# Criterion C:

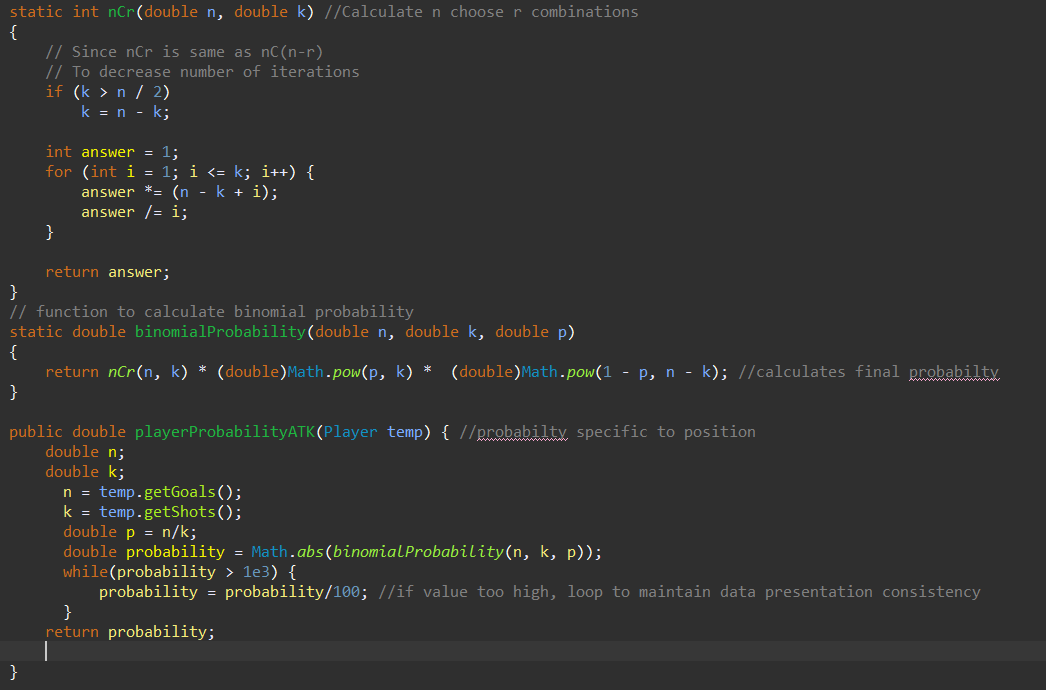
|  |
| --- |
| List of techniques used in the IA for functionality used but not included due to word count restrictions: |
| Abstraction, Inheritance, Encapsulation |

|  |
| --- |
| Technique: Statistical Analysis (Binomial distribution) |
| Link To Success Criterion:  The client must be able to search for specific players and obtain statistics and ranking |
| Source: (“*Binomial.Java.”)* |

Justification: For further accuracy of the ranking system, the “expectedGoals, expectedTackles etc.” is calculated using a binomial distribution. This statistical analysis calculates the probability of a successful tackle, goal or any other key statistic for the position and influences the ranking along with the coach's significance. This is an important part of the algorithm as the probability can greatly influence the ranking and is a key factor to understand the player’s ability in their position.

Explanation: The formula is calculated using 3 methods, nCr, BinomProbability and PlayerProbality to change the function based on position of player. Combinations (nCr) was calculated using for loops, rest was done using the Math library for exponential functions.

Example:



|  |
| --- |
| Technique: Sorting |
| Link To Success Criterion: The client must be able to search for specific players and obtain statistics and ranking |
| Source: (‘Collections.Sort() in Java with Examples’.) |

Justification: The imported data needs to be sorted in order for the client to choose the best possible players based on the rank which is sorted in descending order. It sorts by rank or position allowing greater functionality and efficiency during selection.

Explanation:

The sortByRank is sorted using insertion sort, sortByPosition uses the inbuilt Java arrays library (collections)

Example:



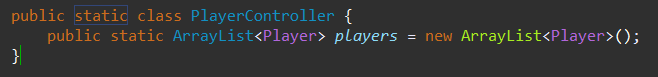
|  |
| --- |
| Technique: Data Structures (Array Lists) |
| Link To Success Criterion:  The client must be able to search for specific players and obtain statistics and ranking  The client must be able to edit, enter and delete specific statistics and/or players if necessary |
| Source: (Dimitriou, Kostas, and Markos Hatzitaskos) |

Justification: ArrayLists in the player class is to store all the player objects in the system. Allowing the client to sort and search by player and ranking

Explanation:

ArrayLists can hold many instances of the player object, hence can be stored and manipulated easily. Each element of the player arraylist contains each statistics which can be used by the ranking and sorting algorithms. As each element is a consistent statistic across all types of players, it improves efficiency and simplifies the programming required. It was chosen for its flexibility in size allowing potential new statistics to be added and the system remaining functional.

