic hashing value in Java, MessageDigest Class is used, under the package java.security. These algorithms are initialized in static method called getInstance(). After selecting the algorithm it calculate the digest value and return the results in byte array.

BigInteger class is used, which converts the resultant byte array into its sign-magnitude representation. This representation is converted into hex format to get the MessageDigest.[1]

Code Example:

[1] "SHA-256 Hash in Java." GeeksforGeeks

```java
//SHA-256 methods from https://www.geeksforgeeks.org/sha-256-hash-in-java/
public static byte[] getSHA(String input) throws NoSuchAlgorithmException
{
    // Static getInstance method is called with hashing SHA
    MessageDigest md = MessageDigest.getInstance("SHA-256");

    // digest() method called
    // to calculate message digest of an input
    // and return array of byte
    return md.digest(input.getBytes(StandardCharsets.UTF_8));
}

public static String toHexString(byte[] hash)
{
    // Convert byte array into signum representation
    BigInteger number = new BigInteger(1, hash);

    // Convert message digest into hex value
    StringBuilder hexString = new StringBuilder(number.toString(16));

    // Pad with leading zeros
    while (hexString.length() < 32)
    {
        hexString.insert(0, '0');
    }

    return hexString.toString();
}

//my method to simplify the whole process to convert string to SHA:
public static String stringToSHA(String input) throws NoSuchAlgorithmException{
    return toHexString(getSHA(input));
}
```

(Linear) Searching Algorithm

Success Criteria: ● System should allow AAs to search students by index and/or by name
Source: "Program to Convert List of String to List of Integer in Java." GeeksforGeeks, 30 Sept. 2018 Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat

Justification: A search was used to allow AAs to find/search students through their index or name to 'view' their profile. This adds a useful feature in the management system that allows for convenient searching for a student, his statistics and applications. Instead of scrolling and manually finding students in the list, it is much easier for a search algorithm.

Explanation: The parseInt method was used in the search algorithm if the user decides to search by index, or an integer. The inputted index value is compared to the pre-inputted information about the student, profile, universities, etc, in the arraylists. If the index is not within the limits, an error message is displayed. If index is in the arraylist and is searched, it is displayed to the user.
If the inputted value isn't an index, the user inputted value of first and/or last name is compared to ones already in the list. And if the list is empty, an error message is shown.
By showing and reviewing this functionality with the client, he was pleased due to its convenience that a student can be searched not only by index but also names.[2]

[2] Wailes, Gareth, 2021

Code Example:

```
                case('V'):
                    boolean found = false;
                    if (currentAA.getStuList().isEmpty()) {
                        System.out.println("List is empty, nothing to view.");
                        break;

                    } else {
                        String viewStudent = IBIO.input("Enter name of student, or index, to view their profile: ");

                        if (isInteger(viewStudent) && Integer.parseInt(viewStudent) >= 0 && Integer.parseInt(viewStudent) < currentAA.
getStuList().size()) {
                            int viewIndex = Integer.parseInt(viewStudent);
                            currentAA.getStuList().get(viewIndex).viewProfile();
                            System.out.println("");
                            currentAA.getStuList().get(viewIndex).listUnis();

                        } else {
                            for (int i = 0; i < currentAA.getStuList().size(); i++) {
                                if (currentAA.getStuList().get(i).getFName().contains(viewStudent) || currentAA.getStuList().get(i).getLName().
contains(viewStudent)) {
                                    found = true;
                                    currentAA.getStuList().get(i).viewProfile();
                                    System.out.println("");
                                    currentAA.getStuList().get(i).listUnis();
                                    break;
                                }
                            }

                            if (found = false) {
                                System.out.println("Cannot find student.");
                            }
                        }

                    }
```

Graphical User Interface (GUI) -- Java Swing

Success Criteria: ● System should have GUI(s) to support easier navigation, readability and functionality ● System should accomplish main objectives of: ○ Displaying information about universities and processes
Source: "Java Swing Tutorial - Javatpoint." www.javatpoint.com Yang, Herong. *Java Swing Tutorials - Herong's Tutorial Examples*. Vol. 4.30, Dr. Herong Yang, 2018.

Justification: GUI, thought time consuming and complex, yields favourable, visually appealing, efficient and easy to navigation interface. For large amounts of information (faq GUI) or specially formatted information (uniRank),GUI gives an aesthetically pleasing look to complicated and crammed information on CLI.

