Project 2

<Wheel of Fortune v2>

CSC17a-48096

Name: Javier Borja

Date: 12/9/2016

Table of Contents

Introduction

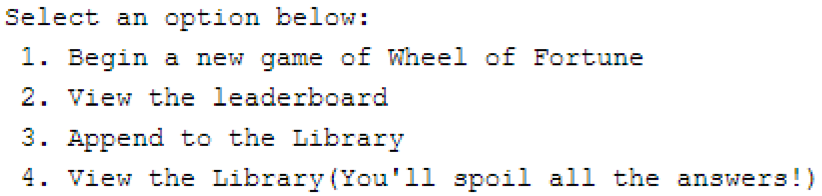
Welcome to Wheel of Fortune

For my second project I took my original project and rewrote it to reflect the concepts I learned in the second half of the semester, emphasizing on classes, inheritance, polymorphism, exceptions, and templates. I originally chose to recreate Wheel of Fortune because I find it fun and I thought it would create an interesting challenge to program. In the end, I found this development cycle to be incredibly helpful in further understanding of the concepts, and I am very proud in the effort that made this project possible.

Tutorial

(Note: You can input uppercase or lowercase characters throughout the game/menus)

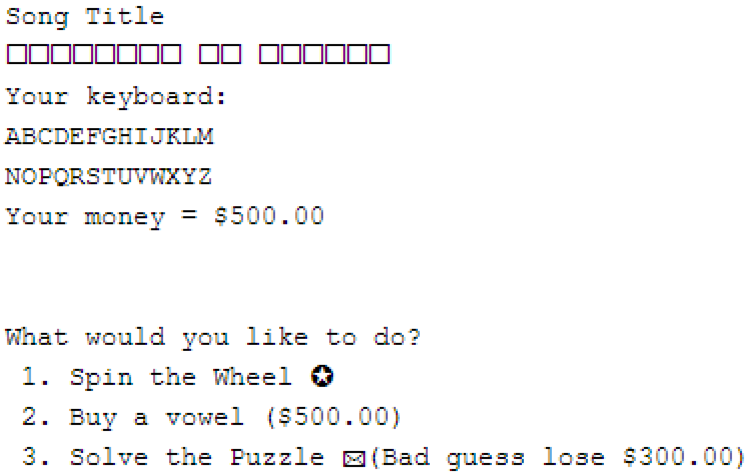
After inputting your name, you start with $500.00 and 0 Points. You will then be taken to a menu. There are four options: Play a game, view the leaderboard, add phrases to the library, or view the entire library. When you view the leaderboard, previous scores of other players will be listed from highest amount of points to the least amount of points. If you wish to append to the library, just follow the onscreen menus to do so. Again you do not need to uppercase any characters. Input validation is included in this program. Lastly, if you wish to view the library, you may do so. However, you will spoil all the answers.



```
Select an option below:
 1. Begin a new game of Wheel of Fortune
 2. View the leaderboard
 3. Append to the Library
 4. View the Library(You'll spoil all the answers!)
```

Playing a Game:

To win, you must guess the phrase; if you run out of money, you lose. Once you begin playing, you are given a category and phrase to guess. Displayed will be your hidden phrase with spaces, used/unused letters, and your money. Select an appropriate option to continue.

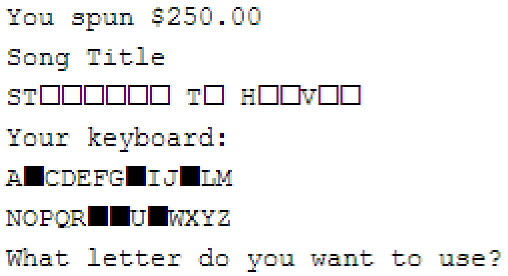


```
Song Title
□□□□□□□□ □□ □□□□□□
Your keyboard:
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
Your money = $500.00


What would you like to do?
 1. Spin the Wheel ✪
 2. Buy a vowel ($500.00)
 3. Solve the Puzzle 🖂(Bad guess lose $300.00)
```

Spin the Wheel:

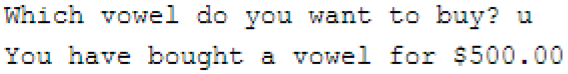
After spinning, you will be displayed a monetary value. If you correctly guess a letter, you will be awarded that amount of money and gain 10 points for each letter in the phrase that matched, else if you guess incorrectly, you will lose that amount. You can keep guessing if you have not used all the letters. Every letter you used will be blacked out and each letter you correctly guessed will be displayed.



```
You spun $250.00
Song Title
ST□□□□□□ T□ H□□V□□
Your keyboard:
A■CDEFG■IJ■LM
NOPQR■■U■WXYZ
What letter do you want to use?
```

Buy a Vowel:

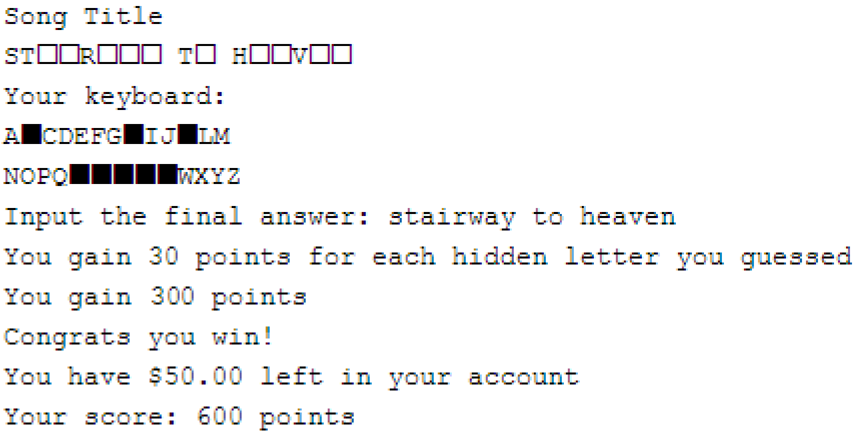
You will be displayed the same graphics as above, except you must buy a vowel. You will lose $500.00 for buying a vowel.



```
Which vowel do you want to buy? u
You have bought a vowel for $500.00
```

Solve the Puzzle:

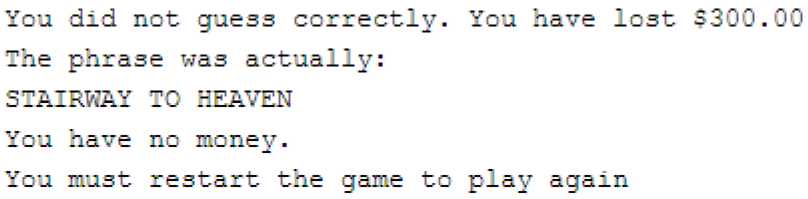
Input the phrase you think is the answer. You do not have to capitalize, but you do need to correctly match all the letters and spaces. If you incorrectly guess, you will lose $300.00; if you correctly guess, you will gain 30 points for each hidden letter revealed. You will then be displayed the amount of money left in your account and current amount of points you have earned.



```
Song Title
ST□□R□□□ T□ H□□V□□
Your keyboard:
A■CDEFG■IJ■LM
NOPQ■■■■■WXYZ
Input the final answer: stairway to heaven
You gain 30 points for each hidden letter you guessed
You gain 300 points
Congrats you win!
You have $50.00 left in your account
Your score: 600 points
```

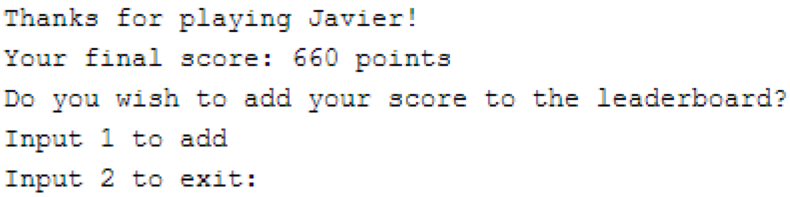
Losing and Leaderboard:

If you run out of money, the correct phrase will be displayed, you then lose the game and have to exit the program. But you will have an option to enter your score into the leaderboard.



```
You did not guess correctly. You have lost $300.00
The phrase was actually:
STAIRWAY TO HEAVEN
You have no money.
You must restart the game to play again
```

If you won, you can exit the program through the menu and still have a chance to enter your score to the leaderboard as well.



```
Thanks for playing Javier!
Your final score: 660 points
Do you wish to add your score to the leaderboard?
Input 1 to add
Input 2 to exit:
```

Have fun playing!

Project Summary

V1:

Project size	**557 Lines**
Lines of code	473 Lines
Comment lines	63 Lines
Blank lines	21 Lines

V2:

Project size	**1178 Lines**
Lines of code	799 Lines
Comment lines	216 Lines
Blank lines	163 Lines

V2 of this game was unexpectedly challenging, even though I was recreating V1. The reason V1 was simpler to code was because I did not have to worry about creating different classes, and passing variables between different functions was simple in V1 because I can just pass by reference; however, in V2 I had to create different pointer variables to pass arguments, and that created many problems of its own. I also had to create arrays of classes and that caused many crashes during development because I had to carefully destroy objects through the destructors.

As you can see from above, the introduction of classes actually increased the lines of code by over 300 lines for the same game. But the advantages are that the code is simpler to follow. There may be a lot of classes but each variable and member function is well commented, so it is actually much easier to read than version one. You may see the development process in my GitHub repository. Included are all previous versions, but running code from previous versions may not work and is not advised.

V2 of Wheel of Fortune starts from Version 5.

https://github.com/javierborja95/JB_CSC17a/tree/master/Project

Version 1

Main is developed along with a function that displays the menu. Functions are created to develop the library and read its contents. A header file is created and includes structures of future variables.

Version 2

The bulk of the program is developed in this stage. A game function is mostly completed with its necessary functions required for play.

Version 3

The point and money system is finalized in this version. The ability to write and read scores to binary files are possible and a leaderboard function to accompany these new features are developed. Comments are added to increase readability.

Version 4

Testing to catch possible glitches in the game are completed. There are very minor changes from version 3, but the game is finalized in this stage.

Version 5

I restarted the coding process. I created a Game class, and the backbones of Letter, Phrase, and Keyboard classes. Main only includes a menu interface to access future modes.

