Criterion C - Development

Introduction

This product is a Java IDE program. It allows the user to enter and save an athletic roster by inputting the various attributes of the roster and those of its players. A Java database is used to store this information, and a Netbeans GUI, utilizing Java Swing components, such as combo boxes and panes, creates an interface in which the user may access and enter data. A number of algorithms will also search for and sort rosters or players, and calculate the deserved awards for players at the push of a button.

Word Count: 92

Summary of Techniques Used

- Parameter Passing
- Inheritance through a hierarchy
    - Abstraction
    - Extends
    - Super and Subclasses
- Writing to a file
    - File Writer
    - Bufferedwriter
- Reading from a file
    - Tokens and Tokenizer
    - Checking to see if file is empty
    - Checking to see if file exists
- Try and Catch
- For Loop
- While Loop
- Simple and Compound Selection
- Bubble Sort
- GUI Tabs
- Parsing
- Casting
- GUI Display Table
- Arrays and ArrayLists
    - Nested ArrayLists
        - Adding
        - Clearing
        - Getting Elements
- Switch
- Instantiation of Classes
- JOptionPane Popups

- Modifiers and Accessors
- String Comparison Using ASCII Values

Word Count: 81

Structure of the Program

What: Inheritance is used to create a hierarchy, extending the superclass Player. The Subclasses are the types of player, Varsity or JV, abstraction including attributes such as gender, and IASAS participation. The Roster class has an attribute which is an ArrayList of Players, and thus any Roster Object will have that ArrayList, as well as defining attributes such as roster type and year. The Roster of Players is then stored in an ArrayList of Rosters, written to and read from a specific file.

Why: Firstly, inheritance was used due to the necessities of the client. There had to be an ability to distinguish between the different types of players. Here, OOP was extremely beneficial, as inheritance and encapsulation meant that the attributes in the super class, attributes naturally held by each type, would be passed (inherited) on, and could only be modified via the methods provided publicly to the user. Abstraction also made certain that should the code be reused, certain parameters had to be satisfied. The ability to create multiple instances of the same object also proved to be a requirement in the development process, as various algorithms made use of the same methods.

Storing the data as a Roster of Players inside of another ArrayList meant that each roster could be treated as its own element/object, and therefore not only could players be sorted to allow for more efficient searching, but so could the Rosters themselves.

Word Count: 239

Data Structures Used

- ArrayList
    - Done due to the fact that it is unknown how many elements must be in the Array (To be determined by the user)
    - Efficient, allowing for a means of organizing Objects
    - ➢ ArrayLists Used:
        - ArrayList of the Object Player (The Roster)
            - Example Attributes:
                - Year
                - Sport
                - Season
- Files

    - Data had to be stored, available to be accessed at a later date. Therefore, a file was required in order to write and save data, as well as read the data.

Word Count: 78

Main Unique Algorithms

The Main unique algorithm in this product is the sequence of the file reading and writing methods:

Writing

- File created in a user public folder in the computer
- String tokenizer used to distinguish between elements (":")
- Ordering of writing to the file had to be kept track of, due to the storing of attributes of each ArrayList
    - Each attribute of the roster added to the same string
    - Each attribute of the players added to a second string.
    - Boths string written to the file together, appending, rather than overwriting.

Reading

- Reading the file had to correspond with the order in which the attributes were written in
    - Bufferedreader, and a next token method used to retrieve each attribute separately.
        - These were stored in an appropriate arrayList.
    - When writing each player, a value denoting the number of players in that roster was also included as a separate attribute.
        - This value was read using the bufferedreader to know when the next token was actually a new roster.
- While loop used to control this process.
- Method returns an ArrayList of the roster ArrayLists.

Method for writing to the file:

Method for reading from the file:

```
public ArrayList<Roster> getListVarsityMale() {

    //Contructing the placeholder variables
    ArrayList<Roster> holderList = new ArrayList<>();

    String sport = "";
    int season = -99;
    int year = -99;
    boolean isVarsity = false;
    int gender = -99;
    ArrayList<Player> playerList = new ArrayList<>();
    boolean tokens = true;
    int numPlayers = -99;

    //Trying to access the file
    try {

        //Filereader set to go directly to the user's home directory.
        FileReader fr = new FileReader(System.getProperty("user.home") + "/Athletics Awards Data/VarsityMale.txt");
        BufferedReader br = new BufferedReader(fr);
        String readIn = br.readLine();

        //Initializing and constructing the class that will check the tokens
        StringTokenizer st = new StringTokenizer(readIn, ":");

        while (tokens) {
            //Getting the attributes for the roster from the file
            sport = st.nextToken();
            season = Integer.parseInt(st.nextToken());
            year = Integer.parseInt(st.nextToken());
            isVarsity = Boolean.parseBoolean(st.nextToken());
            gender = Integer.parseInt(st.nextToken());
            numPlayers = Integer.parseInt(st.nextToken());
            //NEED SIZE OF ARRAYLIST OF PLAYERS TO BE ABLE TO READ THE PLAYERS
            int i = 0;
            //Reading the attributes for each player from the file
            while (i < (numPlayers * 5)) {
                playerList.add(new Varsity(st.nextToken(), st.nextToken(), Integer.parseInt(st.nextToken()), st.nextToken(), Boolean.parseBoolean(st.nextToken())));
                i = i + 5;

                //Conditional that breaks if no more tokens, since the sentinal uses next token.
                //Thus error occurs if there are no more tokens. Hence the break here.
                if (!st.hasMoreElements()) {
                    break;
                }
            }
            holderList.add(new Roster(playerList, sport, season, year, isVarsity, gender));
            //As the process is iterative, the playerList must be cleared after each iteration.

            //Updating the sentinel for the while loop.
            tokens = st.hasMoreElements();
        }


        //Catch in case the file cannot be accessed.
    } catch (Exception e) {
        System.out.println("Error in opening. Get Vmale");
    }
    return holderList;
```

