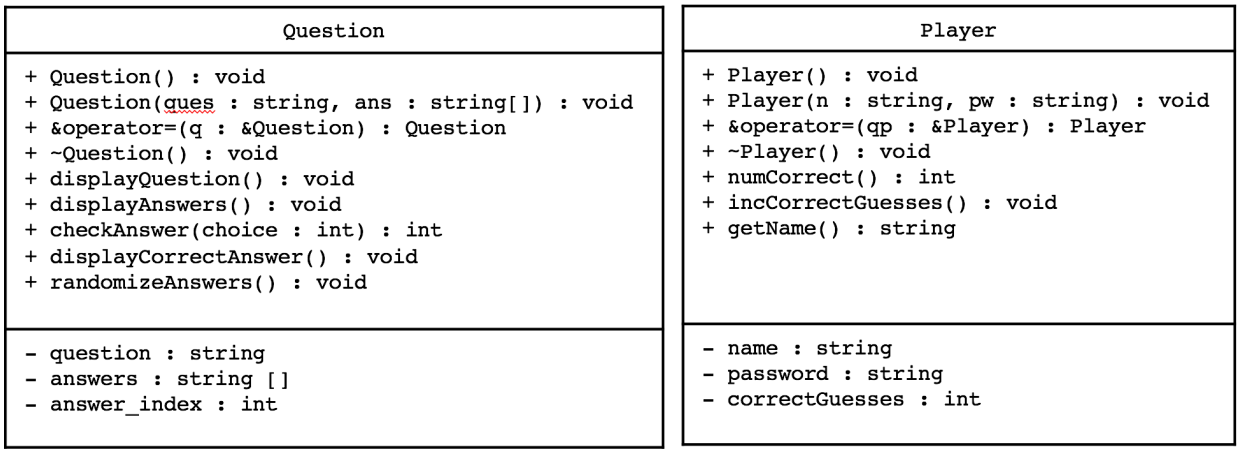
Eric Sehun Oh
CIS 22B

Homework 5 Report: ***Classes***

Data Structure



The two classes used in the program are the Question class and the Player class, as shown in the UML diagram above. When reading in the input file, Question objects are dynamically allocated and are referred to by a pointer to the dynamically allocated array called `Question *questions`. For players, two arrays called `Player players[50],` and `Players currentPlayers[2]` are used to keep track of all players and players playing the game, respectively. There are also two int variables called int num_questions and `int num_players` that are used to keep track of the number of questions and players.

Functions

Question Class Member Functions

```
Question() {};                              // Default constructor
Question(string, string[4]);
      // Custom constructor that takes in a string question and array of strings for
answers
Question& operator=(const Question &q); // Assignment operator
~Question() {};                             // Destructor
void displayQuestion() const;               // Displays the question
void displayAnswers() const;                // Displays list of answers
int checkAnswer(int choice) const;
    // Takes answer from player, checks answer and prints message accordingly (0 if
    //correct, 1 if wrong)
void displayCorrectAnswer() const;          // Displays correct answer.
void randomizeAnswers();
    // Randomize the answers of the question while maintaining correct answer index
```

Player Class Member Functions

```
Player();                                   // Default constructor
~Player() {};                               // Default destructor
Player(string n, string pw);                // Custom constructor
Player &operator=(const Player &qp);        // Assignment operator
```

```
int numCorrect() const;    // Returns the number of correct guesses for a player
void incCorrectGuesses();       // Increments the correctGuesses variable
string getName() const;         // Returns the name of the player
```

Standalone Functions

```
Question *readInFile(string fName, int &qNum);
```

The readInFile function take in a string for the filename, a reference to the number of questions and reads in the input file, reading in each line and returning a pointer to the array of question objects

```
void getPlayers(Player [], Player [], int &);
```

The getPlayers function takes in an array of Player objects for all and currently playing players and a reference to the number of players. The function prompts the players for their names and passwords, creates Player objects and stores them into the corresponding Player arrays. When it is run the first time, the two players are added into the Player array of all players, and sets the two players as currently playing the game. For every consequent time that it is run, it asks the two last players to have played if they would like to play again. Otherwise, it prompts for new players.

```
void swapQuestion(Question *q1, Question *q2);
```

The swapQuestion function takes two pointers to questions and swaps them.

```
void randomizeQuestions(Question *, int);
```

The randomizeQuestions takes in the point to questions array and the number of questions and randomizes the questions array.

```
void playGame(Question *, Player []);
```

The playGame function takes in a pointer to array of Questions and the array of players and allows the players to play the game. It displays 3 questions for each player but in corresponding order. It then checks the players input to check if it was the correct answer. If it was correct, it increments the number of correct guesses the player has and prints Congratulations. If invalid, it displays the correct answer and prints a message saying sorry.

```
void displayWinner(Player currentPlayers[]);
```

The displayWinner function takes in an array of players that are currently playing the game. It compares the number of correct guesses by each player and display a message about which player won the game.

```
void writeToFile(string fName, Player players[], int
num_players);
```

The writeToFile function takes in the file name, the array of players, and the number of players and write the names of each player that played the game to an output file.