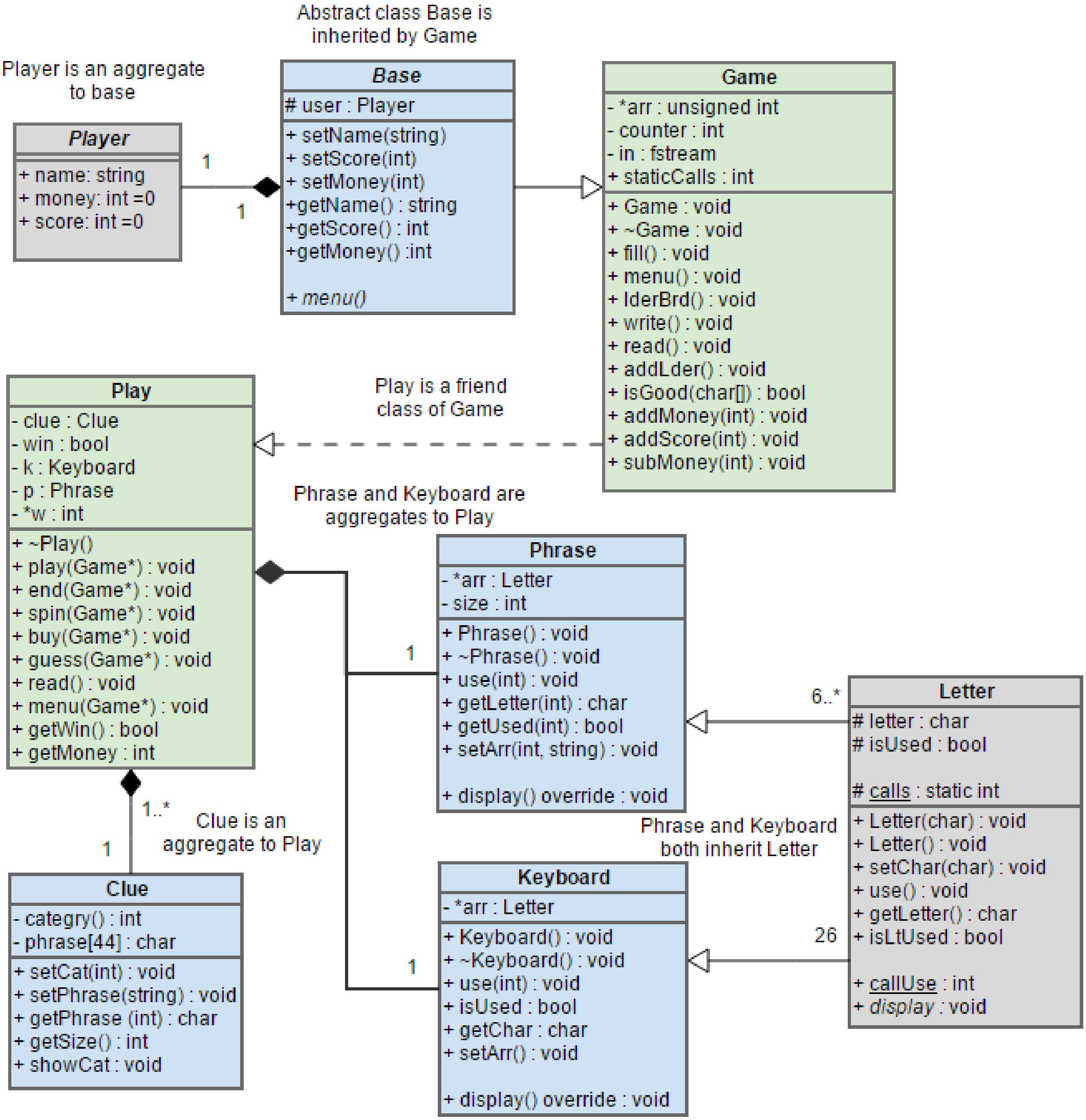
Version 6

This version includes a Play class, a friend of Game class, and in this class I have a sub menu of the actual game mode. In this version I design Phrase and Keyboard, and all the modes of Play class are working, but not as intended.

Version 7

This final development cycle was the longest of all the others. I implemented all other modes, including write, read, leaderboard and writing to leaderboard. I also tried to include missing constructs in this stage. Many problems arose when implementing missing constructs, especially the template construct. Pointer arrays had their own troubles as well, but in the end all problems were eventually solved. Final game is working as intended.

Class/UML Diagrams

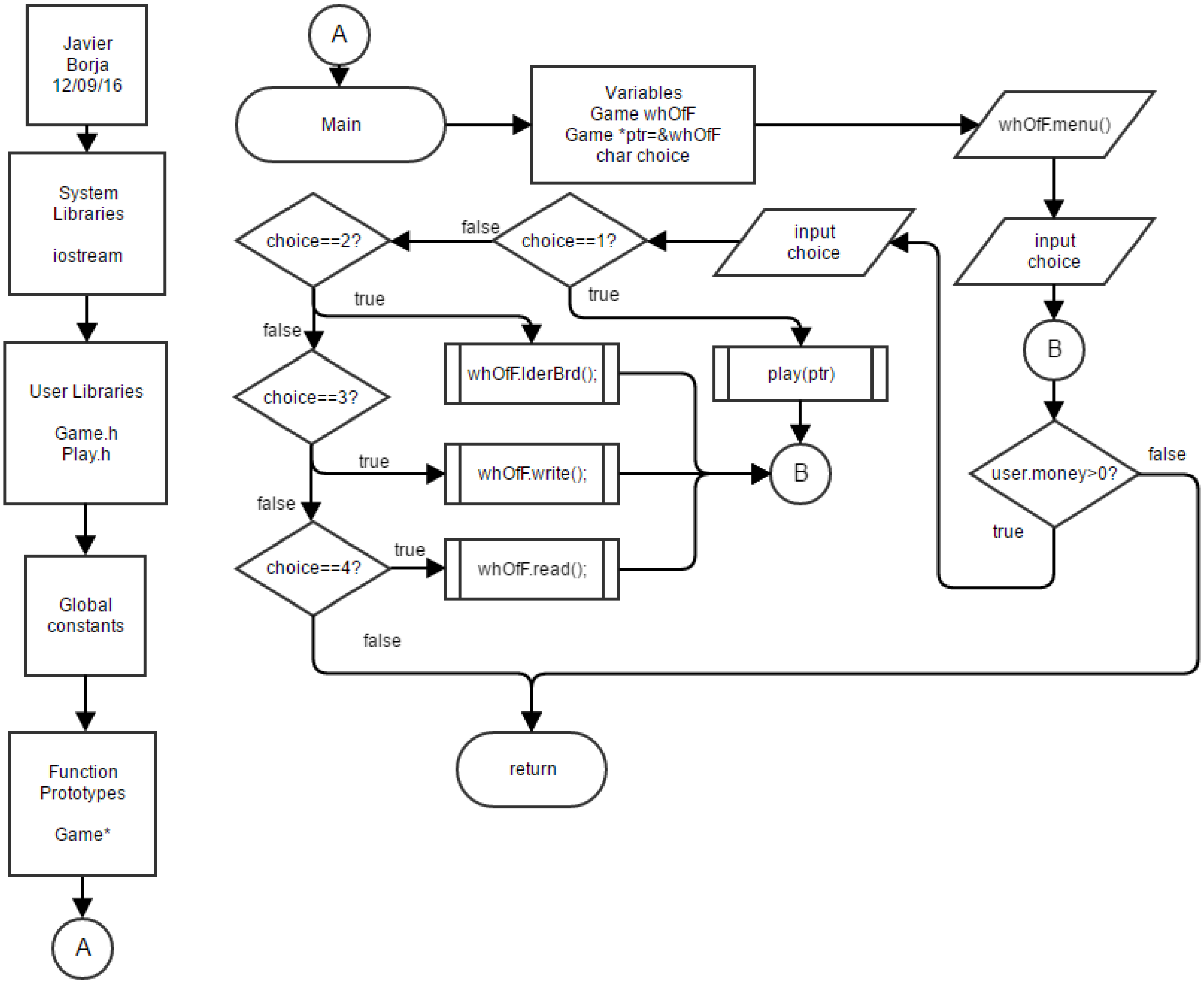


Pseudocode and Flowcharts

Because this V2 of Wheel of Fortune is inherited from V1, many of the functions from V1 are now member functions of different classes from V2. So many of the old pseudocodes and flowcharts are still relevant and applicable to this updated program. The biggest change in pseudocode and flowcharts are in Main and a new function prototype called Play; these will be displayed first. For all other reused flowcharts, I will specify with [brackets], where in code you can find the equivalent member function. I decided not to include simple setters and getters in these pseudocodes because most of the time it is just returning data or copying data into member variables.

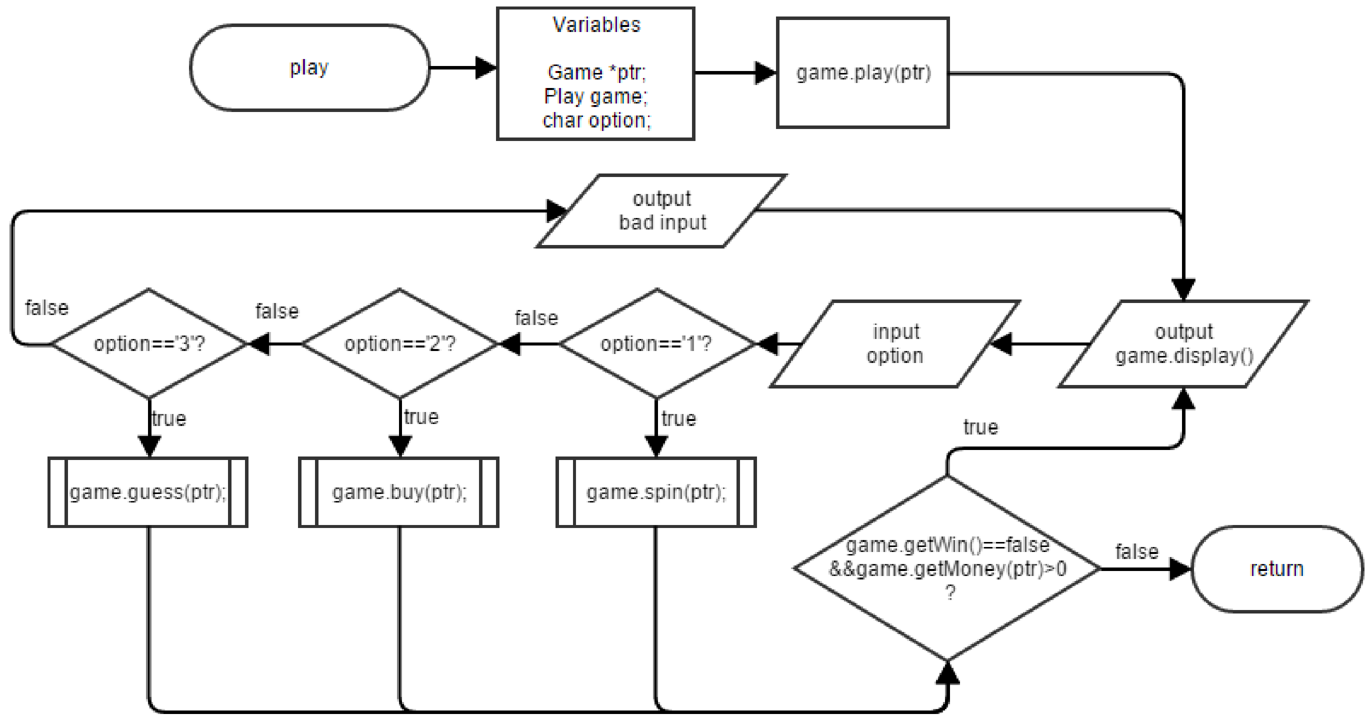
Main:

```
Create a Game object
Show Game.menu
Input menu choice
Do{
  Switch(choice)
    Case 1: Play()
    Case 2: Game.leaderBrd()
    Case 3: Game.write()
    Case 4: Game.read()
}While (choice is 1-4 and Game.getMoney is greater than 0)
```