Word Count: 189

<u>User Interface/GUI</u>

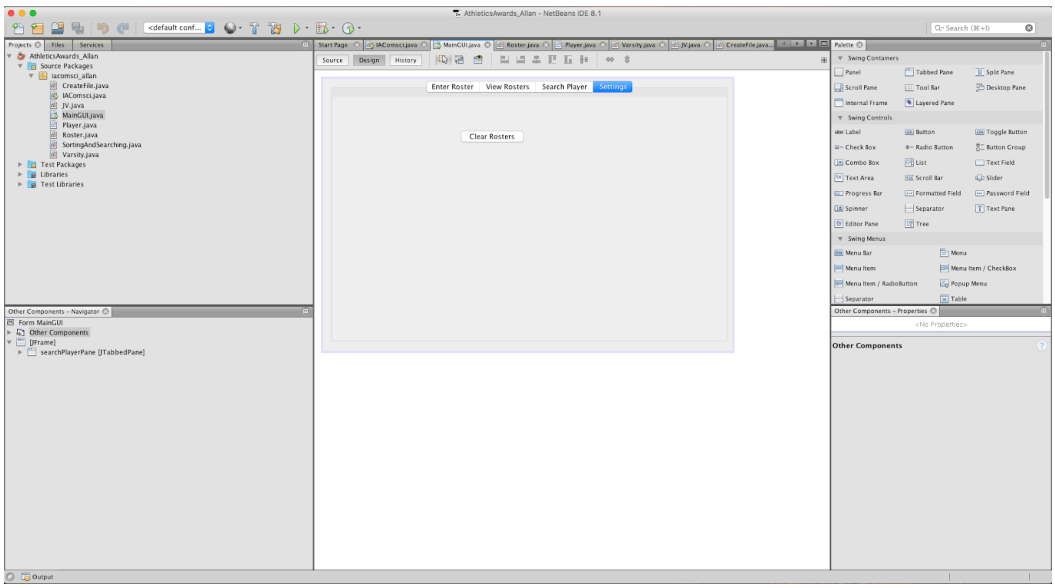
- Combo Box
    - A large part of the GUI, due to its ease of use for the user, as the options are already listed, and all the user has to do is click.
- Check Boxes
    - For simple yes or no questions in regards to attributes, in order to achieve efficiency and user-friendliness.

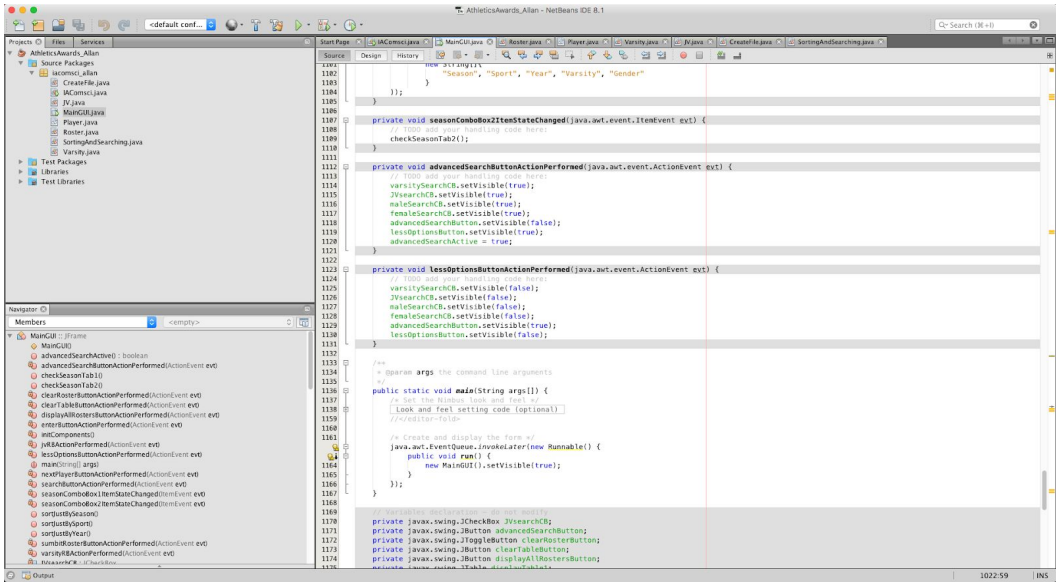
- Labels
    - To indicate to the user which field is for which attribute
- Display Table
    - A spreadsheet in a display table used to display the various rosters. This was used as it is organized and easy to read for the user.
- Tabs & Panes
    - Partitioning each function makes it clear which components do each task, and where those components are located.

Word Count: 110

<u>Software Tools Used</u>

- Netbeans IDE
    - The Netbeans IDE is a popular programming environment used by professionals around the world.
        - The reason it is most useful for me in this project is that it allows me to create an easy to use interface for the user (a requirement for the user's needs), utilizing Java Swing components. The software also implements encapsulation across various functions, such as supplying pre-coded methods, constructors, and shortcuts, making the development process much more efficient.





Word Count: 75