Example:

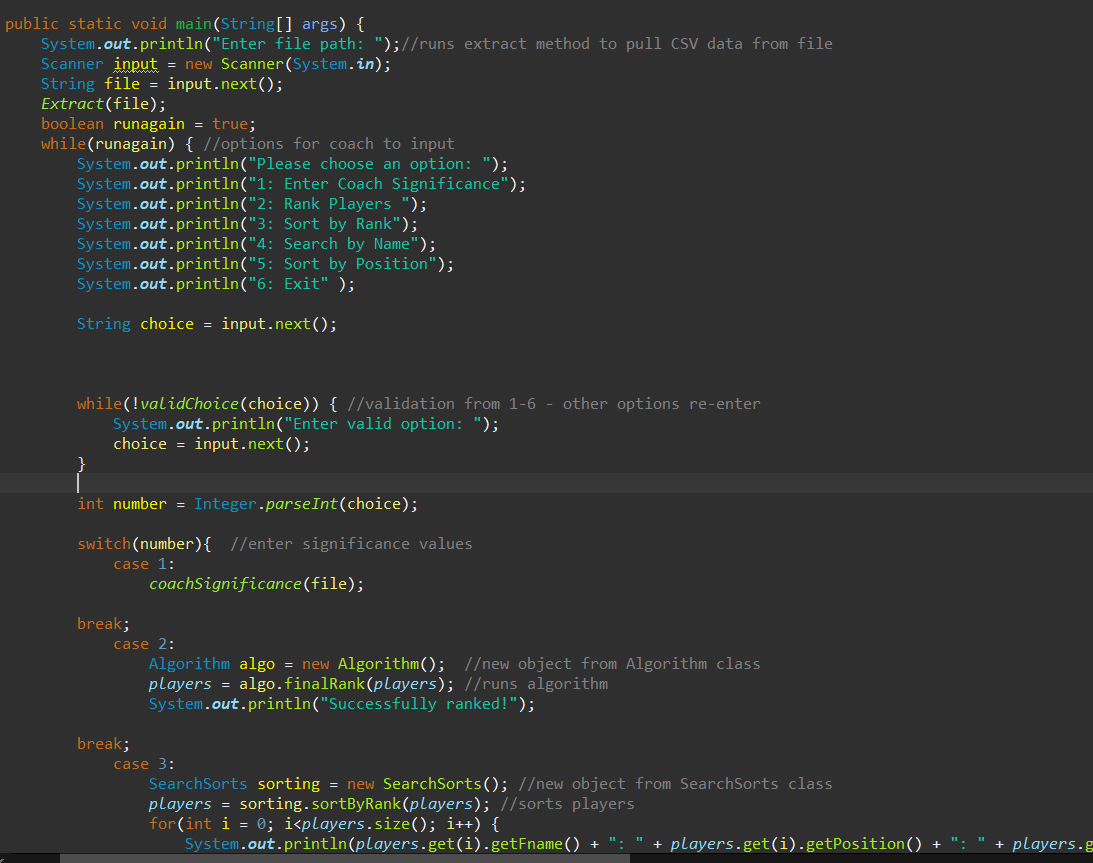


|  |
| --- |
| Technique: Menus (Switch Case) |
| Link To Success Criterion: The client must be able to navigate the CLI with ease: within 5 actions the required function must be found |
| Source: (The Switch Statement (The JavaTM Tutorials > Learning the Java Language > Language Basics).) |

Justification: The client should be able to navigate the menu with ease, intuitive GUI controllers aid in efficiency and ease of use.

Explanation: Data-entry intensive tasks will be done using a CLI (faster entry as only integers are involved). Ranking, searching and other tasks will be done with a CLI using switch case statements by entering appropriate numbers for navigation and inputs where necessary.

Example:



|  |
| --- |
| Technique: File Management (Saving/Loading) |
| Link To Success Criterion:  The data must be able to export as a CSV file for the whole team  The system must allow the head coach some influence on the ranking  The system should be able to save data preventing a reset with each use of the program |
| Source: (Shah, App.) |

Justification: FileManagement in the controller classes ensures the input data by the client is saved and loaded at a later instance when the system is used again. This is necessary as without saving/loading data, repeatedly entering data for functionality is ineffective as a product.