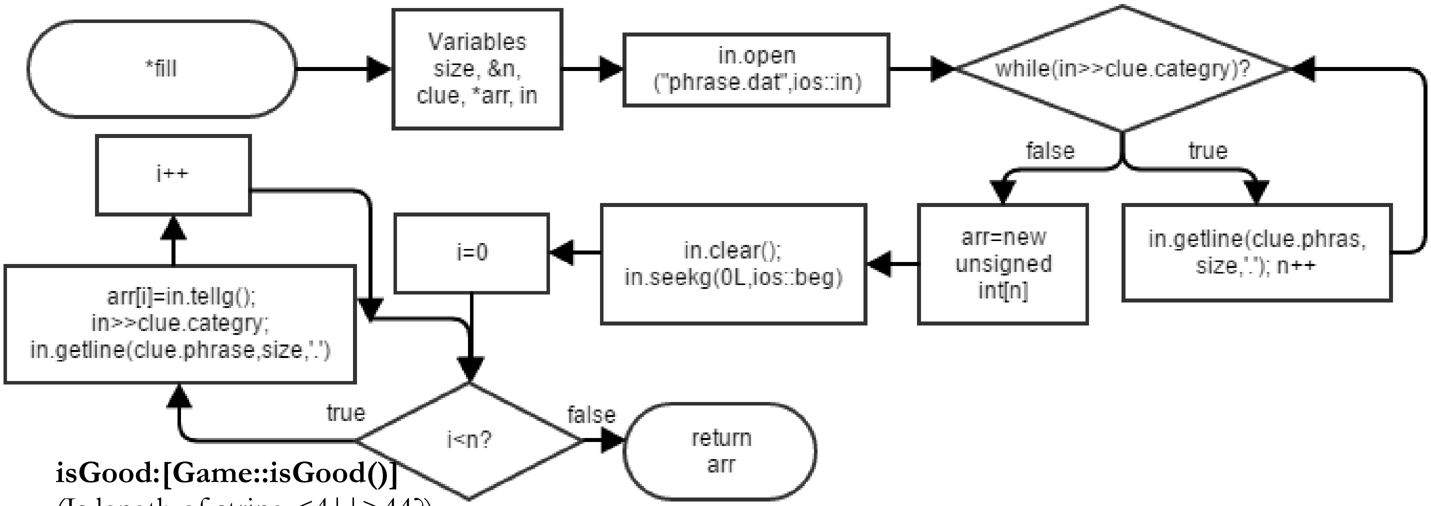
Play:

```
Create Play object
Play.play(with pointer to Game object)
Do{
    Output Play.display()
    Input option
    Switch(option){
        Case 1: Play.spin()
        Case 2: Play.buy()
        Case 3: Play.guess()
        Default: Output error message
    }
}While(Play.getWin()==false and play.getMoney() is greater than 0)
```



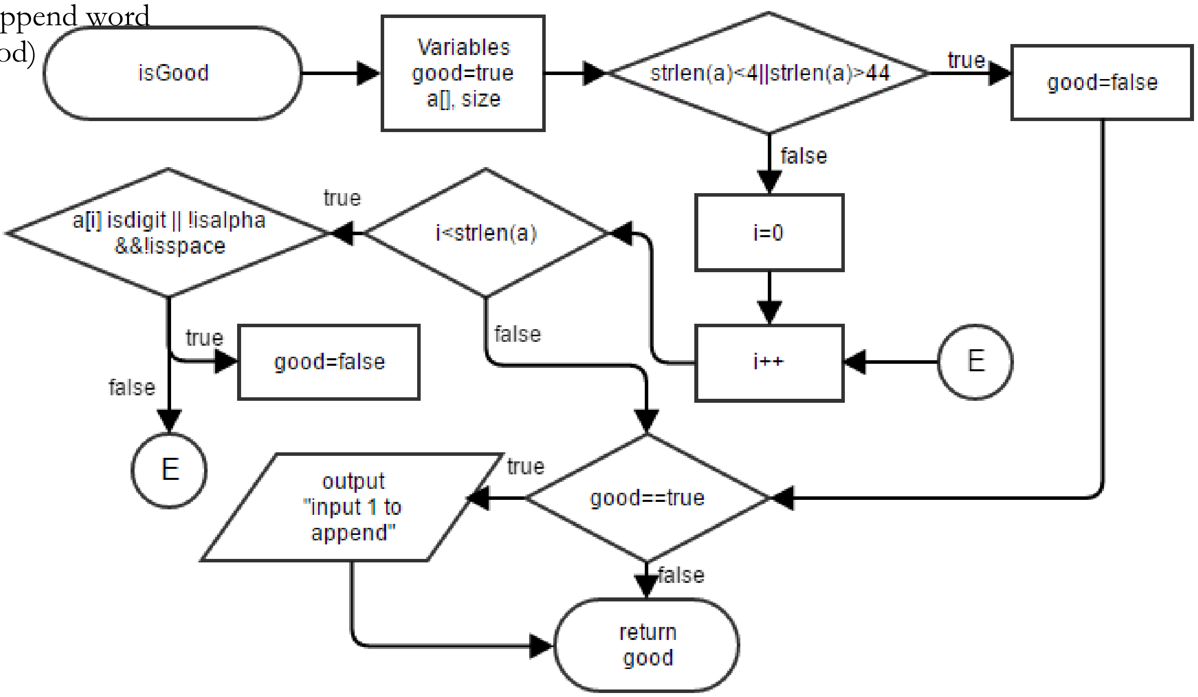
Fill: [Game::fill()]

```
Open phrase file
While(get category)
    Get clue phrase
    Size of array++
Allocate memory by size of array
Seek to beginning of file
For(i=0; i<size of array; i++){
    While(get category)
        Get clue phrase
        Set position of index
} return array
```



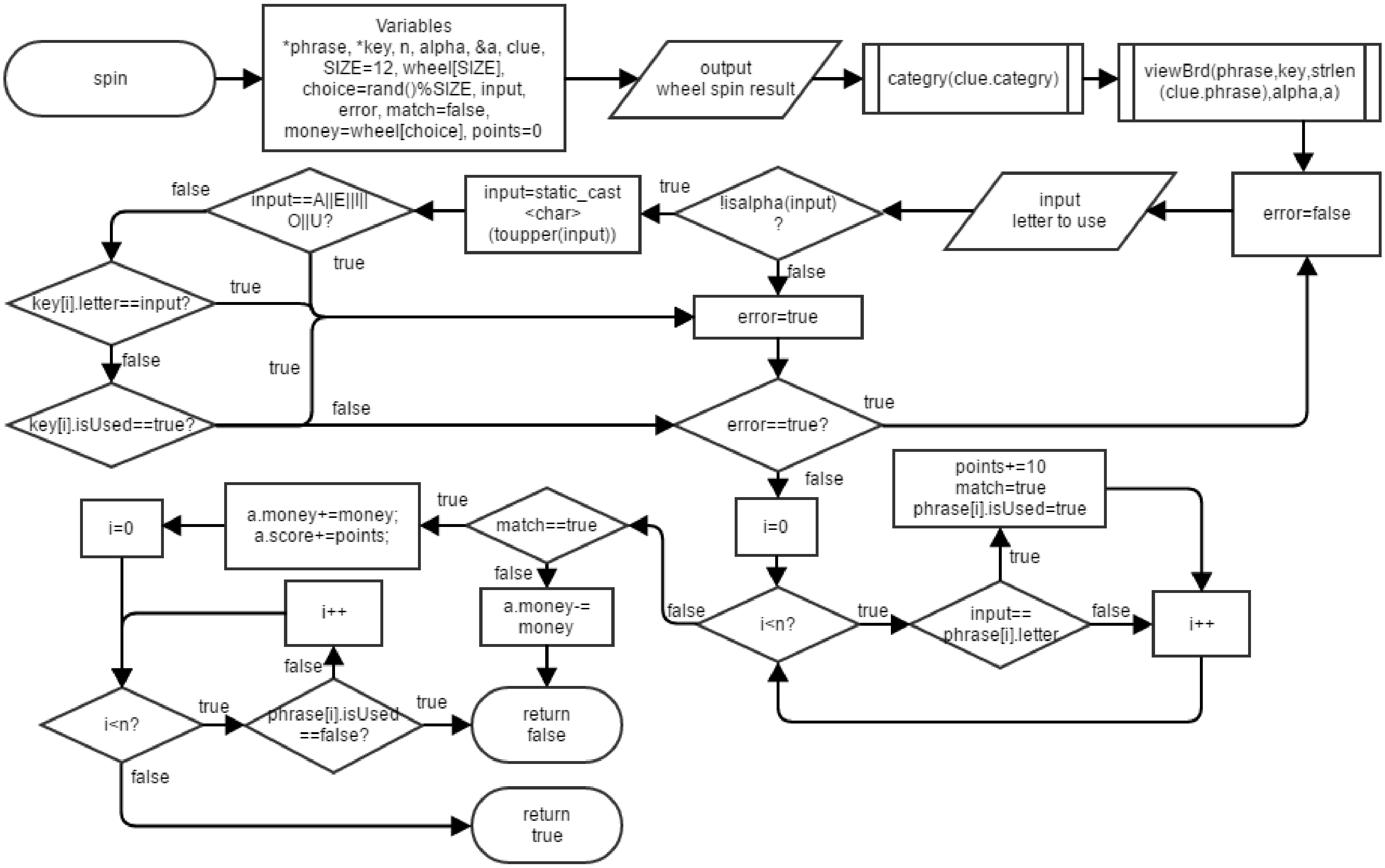
isGood:[Game::isGood()]

```
(Is length of string <4||>44?)
T: Good=false
F: For(i=0;i<length of string;i++){
    (If character is not letter or space) good=false
If(good==true)
T: Ask to append word
Return (good)
```



