Class 1: Scores

* pubic:
* Scores();
  + Constructor for Scores
* ~Scores();
  + Destructor for Scores
* Scores(int score, int name, int size);
  + Constructor with variables for Scores
* int getScore(int index) const;
  + Getter for the individual scores in the array
* int\* getScoreArr() const;
  + Getter for a pointer to the Scores array
* void setScore(int score, int index);
  + Setter for adding numbers to the array
* unsigned int getSize() const;
  + Way to measure the size of the array
* void printScore(int size) const;
  + Print method to print out the entire array
* enum Location;
  + List of Enum’s for affecting the set name function
* void setParkName(int parkName);
  + Way to let the player choose where the game will be played, which will change the size of the array
* string get playerName() const;
  + Way of getting the player name to add to the score array
* void setName(string playerName);
  + Way to set the players name using user input
* private:
* int\* courseScores;
  + Static Array for holding the scores
* int score{ 0 };
  + Int to pass into the array
* int parkName{ 0 };
  + Int to set the location and the size of the array
* int size{ 0 };
  + Int for the size of the array
* string playerName{ “ “ };
  + name of the person playing

Class 2: Pars

* pubic:
* Pars();
  + Basic constructor for Pars
* ~Pars();
  + Destructor for Pars
* int getPar(int index) const;
  + Getter for a number par in the par array
* void setRegularPar(int par, int index);
  + Setter for the Regular par in the array
* void setYouthPar(int par, int index);
  + Setter for the Youth par in the array
* void printPar(int size) const;
  + Prints the whole par array
* int\* getParArr() const;
  + Getter for the Pars Array
* private:
* int\* coursePars;
  + Static Array to hold Par

Class 3: Results

* pubic:
* Results();
  + Basic constructor for Results
* ~Results();
  + Destructor for Results
* int\* getResultArr() const;
  + Getter for the Results array
* void compareScore(int\* coursePars, int\* courseScores, int size);
  + Compairs the inputed score to the par and then returns the +/- score
* void printResults(int size) const;
  + Print function that prints the results array
* void printFinalCompare(int\* coursePars, int\* courseScores, int size) const;
  + Print function that prints out the par, score, and results arrays
* void printTotal(int\* coursePars, int\* courseScores, int size);
  + Print function that prints out the added up scores, par, and total arrays
* void addToMap(Results courseResults);
  + Add the most recent game into the first multi-map
  + Add the most recent game into the second multi-map and then ranks the total score to adjust its position
* private:
* int\* courseResults;
  + Static Array for the Results
* multimap<int, courseResults> recent;
  + Multi-Map to hold the games in chronological order
* multimap<int, courseResults> ranked;
  + Multi-Map to adjust the games in the order of the best game

Class 4: SoloGame: private Results

* pubic:
* SoloGame();
  + Basic constructor for Games
* ~SoloGame();
  + Destructor for Games
* int\* topScore(Results\* ranked);
  + Returns the top score that was achieved in the game set
* int\* allScore(Results\* recent);
  + Returns the score of all of the games played in the entered set
* private:

Class 4: GroupGame: private Results

* pubic:
* GroupGame();
  + Basic constructor for Games
* ~GroupGame();
  + Destructor for Games
* int\* topScoreGroup(Results\* ranked);
  + Calculates and returns the top score for each of the players
* int\* topScoreWinner(Results\* ranked);
  + Calculates and returns the top score for each of the players
* int\* allScoreGroup(Results\* recent);
  + Returns the score of all the player in all of the games played in the entered set
* private:

The idea that my partner and I had for our challenge project two was to continue the implantation of our disc golf scoring system. We plan to include four different classes for each step of the scoring process. There will be three separate courses with a fixed array tied to Pars, Scores, and Results. These arrays would have a static capacity of eighteen, and depending on the park that the user selects, the size will change accordingly. The first class will be to store the scores for each hole. The program will ask the user to input the score that they threw, which will then be placed into a static array based on how many holes the user plans on throwing. The second class will be the par’s for each course. It will contain an array for the course that was selected and then provide a par count for the array of throws. It will also give you the option to throw regular or youth par. This will change the number of throws needed for par while keeping the amount of holes the same. Finally, the last class will be to check the scores vs. the pars of the course that the thrower is at. It will return the score compared to the par of the whole so that you know your result at that hole. We will also include an inheritance of the results class along with two multi-maps. Results will keep track and rank all of the games that the user chooses to play. It will create objects depending on the user input and then store them in both multi-maps. The first multi-map will rank the games in the order that they were played. The second multi-map will rank the games in the order of the best scoring. At the end of the inputted games, the user will then have the option to see which games they did the best on and in which chronological order it was in. We will have our inheritance work in Results as well. We plan on making Results a super class with two children being solo and group play modes. This will let the user play with friends if they want to or play by themselves.