Explanation: GUI was coded using Java Swing. Hard-coding it allowed me to modify anything to developer's (my) liking. NetBeans does not allow all code to be editable when using their inbuilt GUI builder. This GUI isseparate class which I call in other methods. When prompted from menu-driven interface, GUI option (switch case) will openJFrame window with stated function; like "Displays 'FAQ' menu" or "Displays University Rankings". In the code below, I've used JTable to display universities and their attributes like name, location, SAT 25%ile, etc.
In faqGUI, I've used several Panels, which are displayed (setVisible(true)) when inputted to specific Strings. It shows inputted panels and hides other ones at same time. Back to main allows me to hide all, and display main/home page. (Not included in screenshot since lot of lines for simple show and hide panels)
Java Swing (& AWT) imports used:

- Borders

- Frames
- Buttons
- Labels
- Panels
- TextAreas
- TextFields
- Tables
- TableHeaders
- ActionListener -- ActionPerformed

Code Example:

```
// Column Names  -- overall
        String[] column1 = { "U Name", "Country", "City", "Overall", "Maths", "Econs", "Medicine", "IB min", "IB max", "SAT 25%ile", "SAT 75%
ile", "ACT 25%ile", "ACT75%ile"};
// Data to be displayed in JTable
        String[][] data1 = {
            { " MIT", " US", " Cambridge, Massachusetts", "   1", "   1", "   2", "   12", "  37", "  40", "  1450", "  1580", "  33", "  36"},

            { " Stanford", " US", " Stanford, California", "   2", "   3", "   3", "   4", "  38", "  41", "  1460", "  1570", "  32", "  36"},
            { " Harvard", " US", " Cambridge, Massachusetts", "   3", "   2", "   1", "   1", "  40", "  44", "  1470", "  1580", "  32", "
35"},
            { " Oxford", " UK", " Oxford, England", "   5", "   5", "   9", "   2", "  38", "  40", "  1485", "  1550", "  32", "  35"},
            { " Cambridge", " UK", " Cambridge, England", "   7", "   4", "   10", "   3", "  39", "  42", "  1495", "  1575", "  33", "  35"},
            { " UCL", " UK", " London, England", "   10", "   49", "   17", "   8", "  35", "  39", "  1430", "  1525", "  30", "  34"},
            { " Yale", " US", " New Haven, Conneticut", "   17", "   21", "   8", "   9", "  41", "  44", "  1495", "  1595", "  34", "  36"}
        };
        //table properties -- contents, row height, header, etc etc
        rank1 = new JTable(data1, column1);
        header = rank1.getTableHeader();
        rank1.add(header, BorderLayout.NORTH);
        rank1.setModel(modeleColonnesNoEdit(data1, column1));
        rank1.setRowHeight(30);

        //sets column width
        rank1.getColumnModel().getColumn(0).setPreferredWidth(70);
        rank1.getColumnModel().getColumn(1).setPreferredWidth(55);
        rank1.getColumnModel().getColumn(2).setPreferredWidth(150);
        rank1.getColumnModel().getColumn(3).setPreferredWidth(50);
        rank1.getColumnModel().getColumn(4).setPreferredWidth(50);
        rank1.getColumnModel().getColumn(5).setPreferredWidth(50);
        rank1.getColumnModel().getColumn(6).setPreferredWidth(55);
        rank1.getColumnModel().getColumn(7).setPreferredWidth(70);
        rank1.getColumnModel().getColumn(8).setPreferredWidth(70);
        rank1.getColumnModel().getColumn(9).setPreferredWidth(70);
        rank1.getColumnModel().getColumn(10).setPreferredWidth(70);
        rank1.getColumnModel().getColumn(11).setPreferredWidth(70);
        rank1.getColumnModel().getColumn(12).setPreferredWidth(70);
```

Data Structures

Success Criteria: ● System should have Review' page where each component can be viewed ● System should store values inputted by users
Source: Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat

Justification: Since Array lists are more flexible, easier to use and edit than arrays, I have chosen to use them over arrays. Array lists allow easier editing, removing, searching and sorting, etc.

Explanation: Array lists are used to hold AA's students and Student's university choices. Since it is easier to edit and more feasible, it allows clients to change content without any complications.

Code Example:

```
public class Student extends Person implements Serializable {

    private AcademicAdvisor AA; //Aggregation
    private ArrayList<University> universityList = new ArrayList<>();
    private ArrayList<String> applyCountry = new ArrayList<>();
    private ArrayList<Message> messages = new ArrayList<>();
    private ArrayList<Subject> subjects = new ArrayList<>();
    private ArrayList<Grades> subjectGrade = new ArrayList<>();
    private ArrayList<Grades> effortGrade = new ArrayList<>();
```

Menu/Choice Interface

Success Criteria: ● System features login page ● Student can edit profile; AA can create accounts -- navigation is through menu-driven interface ● FAQ bot displays choices to select for information about that choice -- input choice
Source: Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science: For the IB Diploma Program (International Baccalaureate). Express Publishing, 2018. Drien, Marcos, 2021

Justification: Users can navigate programs efficiently to add, edit or remove data, instead of inputting commands in Command Line Interface.

Explanation: This is possible by extensive use of Switch case, which allows programs to respond differently for each user input. It allows for effective and flexible use as per user's choice. below code shows switch case within switch case that is inside another switch case.