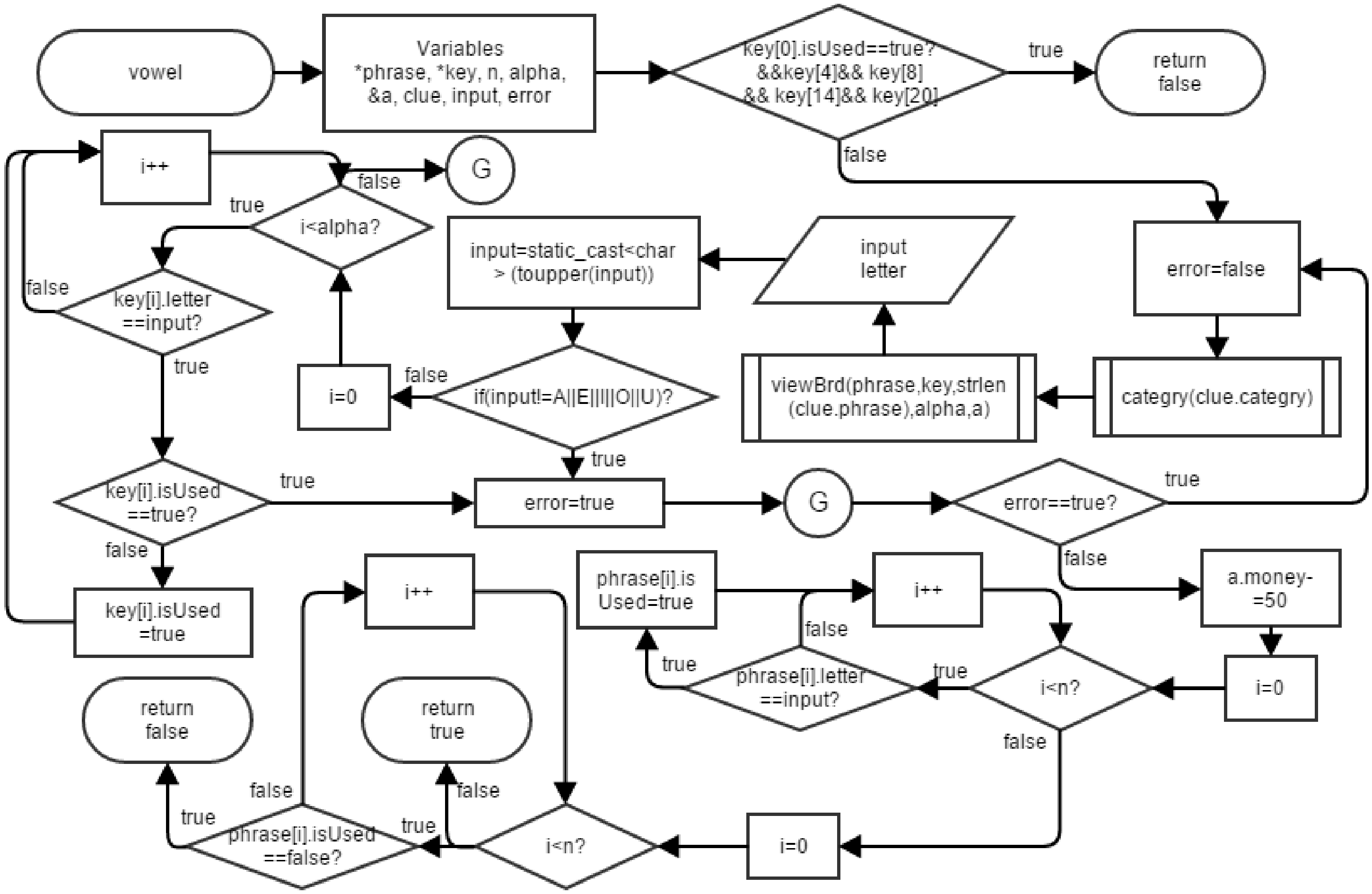
Spin:[Play::spin()]

```
Spin wheel
Show board and keyboard
Do{
   Error=false
   Input letter to use
   (is letter==vowel||non alphabet||or already used?)
      Error=true
While(error==true)
If(letter input==hidden letter)
   T: {add points
   Match=true}
If(match==true)
   T:{Add money
   Add points to score
   Make hidden letters shown
   (If all letters revealed) Return win
   }
   F: lose money, return loss
```



Vowel: [Play::buy()]

```
(if all vowels are used) return
Do{
    Error =false
    Show board and keyboard
    Input letter
    (if input is not vowel)
        Error =true
    (if vowel is used)
        T: error =false
        F: make key used
While(error==true)
Subtract money
Reveal vowels from phrase
(If all letters are revealed) return win
Else return loss
```



Guess:[Play::guess()]

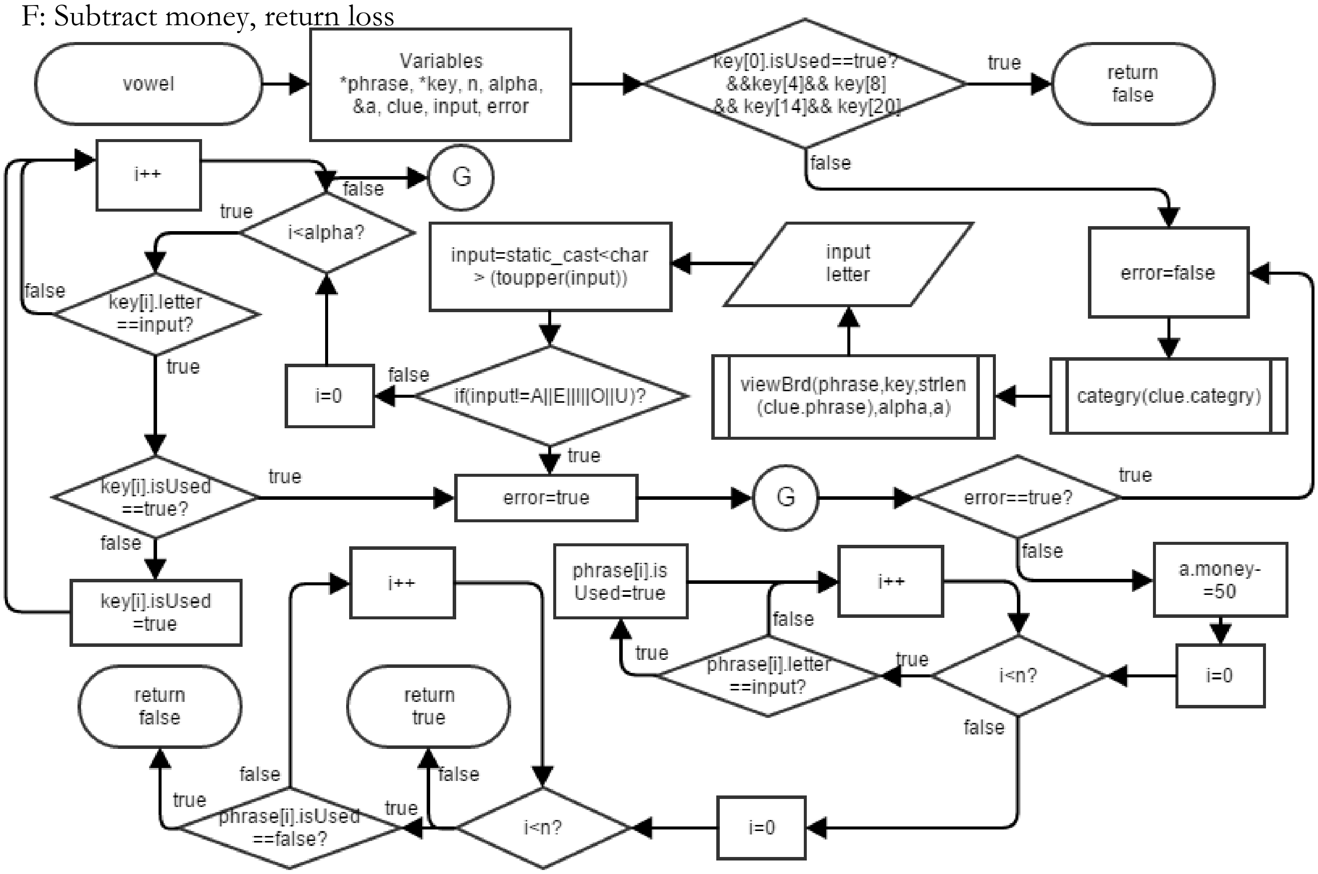
Show board and keyboard

Input phrase

(if input matches board phrase)

    T: Return win

    F: Subtract money, return loss



viewBrd:[Play::display][Keyboard::display()][Phrase::display()]

(If phrase letter is hidden)

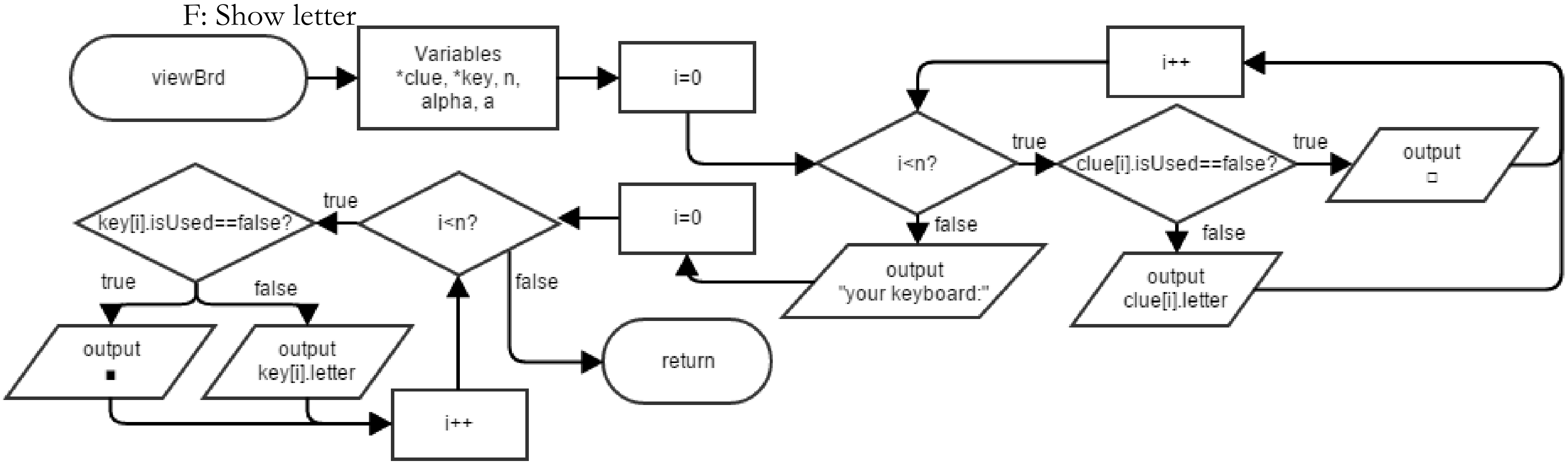
    T: Show square

    F: Show letter

(If keyboard letter is used)

