Team Moon Cakes

Prof. Ron Mak

CMPE 135

9/26/2018

Assignment #3

For this assignment, our task is to create a simple rock, paper, scissors application as a command line tool and demonstrate how we implemented the Law of Demeter, used cohesive classes, and loosely-coupled classes.

To begin, the way that we chose to create our classes was to create a class for each party involved. We decided to encapsulate our classes into a class called Player and another class called Computer. Each class with hold each value corresponding each party. These values (which are private) include the choice for each party, the wins, losses, and name of each. The computer is a subclass of player which will have functions that determine its choice such as randomChoice and will show the previous successes and fails that the computer has. This will help with expanding on the code in the future.

|  |
| --- |
| Player.h |
| class Player  {  public:  Player();  ~Player();  void setScore(int s);  void setName(std::string n);  void setChoice(std::string c);  void setWins(int w);  void setLosses(int l);  void setTies(int t);  int getScore();  std::string getName();  std::string getChoice();  int getWins();  int getLosses();  int getTies();  protected:  int score;  int id;  int wins, losses, ties;  std::string name;  std::string choice;  }; |

|  |
| --- |
| Computer.h |
| class Computer : public Player  {  public:  Computer();  ~Computer();  std::string randomChoice();  void setPreviousChoice(Player &p);  void setPreviousSuccess(Player &p);  std:string getPreviousChoice();  private:  std::string previousChoice;  bool previousSuccess;  }; |

We are using the Law of Demeter in how we are organizing for the future changes. By definition, making classes using the Law of Demeter creates a program that does not need to know the details of how its input will affect the inner workings. For example, the input from the human player that decides what option of “rock, paper or scissors” for the first iteration of this program is a prompt from the computer. By designing the program using the Law of Demeter, when the program is rehashed, it will be simple to change the input without affecting other classes.

As aforementioned, the program contains a parent class titled “Player” for each of the parties involved in the game itself. Player will hold all the necessary values for comparison and the subclass computer will help with the computer’s decision making when playing the user. The Computer class inherits from the Player and receives information from public members only while maintaining data restrictions.

However, we have loosely coupled the class computer. In the computer class, we still hold the values that are corresponding to each party involved. But the member functions are different. The member functions for computer class help dictate what code should be used for the computer in the later stages of the project. For now this is a random choice that we choose. When we implement the machine learning algorithm in the future, the code will change showing the previous and next choices in the class and then making a decision based on that rather than a random number generator or some type of random choice made by the computer.

|  |
| --- |
| compare() |
| void compare(){  if ( (getP\_select() == "paper" && getC\_select() == "rock") ||  (getP\_select() == "scissors" && getC\_select() == "paper")||  (getP\_select() == "rock" && getC\_select() == "scissors"))  {  std::cout<< "Player Winner\n";  getloss();  getgamecount();  }  else if((getP\_select() == "rock" && getC\_select() == "paper")||  (getP\_select() == "paper" && getC\_select() == "scissors")||  (getP\_select() == "scissors" && getC\_select() == "rock")){  std::cout<< "Player Loses\n";  getloss();  getgamecount();  }  else if((getP\_select() == "rock" && getC\_select() == "rock")||  (getP\_select() == "paper" && getC\_select() == "paper")||  (getP\_select() == "scissors" && getC\_select() == "scissors")){  std::cout<< "Its a tie\n";  getloss();  getgamecount();  }  else{  std::cout<< "invalid response\n";  }  } |

Due to time constraints, the design specifications were not fully implemented as hoped for this assignment. Many changes were put into effect to create a workaround solution for minimum viable code that would compile without errors and display output as we would like. However, precautions are being taken into consideration to improvement the development cycle of the code base to match a less coupled, cohesive structure for the programs to run smoothly and be readable amongst group members and faculty.