Code Example:

```
System.out.println("\nControls");
System.out.println("A - Add student");
System.out.println("R - Remove student");
System.out.println("V - View student's profile / Edit profile");
System.out.println("F - Displays 'FAQ' Menu"); //GUI here for easier readability, functionality & access
System.out.println("U - Displays university rankings"); //GUI here for easier readability & access
System.out.println("M - Open 'Messages' Menu");
System.out.println("L - Log out/Exit");
option = IBIO.inputChar("Enter option --> ");
option = Character.toUpperCase(option);
switch (option)
{
    case ('A'):
        addStudent();
        break;

    case ('U'):
        {
        try {
            new uniRank();
        } catch (Exception ex) {
            Logger.getLogger(AAMethods.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
        break;
```

(switch case continues...)

Validation Exception/Error Handling

Success Criteria: ● System should have data verification and validation and prohibit input of invalid data (for appropriate fields).
Source: Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science: For the IB Diploma Program (International Baccalaureate). Express Publishing, 2018 Drien, Marcos, 2021 “Exception Handling in Java | Java Exceptions - Javatpoint.” Www.Javatpoint.Com “Java Exception Handling: How to Specify and Handle Exceptions.” Stackify, 17 July 2017

Justification: Validation Error Handling prohibits system to run unusually, and not like it should. It prevents any breaks in system, and erroneous values are dealt with, without hindering function. Try-Catch blocks find errors and process it differently.

Explanation: program stops when an EOF (End of File) Exception is caught. This happens if AA objects are empty, when trying to load them. Exception Handling identifies this error, and doesn't hinder process of system, when usually it would stop running.

Code Example:

```java
public static boolean isInteger(String s) {
    try {
        Integer.parseInt(s);
    } catch(NumberFormatException | NullPointerException e) {
        return false;
    }
    // only got here if we didn't return false
    return true;
}
```

Inheritance

Success Criteria: ● System should allow students to edit, add or remove subjects -- 6 subjects + EE + TOK -- and profile -- intended path of study, countries, etc.
Source: Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science: For the IB Diploma Program (International Baccalaureate). Express Publishing, 2018 Drien, Marcos, 2021

Justification: AA and Student inherit from Person class which consists of firstName, lastName, id, username and password.

Explanation: Implementing inheritance promotes code reusability and readability. When child class inherits properties and functionality of parent class, we need not to write same code again in child class. This makes it easier to reuse code, makes us write less code and code becomes much more readable.[3]

AA and students have several similar variables as Person, thus they inherit these, reducing code duplication

Code Example:

```
public class ExtendedEssay extends Subject implements Serializable{
public class AcademicAdvisor extends Person implements Serializable{
public class Student extends Person implements Serializable {
```

[3] Singh, Chaitanya. "Inheritance Advantages." *Beginnersbook.com*, 11 Sept. 2017

Aggregation

Success Criteria: ● AAs can add predicted grades which can only be viewed, and not edited, by students ● Students can add universities which can only be viewed by AA ● Students can edit their own information like test scores, path of study, etc
Source: Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science: For the IB Diploma Program (International Baccalaureate). Express Publishing, 2018 Drien, Marcos, 2021

Justification: Aggregation allows current user to access all Objects in system. This fulfills both success criteria because it allows formore efficient, connected system. It allows easier modification and reuse of code, therefore making it more efficient and organised.

Explanation: Student has subject; AA hasstudent; Student has university; Student haspath of study. If I need to change the contents of a subject, I don't need to modify all classes, and can only modify Subject class.

Code Example:

```
public class Student extends Person implements Serializable {

    private AcademicAdvisor AA; //Aggregation
    private ArrayList<University> universityList = new ArrayList<>();
    private ArrayList<String> applyCountry = new ArrayList<>();
    private ArrayList<Message> messages = new ArrayList<>();
    private ArrayList<Subject> subjects = new ArrayList<>();
    private ArrayList<Grades> subjectGrade = new ArrayList<>();
    private ArrayList<Grades> effortGrade = new ArrayList<>();
```

Polymorphism/Overloading

Success Criteria: ● System should allow AA to create and assign student accounts to class/groups ● System should allow deleting, editing and changing grades, documentation (AAs and teachers only), choices, profile (students only), etc
Source: Anderson, Julie. Java Illuminated. Jones & Bartlett Learning, 2018. 5th Edition. Open WorldCat Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science: For the IB Diploma Program (International Baccalaureate). Express Publishing, 2018 Drien, Marcos, 2021

Justification: Polymorphism allows methods to have same name but different input parameters. This allows for code reuse, making it more convenient, efficient and organised; there is no need to remember different method names -- often leading to confusion.

Explanation: AAs can remove students through two ways. They can remove a student through inputting his name or index of student. These methods are named same -- removeStudent -- but has 2 different input parameters: String and int.

Code Example:

```
public void removeStudent(String name) throws IOException {
    //removes student when inputted String name
```

```
public void removeStudent(int index) throws IOException {
    //removes student when inputted int index of student
```

Word Count: 994