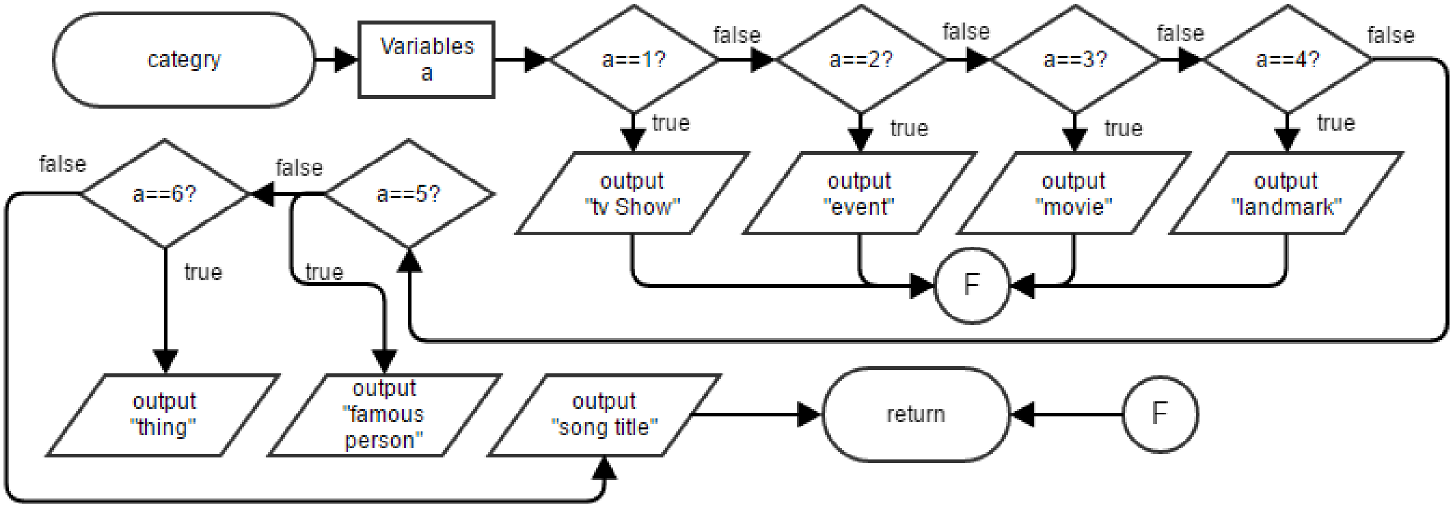
    T: Show square

    F: Show letter



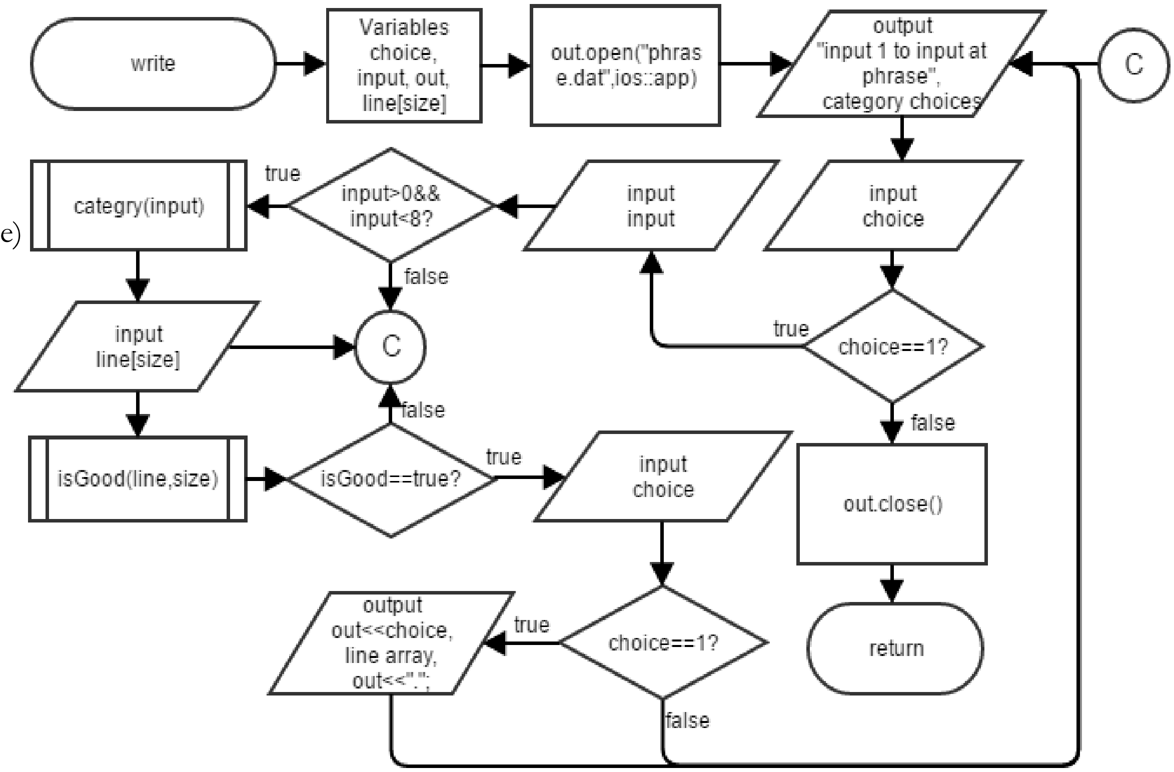
Category:[Clue::showCat()]

```
Switch(number)
   Display category based on number
```



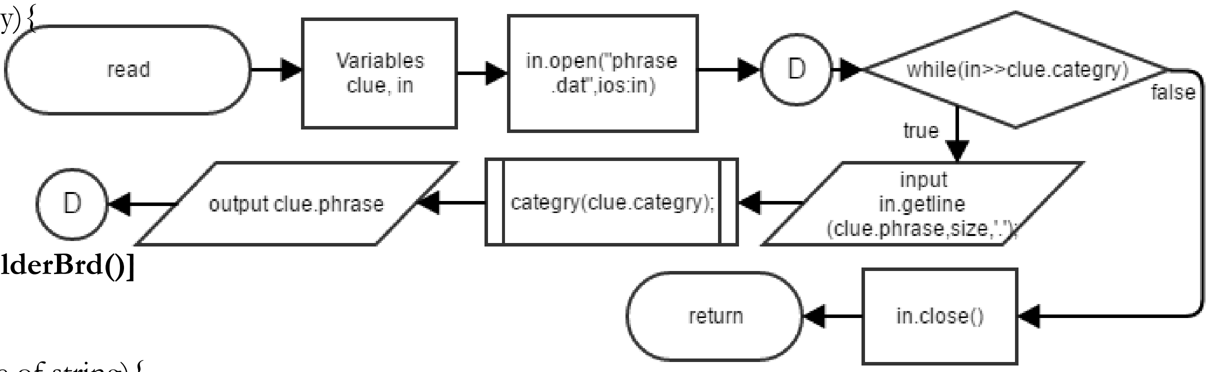
Write: [Game::write()]

```
Open file to append
Input category
Input phrase
isGood(phrase)
(if good)
   T:Ask to input
   Input choice
   (If choice is true)
      write to file
Close file
```



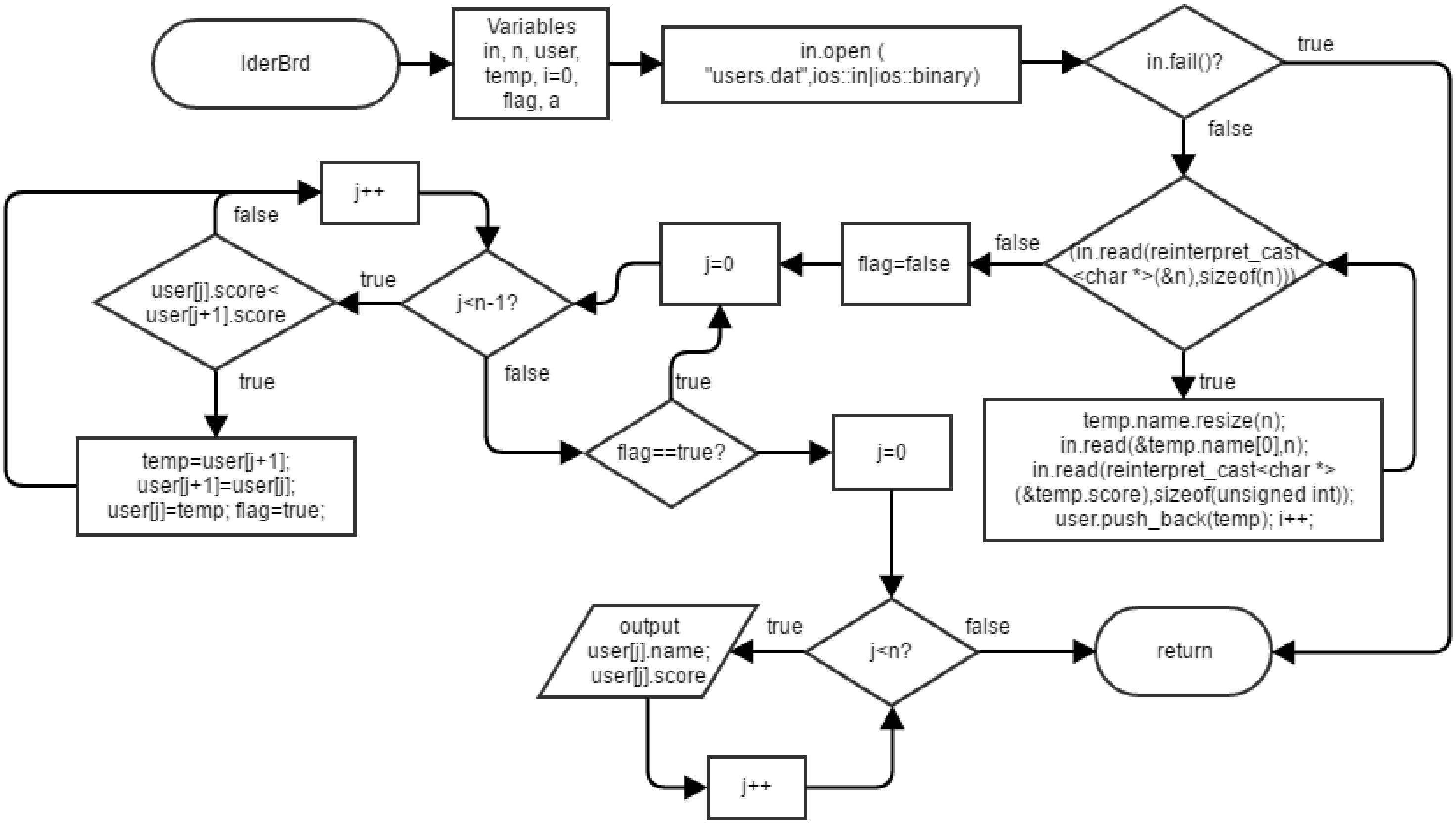
Read:[Game::Read()]

```
Open phrase file
(while get category){
   Categry()
   Output phrase
} close file
```



lderBrd:[Game::lderBrd()]

```
Open Binary file
If(fail) return
While(reading size of string){
   Get name string and score
   Push back array with name and score
}do{
   For(i=0;i<number of elements in array; i++){
      Flag=false
      If (element i score is greater>than element i+1 score)
         Swap
         Flag=true
}while(flag=true)
Display sorted names and scores
```


lderBrd
Variables in, n, user, temp, i=0, flag, a
in.open ("users.dat",ios::in|ios::binary)
in.fail()?
true
false
(in.read(reinterpret_cast <char *>(&n),sizeof(n)))
false
true
temp.name.resize(n); in.read(&temp.name[0],n); in.read(reinterpret_cast<char *> (&temp.score),sizeof(unsigned int)); user.push_back(temp); i++;
flag=false
j=0
j<n-1?
true
false
user[j].score< user[j+1].score
true
false
j++
temp=user[j+1]; user[j+1]=user[j]; user[j]=temp; flag=true;
flag==true?
true
j=0
j<n?
true
false
output user[j].name; user[j].score
j++
return

Major Variables:

These variables are private and protected members of the different classes. Local variables of different member functions are not included.