Explanation: To load and save edits, the PlayerController class saves any inputs such as coach significance whenever the specified methods are referenced in the code. The methods use java classes/libraries such as FileReader/FileWriter and BufferedReader/Writer.

Example:

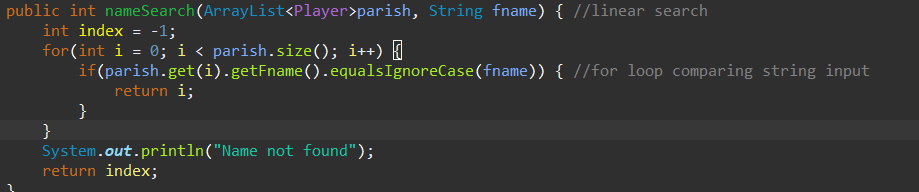


|  |
| --- |
| Technique: Data structure traversal (Searching - Linear Search) |
| Link To Success Criterion: The client must be able to search for specific players and obtain statistics and ranking |
| Source: (‘Linear Search in Java - Javatpoint’) |

Justification: Search methods in the SortSearch class enables the client to search players by name, rank, position.

Explanation: A linear search algorithm is used which iterates through the ArrayList until (or if) the specified input is found.

Example:



|  |
| --- |
| Technique: Overloading |
| Link To Success Criterion:  The client must be able to edit, enter and delete specific statistics and/or players if necessary  The system must allow the head coach some influence on the ranking |
| Source: (Drien Vargas) |

Justification: Overloading allows methods to be used with the same name. This is necessary as the calculatedRank class is initialised as null and another instance of the same class is used to calculate the rank using the ArrayList and rank algorithm.

Explanation: There are multiple methods with the same name, the distinction is the parameters involved in which the rank is either initialised as null or calculated at a later time.

Example:





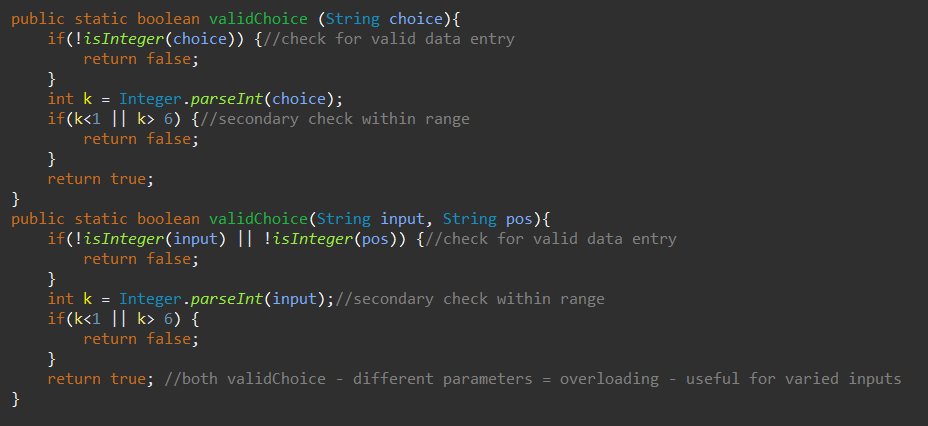
|  |
| --- |
| Technique: Validation (while loops/conditional statements) |
| Link To Success Criterion:  The system should be able to validate and verify data entry (incorrect format/data type) witch appropriate error catching |
| Source: (“Atkins, John”) |

Justification: Ensures valid data input which stops the system from recording and saving incorrect/invalid data with potential negative consequences on the system at a later launch.

Explanation:

Using while-loops, the system forces the client to re-enter information until the input is valid, preventing invalid data input.

Example:

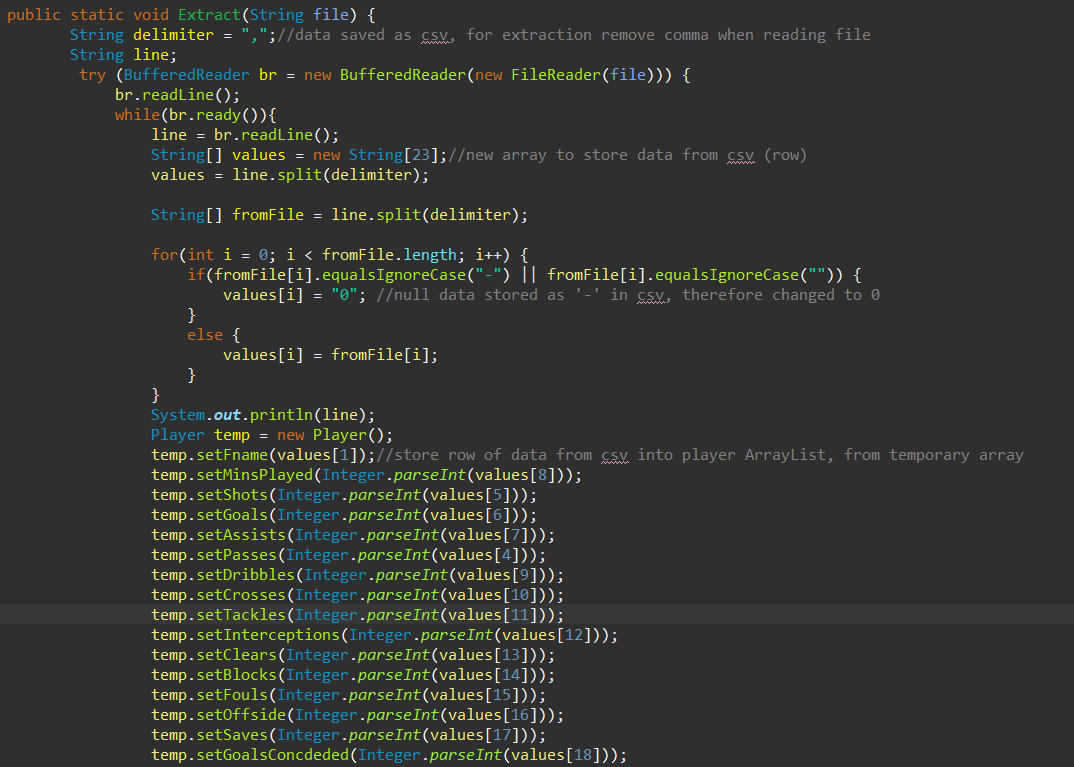


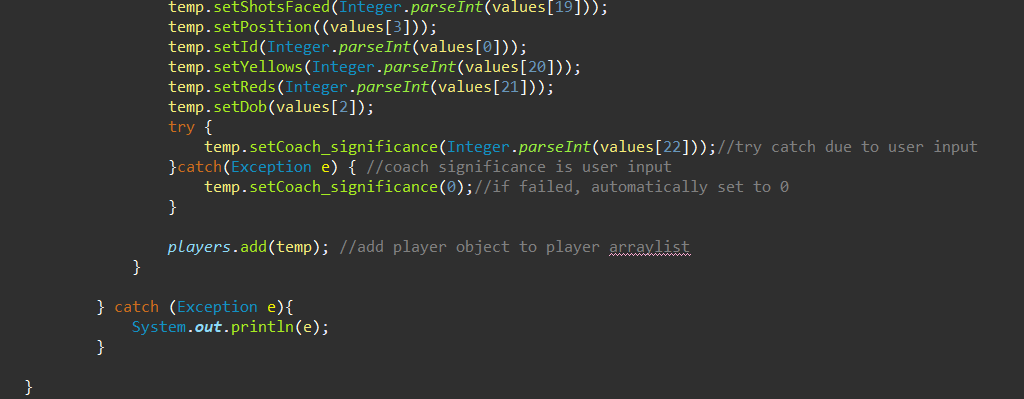
|  |
| --- |
| Technique: Exception Handling |
| Link To Success Criterion:  The system should be able to validate and verify data entry (incorrect format/data type) with appropriate error catching |
| Source: (‘Java Try-Catch - Javatpoint’) |

Justification: Try/catch used to handle potential issues in reading data from CSV files and handling user inputs.

Explanation: Some user inputs are not limited to integers (string is used if a position change is required”, hence exception handling used to avoid the OutOfBounds error. Using try/catch improved the effectivity of validation as it is more capable than simple while loops

Example:





|  |
| --- |
| Technique: Polymorphism |
| Link To Success Criterion:  The client must be able to search for specific players and obtain statistics and ranking  The data must be able to export as a CSV file for the whole team |
| Source: (Dimitriou, Kostas, and Markos Hatzitaskos.) |

Justification: When viewing sorted/unsorted data, ToString method is used to make printing information for viewing easier.

Explanation: Needed to display data in a format requested by the client which is easily readable. (Drien Vargas)

Example:



Word Count: 853