Type	Variable Name	Description	Location
Player	user	Contains user name, money and score	Play.h, Base.h
unsigned int	categry	Number that represents a category	Clue.h
char	phrase[44]	Phrase that the player will guess	Clue.h
unsigned int	*arr	Index array to find categories and clues	Game.h
int	counter	Size of library	Game.h
fstream	in	Input from file	Game.h
int	staticCalls	Keeps track of Letter calls	Game.h
Letter	*arr	Array of letters	Keyboard.h
char	letter	Character that makes up a phrase	Letter.h
Letter	*arr	Array of letters	Phrase.h
int	size	Size of array	Phrase.h
Clue	clue	Category and clue phrase	Play.h, Clue.h
bool	win	Keeps track of win/lose status	Play.h
Keyboard	k	A virtual keyboard that player can choose letters from	Play.h
Phrase	p	A phrase which hides its letters, player has to guess the phrase	Play.h
int	*w	Wheel of spin options	Play.h
string	name	Name of user	Player.h
int	money	User’s money	Player.h
unsigned int	score	User’s score	Player.h

Concepts utilized:

Missing constructs:

Multiple Inheritance- You can see the many different relationships between the classes, including regular inheritance, in the Class/ UML Diagram section. But not one class was inherited by two or more base classes.

Breadth of structure concepts- I have structures, but because of the inclusion of classes, I have limited the amount of structure concepts. I do not have pointers to structures or returning a structure from a function.

Enumerators from Chapter 11.12.

Included constructs:

Every other construct is listed below with the location in code. You can find the location with **ctrl-f** and copy/pasting. It will be in the last section: **Source Code**.

From: Tony Gaddis, Starting out with C++ *From Control Structures through Objects*, Eighth Edition.

Chapter	Construct	Location
9.2	Pointer Variables	unsigned int *arr;
9.3	Pointer Arrays	Letter *ptr=new Letter[n];
9.5	Initializing pointer as reference	Game *ptr=&whOfF;
9.7	Pointers as Function Parameters	void play(Game *);
9.8	Dynamic Memory Allocation	int *w=new int[WHEEL];
10.1	Character Testing	if(isdigit(a[i])||(!isalpha(a[i])&&!isspace(a[i])))
10.2	Character Conversion	cout<<static_cast<char>(toupper(a[i]));
10.3	C-string arrays	char line[SIZE];
10.4	Library Functions for Working with C-Strings	p.setArr(strlen(s),s);
10.6	Writing your Own C-String-Handling Functions	bool isGood(char[]);
10.7	More About the C++ String Class	Clue::setPhrase(string s){ … … phrase[i]=s[i];
11.1	Abstract Data Types	struct Player{ string name; int money; unsigned int score;
11.2	Combining Data into Structures	struct Player{ string name; int money; unsigned int score;
11.3	Accessing Structure Members	user.score+=n;
11.5	Arrays of Structures	vector<Player> user;

11.10	Knowing when to use ., ->, and *	if(p.getUsed(i)==false){
		a->in.getline(s,SIZE,'.');
		*(t1)+t2;

12.1	File Operations	fstream in;
12.2	File Output Formatting	cout<<setw(5)<<right<<user[j].score<<" points"<<endl<<endl;

12.4	Error Testing	if(in.fail()){cout<<"CRITICAL ERROR: File opening failed"<<endl;exit(1);
12.5	Member Functions for Reading and Writing Files	a->in.get();
12.7	Binary Files	in.read(reinterpret_cast<char *>(&temp.score),sizeof(unsigned int));
12.8	Creating Records with Structures	out.write(reinterpret_cast<char *>(&user.score),sizeof(unsigned int));
12.9	Random-Access Files	a->in.seekg(a->arr[index],ios::beg);
13.2	Introduction to classes	class Letter{
13.3	Defining an Instance of a Class	Game whOfF;
13.4	Private Members	private: unsigned int categry;
13.5	Separating Class Specification from Implementation	“Clue.h” “Clue.cpp”
13.6	Inline Member Functions	char getPhrase(int i) {return phrase[i];}
13.7	Constructors	Game();
13.8	Passing Arguments to Constructors	Letter(char);
13.9	Destructors	~Play() {delete[] w;}

13.10	Overloading Constructors	Letter(char);
		Letter();

13.12	Array of Objects	Letter *arr;
13.16	UML Diagrams	(see Class/UML Diagram section)

14.1	Static Members	static int calls;
		static int callUse(){return calls;}

14.2	Friends of Classes	friend class Play;
14.5	Operator Overloading	void operator + (int n){money=money+n; }
14.7	Aggregation	class Play{ private: Clue clue;
15.1	Inheritance	class Phrase: public Letter{
15.2	Protected Members	class Letter{ protected: char letter; bool isUsed; static int calls;
15.4	Redefining Base Class Functions	void display() override;
15.6	Polymorphism	void Phrase::display(){ … … arr[i].display();

15.7	Abstract Base Classes	class Base{ …
		… virtual void menu()=0;

16.1	Exceptions	try{… … throw "Input must be part of the alphabet";… … catch(char const* s){
16.2	Function Templates	template<class T1,class T2> void add(T1 *t1,T2 t2){ *(t1)+t2; }
16.5	Introduction to Standard Template Library	vector<Player> user;… ... temp.name.resize(n);… … user.push_back(temp);

References:

Gaddis, Tony. Starting out with C++ from Control Structures through Objects. 8th ed. Pearson Addison-Wesley, 2014. Print.

http://wheeloffortuneanswer.com/

Copied phrases to fill dictionary.

Source Code:

```
/*
 * File:   main.cpp
 * Author: Javier Borja
 * Created on December 7, 2016, 1:00 PM
 * Purpose: Wheel of fortune. Player guesses a phrase with category as a clue.
 */

//System Libraries
#include <iostream>   //Input/Output
using namespace std;

//User Libraries
#include "Game.h"
#include "Play.h"

//Global Constants

//Function Prototypes
void play(Game *);

//Execution

int main(int argc, char** argv){
    //Variables
    Game whOfF;       //Wheel of fortune Game object
    Game *ptr=&whOfF; //Pointer to pass Game object
    char choice;      //Menu choice

    //Input Data
    do{
        whOfF.menu();
        cin>>choice;
        cin.ignore();

    //Process Data
        switch(choice){
            case'1':{
                play(ptr);
                break;
            }
            case'2':{
                whOfF.lderBrd();
                break;
            }
            case'3':{
                whOfF.write();
                break;
            }
            case'4':{
                whOfF.read();
                break;
            }
        }
    }while((choice=='1'||choice=='2'||
            choice=='3'||choice=='4')&&whOfF.getMoney()>0);

    //Process Data

    return 0;
}

void play(Game *ptr){
    //Variables
    Play game;      //Play object
    char option;    //Menu option
    game.play(ptr); //Start playing

    //Input Data
    do{
        game.display(); //Display hidden phrase and available keyboard letters
        do{
            game.menu(ptr); //Display menu
            cin>>option;
            cin.ignore();
            switch(option){
                case'1':
                    game.spin(ptr);
                    break;
                case'2':
                    game.buy(ptr);
                    break;
                case'3':
                    game.guess(ptr);
                    break;
                default: cout<<"ERROR: Bad Input"<<endl;
            }
        }while(option<49||option>51);
    //Loop until win or lose
    }while((game.getWin()==false)&&(game.getMoney(ptr)>0));
}
/* File:   Base.h
 * Author: Javier B
 * Created on December 9, 2016, 5:40 PM
 * Purpose: Class Specification File for a Base class
 */

#ifndef BASE_H
#define BASE_H

//System Libraries
using namespace std; //Namespace of the System Libraries

//User Libraries
#include "Player.h"

class Base{ //A class meant to be inherited as a base to an actual game class
```

```
    protected:
        Player user;
    public:
        //Mutators
        void setName(string s)        //Sets a player's name
        {user.name=s;}
        void setScore(unsigned int n) //Sets a player's score
        {user.score=n;}
        void setMoney(int n)          //Sets a player's money
        {user.money=n;}

        //Accessors
        string getName(){return user.name;}
        unsigned int getScore(){return user.score;}
        int getMoney(){return user.money;}

        //Member Function
        virtual void menu()=0; //Displays a menu
};

#endif /* BASE_H */
/* File:   Clue.h
 * Author: Javier B
 * Created on December 5, 2016, 6:21 PM
 * Purpose: Class Specification File for Clue class
 */

#ifndef CLUE_H
#define CLUE_H

//System Libraries
#include <string>    //Strings
#include <iostream>  //Input/Output
#include <fstream>   //File input/Output
#include <cstring>   //Cstrings for strlen() function
using namespace std; //Namespace of the System Libraries

//User Libraries

class Clue{
    private:
        unsigned int category;//Number to represent a category
        char phrase[44];     //Max Phrase length
    public:
        //Mutators
        void setCat(unsigned int);
        void setPhrase(string);

        //Accessors
        char getPhrase(int i)
        {return phrase[i];}
        int getSize()
        {return strlen(phrase);}

        //Output
        void showCat();
```

```
};

#endif /* CLUE_H */
/* File:   Game.h
 * Author: Javier B
 * Created on December 5, 2016, 5:19 PM
 * Purpose: Class Specification File for Game class
 */

#ifndef GAME_H
#define GAME_H

//System Libraries
#include <cstdlib>   //Random seed
#include <ctime>     //Time
#include <fstream>   //File Input/Output
#include <iomanip>   //Output manipulation
#include <vector>    //Vectors
using namespace std; //Namespace of the System Libraries

//User Libraries
#include "Base.h"
#include "Player.h"
#include "Clue.h"
#include "addSub.h"

//Variables
const int SIZE=44; //Max Size of Char array

class Game: public Base{
    private:
        unsigned int *arr;         //Index array to find categories and clues
        int counter;               //Size of Library
        fstream in;                //Input

    public:
        int staticCalls; //Keeps track of Letter calls, just for fun
        //Constructor
        Game();  //Introduction, sets random seed, creates library

        //Destructor
        ~Game(); //Deletes library, closes file streams, appends to leaderboard

        //Member Functions
        void fill();          //Creates an index to the library
        void menu();          //Displays the menu
        void lderBrd();       //Displays a leaderboard
        void write();         //Appends to the library
        void read();          //Displays the entire library
        void addLder();       //Adds profile to leaderboard
        bool isGood(char[]);  //Input verification

        //Add Functions
        void addMoney(int);
        void addScore(unsigned int);
        //Subtract Functions
        void subMoney(int);

        //Play class can access private members of Game class
        friend class Play;
};

#endif /* GAME_H */
/* File:   Keyboard.h
 * Author: Javier B
 * Created on December 6, 2016, 12:12 PM
 * Purpose: Class Specification File for Keyboard class
 */

#ifndef KEYBOARD_H
#define KEYBOARD_H

//System Libraries
#include <iostream>  //Input/Output
using namespace std; //Namespace of the System Libraries

//User Libraries
#include "Letter.h"

//Constants
const int ALPHA=26; //Size of the alphabet

class Keyboard: public Letter{
    public:
        Letter *arr; //Array of letters
    public:
        //Constructor
        Keyboard();

        //Destructor
        ~Keyboard();

        //Mutators
        void use(int i)
        {arr[i].use();}

        //Accessors
        bool isUsed(int i)
        {return arr[i].isLtUsed();}
        char getChar(int i)
        {return arr[i].getLetter();}

        //Member functions
        void display() override;
        void setArr();
};

#endif /* KEYBOARD_H */
/* File:   Letter.h
 * Author: Javier B
 * Created on December 5, 2016, 6:08 PM
 * Purpose: Class Specification File for Letter class
 */

#ifndef LETTER_H
#define LETTER_H

//System Libraries
#include <iostream>  //Input/Output
using namespace std; //Namespace of the System Libraries

//User Libraries

class Letter{
    protected:
        char letter;
        bool isUsed;
        static int calls;
    public:
        //Constructors
        Letter(char);
        Letter();

        //Mutators
        void setChar(char a)
        {letter=a;}
        void use()
        {isUsed=true;}

        //Accessors
        char getLetter(){return letter;}
        bool isLtUsed(){return isUsed;}
        static int callUse(){return calls;}

        //Member functions
        virtual void display(){cout<<letter;}
};

#endif /* LETTER_H */
/* File:   Phrase.h
 * Author: Javier B
 * Created on December 6, 2016, 12:12 PM
 * Purpose: Class Specification File for Phrase class
 */

#ifndef PHRASE_H
#define PHRASE_H

//System Libraries
#include <iostream>  //Input/Output
#include <string>    //String Library
using namespace std; //Namespace of the System Libraries

//User Libraries
#include "Letter.h"

class Phrase: public Letter{
    private:
        Letter *arr; //Array of letters
        int size;    //Size of array
    public:
        //Constructor
        Phrase();

        //Destructor
        ~Phrase();

        //Mutators
        void use(int i)
        {arr[i].use();}

        //Accessors
        char getLetter(int);
        bool getUsed(int);

        //Member Functions
        void setArr(unsigned int,string);
        void display() override;
};

#endif /* PHRASE_H */
/* File:   Play.h
 * Author: Javier B
 * Created on December 5, 2016, 8:56 PM
 * Purpose: Class Specification File for play class
 */

#ifndef PLAY_H
#define PLAY_H

//System Libraries
#include <string>    //Strings
using namespace std; //Namespace of the System Libraries

//User Libraries
#include "Game.h"
#include "Keyboard.h"
#include "Phrase.h"

//Variables
const int WHEEL=12; //Size of wheel

class Play{
    private:
        Clue clue;   //Category and clue phrase
        bool win;    //Win or lose
        Keyboard k;  //Keyboard
        Phrase p;    //Phrase
        int *w;      //Wheel of spin options
    public:
        //Destructor
        ~Play()
        {delete[] w;}
        //Member Functions
        void play(Game*); //The actual game
        void end(Game*);  //Ending screen, win or lose
        void spin(Game*); //Spin the wheel
        void buy(Game*);  //Buy a vowel
        void guess(Game*);//Guess the phrase
        void display();   //Display the keyboard and hidden phrase
        void menu(Game*); //Outputs the game menu

        //Accessors
        bool getWin()          //Returns win boolean
        {return win;}
        int getMoney(Game *a)  //Returns player's money
        {return a->getMoney();}
};

#endif /* PLAY_H */
/* File:   Player.h
 * Author: Javier B
 * Created on December 5, 2016, 5:38 PM
 * Purpose: Struct Specification File for Player
 */

#ifndef PLAYER_H
#define PLAYER_H

//System Libraries
#include <iostream>
#include <string>
using namespace std; //Namespace of the System Libraries

//User Libraries

struct Player{
    string name;
    int money;
    unsigned int score;

    Player(){
        money=50; //Player starts with $500.00
        score=0;  //Player starts with 0 points
    }

    //Add money
    void operator + (int n){
        money=money+n;
    }
    void operator - (int n){
        money=money-n;
    }
};

#endif /* PLAYER_H */
/* File:   addSub.h
 * Author: Javier B
 * Created on December 8, 2016, 1:19 PM
 * Purpose: Specification File for adding and subtracting templates
 */

#ifndef ADDSUB_H
#define ADDSUB_H

//System Libraries
using namespace std; //Namespace of the System Libraries

//User Libraries

template<class T1,class T2>
void add(T1 *t1,T2 t2){
    *(t1)+t2;
}

template<class T1,class T2>
void sub(T1 *t1,T2 t2){
    *(t1)-t2;
}

#endif /* ADDSUB_H */
/* File:   Clue.h
 * Author: Javier B
 * Created on December 5, 2016, 6:21 PM
 * Purpose: Class Implementation File for Clue class
 */

//User Libraries
#include "Clue.h"

void Clue::setCat(unsigned int n){
    category=n;
}

void Clue::setPhrase(string s){
    //Input Data
    for(int i=0;i<s.length();i++){
        phrase[i]=s[i];
    }
}

void Clue::showCat(){
    //Output Data
    switch(category){
        case 1:
            cout<<"TV Show"<<endl;
            break;
        case 2:
            cout<<"Event"<<endl;
            break;
        case 3:
            cout<<"Movie"<<endl;
            break;
        case 4:
            cout<<"Landmark"<<endl;
            break;
        case 5:
            cout<<"Famous Person"<<endl;
            break;
        case 6:
            cout<<"Thing"<<endl;
            break;
        default:
            cout<<"Song Title"<<endl;
    }
}
/* File:   Game.h
 * Author: Javier B
 * Created on December 5, 2016, 5:19 PM
 * Purpose: Class Implementation File for Game class
 */

//User Libraries
#include "Game.h"

Game::Game(){
    //Set Random seed
    srand(static_cast<unsigned int>(time(0)));

    //Fill Library
    counter=0;
    fill();

    //Input
    cout<<"Input your name: ";
    getline(cin,user.name);
    cout<<"Welcome to Wheel of Fortune "<<user.name<<"!\n";
}

Game::~Game(){
    //Close Files
    in.close();

    //Deallocate Memory
    delete[] arr;
    arr=nullptr;

    //Output Data
    char choice;
    cout<<"Thanks for playing "<<user.name<<"!"<<endl;
    cout<<"Your final score: "<<user.score<<" points"<<endl;
    cout<<"Do you wish to add your score to the leaderboard?\n"
            "Input 1 to add\n"
            "Input 2 to exit: ";
    cin>>choice;
    if(choice=='1'){
        addLder(); //Add to leaderboard
    }
    cout<<"Fun fact: You called class Letter ";
    if(staticCalls<0||staticCalls==32768) cout<<"0 times"<<endl;
    else cout<<staticCalls<<" times"<<endl;
}

void Game::addLder(){
    //Variables
    fstream out; //Output in binary
    int n;       //Size of string

    //Output Data
    out.open("users.dat",ios::out|ios::app|ios::binary);
    n=user.name.size();
    out.write(reinterpret_cast<char *>(&n),sizeof(n));
    out.write(user.name.c_str(),n);
    out.write(reinterpret_cast<char *>(&user.score),sizeof(unsigned int));
    cout<<"Your score has been added"<<endl;

    //Close files
    out.close();
}

void Game::fill(){
    //Variables
    char s[SIZE];     //Temp char array
    unsigned int a;   //Temp char

    //Open File
    in.open("phrase.dat",ios::in);
    if(in.fail()){
        cout<<"CRITICAL ERROR: File opening failed"<<endl;
        exit(1);
    }

    //Input Data
    while(in>>a){
        in.getline(s,SIZE,'.');
        counter++;    //Add to size of array
    }

    //Allocate Memory
    arr=new unsigned int[counter];
    in.clear();
    in.seekg(0L,ios::beg); //Go back to beginning of file

    //Process Data
    for(int i=0;i<counter;i++){
        arr[i]=in.tellg(); //Each index has a position
        in>>a;
        in.getline(s,SIZE,'.');
    }
}

void Game::lderBrd(){
    //Variables
    fstream in;         //Input from file
    int n;              //Size of string that is read from file
    vector<Player> user;//Array of Player structures
    Player temp;        //Temp Player to swap for
    int i=0;            //Size of array
    bool flag;
    string a;           //Player inputs to continue
    try{
        //Open files
        in.open("users.dat",ios::in|ios::binary);
        if(in.fail()){
            throw "users.dat not found";
        }
        //Input Data
        while(in.read(reinterpret_cast<char *>(&n),sizeof(n))){ //Get size of string
            temp.name.resize(n);     //Resize string size by size
            in.read(&temp.name[0],n);//In name and score
            in.read(reinterpret_cast<char *>(&temp.score),sizeof(unsigned int));
            user.push_back(temp);    //Push back array by one at a time
            i++;
        }

        //Process Data
        do{
            flag=false;;
            for(int j=0;(j<i-1);j++){
                if(user[j].score<user[j+1].score){ //Swap greatest to least
                    temp=user[j+1];
                    user[j+1]=user[j];
                    user[j]=temp;
                    flag=true;
                }
            }
        }while(flag==true);

        //Output Data
        cout<<"Sorted Leaderboard:"<<endl;
        for(int j=0;j<i;j++){
            cout<<user[j].name<<endl;
            cout<<setw(5)<<right<<user[j].score<<" points"<<endl<<endl;
        }
        cout<<"Press enter to continue";
        getline(cin,a);
    }
    catch(char* const s){
        in.close();
        cout<<s<<endl;
    }

    //Close files
    in.close();
}

void Game::read(){
    //Variables
    Clue clue;      //Temporary Clue to fill
    unsigned int n; //Categories are numbered
    char s[SIZE];   //String to hold phrase
    //Open File
    in.clear();
    in.seekg(0L,ios::beg); //Go back to beginning of file

    //Input Data
    while(in>>n){          //Repeat until in can't extract a char
        in.getline(s,SIZE,'.');

    //Output Data
        clue.setCat(n); //Set category
        clue.showCat(); //View category
        cout<<s<<endl;  //Output string
    }

    cout<<"Input anything to continue: ";
    cin.get();
}

void Game::write(){
    //Variables
    char choice;    //Menu choice
    char input;     //Input for sub-menu
    fstream out;    //Output to file
    char line[SIZE];//Character array of size=44
    Clue clue;

    //Open File
    out.open("phrase.dat",ios::app);

    //Input Data
    cout<<endl<<"Input 1 to input a phrase\n"
            "Input 0 to exit: ";
    cin>>choice;
    cin.ignore();
    if(choice=='1'){
        cout<<endl<<"Input a category:\n";
        for(int i=1;i<=7;i++){
            cout<<i<<" ";
            clue.setCat(i);
            clue.showCat();
        }
        cout<<endl<<"0 Exit"<<endl;
        cin>>input;
        cin.ignore();

    //Output Data
        if(input>48&&input<56){ //If input is '1'-'7'
            clue.setCat(input-48);
            cout<<"Input your phrase(max 44 characters): "<<endl;
            cin.getline(line,SIZE);
            if(isGood(line)){ //If input is good ask if wish to append
                cin>>choice;
                cin.ignore();
                if(choice=='1'){
                    out<<input;
                    for(int i=0;i<strlen(line);i++){
                        out<<static_cast<char>(toupper(line[i])); //Make uppercase
                    }
                    out<<"."<<endl;
                    cout<<"You must restart the game for effects to take effect"<<endl;
                }
            }
        }
    }

    //Close File
    out.close();
}

bool Game::isGood(char a[]){
    //Process Data
    try{
        if(strlen(a)<4||strlen(a)>44){ //If char array doesn't fit size limit
            throw "ERROR: Phrase must be greater than 3 characters and less than 44";
        }
        for(int i=0;i<strlen(a);i++){
            if(isdigit(a[i])||(!isalpha(a[i])&&!isspace(a[i]))){//If not space or letter
                throw "ERROR: Input must be characters only\n";
            }
        }

    //Output Data
        cout<<"Do you really wish to add the following phrase?"<<endl;
        for(int i=0;i<strlen(a);i++){
            cout<<static_cast<char>(toupper(a[i]));
        }
        cout<<endl<<endl<<"Input 1 to append\n"
                "Or anything else to cancel: ";
        return true;
    }

    //Catch errors
    catch(char const* s){
        cout<<s<<endl;
        return false;
    }
}

void Game::menu(){
    //Output Data
    cout<<"Your money: $"<<user.money*10<<".00\n"
        "Your score: "<<user.score<<" points\n\n"
        "Select an option below:\n"
        " 1. Begin a new game of Wheel of Fortune\n"
        " 2. View the leaderboard\n"
        " 3. Append to the Library\n"
        " 4. View the Library(You'll spoil all the answers!)\n\n"
        "Any other input to exit: ";
}

void Game::addMoney(int n){
    //Process Data
    add(&user,n);
}

void Game::addScore(unsigned int n){
    //Process Data
    user.score+=n;
}

void Game::subMoney(int n){
    //Process Data
    sub(&user,n);
}
/* File:   Game.h
 * Author: Javier B
 * Created on December 5, 2016, 5:19 PM
 * Purpose: Class Implementation File for Game class
 */

//User Libraries
#include "Game.h"

Game::Game(){
    //Set Random seed
    srand(static_cast<unsigned int>(time(0)));

    //Fill Library
    counter=0;
    fill();

    //Input
    cout<<"Input your name: ";
    getline(cin,user.name);
    cout<<"Welcome to Wheel of Fortune "<<user.name<<"!\n";
}

Game::~Game(){
    //Close Files
    in.close();

    //Deallocate Memory
    delete[] arr;
    arr=nullptr;

    //Output Data
    char choice;
    cout<<"Thanks for playing "<<user.name<<"!"<<endl;
    cout<<"Your final score: "<<user.score<<" points"<<endl;
    cout<<"Do you wish to add your score to the leaderboard?\n"
            "Input 1 to add\n"
            "Input 2 to exit: ";
    cin>>choice;
    if(choice=='1'){
        addLder(); //Add to leaderboard
    }
    cout<<"Fun fact: You called class Letter ";
    if(staticCalls<0||staticCalls==32768) cout<<"0 times"<<endl;
    else cout<<staticCalls<<" times"<<endl;
}

void Game::addLder(){
    //Variables
    fstream out; //Output in binary
    int n;       //Size of string

    //Output Data
    out.open("users.dat",ios::out|ios::app|ios::binary);
    n=user.name.size();
    out.write(reinterpret_cast<char *>(&n),sizeof(n));
    out.write(user.name.c_str(),n);
    out.write(reinterpret_cast<char *>(&user.score),sizeof(unsigned int));
    cout<<"Your score has been added"<<endl;

    //Close files
    out.close();
}

void Game::fill(){
    //Variables
    char s[SIZE];     //Temp char array
    unsigned int a;   //Temp char

    //Open File
    in.open("phrase.dat",ios::in);
    if(in.fail()){
        cout<<"CRITICAL ERROR: File opening failed"<<endl;
        exit(1);
    }

    //Input Data
    while(in>>a){
        in.getline(s,SIZE,'.');
        counter++;    //Add to size of array
    }

    //Allocate Memory
    arr=new unsigned int[counter];
    in.clear();
    in.seekg(0L,ios::beg); //Go back to beginning of file

    //Process Data
    for(int i=0;i<counter;i++){
        arr[i]=in.tellg(); //Each index has a position
        in>>a;
        in.getline(s,SIZE,'.');
    }
}

void Game::lderBrd(){
    //Variables
    fstream in;         //Input from file
    int n;              //Size of string that is read from file
    vector<Player> user;//Array of Player structures
    Player temp;        //Temp Player to swap for
    int i=0;            //Size of array
    bool flag;
    string a;           //Player inputs to continue
    try{
        //Open files
        in.open("users.dat",ios::in|ios::binary);
        if(in.fail()){
            throw "users.dat not found";
        }
        //Input Data
        while(in.read(reinterpret_cast<char *>(&n),sizeof(n))){ //Get size of string
            temp.name.resize(n);     //Resize string size by size
            in.read(&temp.name[0],n);//In name and score
            in.read(reinterpret_cast<char *>(&temp.score),sizeof(unsigned int));
            user.push_back(temp);    //Push back array by one at a time
            i++;
        }

        //Process Data
        do{
            flag=false;;
            for(int j=0;(j<i-1);j++){
                if(user[j].score<user[j+1].score){ //Swap greatest to least
                    temp=user[j+1];
                    user[j+1]=user[j];
                    user[j]=temp;
                    flag=true;
                }
            }
        }while(flag==true);

        //Output Data
        cout<<"Sorted Leaderboard:"<<endl;
        for(int j=0;j<i;j++){
            cout<<user[j].name<<endl;
            cout<<setw(5)<<right<<user[j].score<<" points"<<endl<<endl;
        }
        cout<<"Press enter to continue";
        getline(cin,a);
    }
    catch(char* const s){
        in.close();
        cout<<s<<endl;
    }

    //Close files
    in.close();
}

void Game::read(){
    //Variables
    Clue clue;      //Temporary Clue to fill
    unsigned int n; //Categories are numbered
    char s[SIZE];   //String to hold phrase
    //Open File
    in.clear();
    in.seekg(0L,ios::beg); //Go back to beginning of file

    //Input Data
    while(in>>n){          //Repeat until in can't extract a char
        in.getline(s,SIZE,'.');

    //Output Data
        clue.setCat(n); //Set category
        clue.showCat(); //View category
        cout<<s<<endl;  //Output string
    }

    cout<<"Input anything to continue: ";
    cin.get();
}

void Game::write(){
    //Variables
    char choice;    //Menu choice
    char input;     //Input for sub-menu
    fstream out;    //Output to file
    char line[SIZE];//Character array of size=44
    Clue clue;

    //Open File
    out.open("phrase.dat",ios::app);

    //Input Data
    cout<<endl<<"Input 1 to input a phrase\n"
            "Input 0 to exit: ";
    cin>>choice;
    cin.ignore();
    if(choice=='1'){
        cout<<endl<<"Input a category:\n";
        for(int i=1;i<=7;i++){
            cout<<i<<" ";
            clue.setCat(i);
            clue.showCat();
        }
        cout<<endl<<"0 Exit"<<endl;
        cin>>input;
        cin.ignore();

    //Output Data
        if(input>48&&input<56){ //If input is '1'-'7'
            clue.setCat(input-48);
            cout<<"Input your phrase(max 44 characters): "<<endl;
            cin.getline(line,SIZE);
            if(isGood(line)){ //If input is good ask if wish to append
                cin>>choice;
                cin.ignore();
                if(choice=='1'){
                    out<<input;
                    for(int i=0;i<strlen(line);i++){
                        out<<static_cast<char>(toupper(line[i])); //Make uppercase
                    }
                    out<<"."<<endl;
                    cout<<"You must restart the game for effects to take effect"<<endl;
                }
            }
        }
    }

    //Close File
    out.close();
}

bool Game::isGood(char a[]){
    //Process Data
    try{
        if(strlen(a)<4||strlen(a)>44){ //If char array doesn't fit size limit
            throw "ERROR: Phrase must be greater than 3 characters and less than 44";
        }
        for(int i=0;i<strlen(a);i++){
            if(isdigit(a[i])||(!isalpha(a[i])&&!isspace(a[i]))){//If not space or letter
                throw "ERROR: Input must be characters only\n";
            }
        }

    //Output Data
        cout<<"Do you really wish to add the following phrase?"<<endl;
        for(int i=0;i<strlen(a);i++){
            cout<<static_cast<char>(toupper(a[i]));
        }
        cout<<endl<<endl<<"Input 1 to append\n"
                "Or anything else to cancel: ";
        return true;
    }

    //Catch errors
    catch(char const* s){
        cout<<s<<endl;
        return false;
    }
}

void Game::menu(){
    //Output Data
    cout<<"Your money: $"<<user.money*10<<".00\n"
        "Your score: "<<user.score<<" points\n\n"
        "Select an option below:\n"
        " 1. Begin a new game of Wheel of Fortune\n"
        " 2. View the leaderboard\n"
        " 3. Append to the Library\n"
        " 4. View the Library(You'll spoil all the answers!)\n\n"
        "Any other input to exit: ";
}

void Game::addMoney(int n){
    //Process Data
    add(&user,n);
}

void Game::addScore(unsigned int n){
    //Process Data
    user.score+=n;
}

void Game::subMoney(int n){
    //Process Data
    sub(&user,n);
}
/* File:   Keyboard.h
 * Author: Javier B
 * Created on December 6, 2016, 12:12 PM
 * Purpose: Class Specification File for Keyboard class
 */

//User Libraries
#include "Keyboard.h"

Keyboard::Keyboard(){
    setArr();
}

Keyboard::~Keyboard(){
    //Deallocate Memory
    delete[] arr;
    arr=nullptr;
}

void Keyboard::display(){
    //Output Data
    cout<<endl<<"Your keyboard:"<<endl;
    for(int i=0;i<ALPHA;i++){         //Go through keyboard array
        if(arr[i].isLtUsed()==false){ //If letter has not been used, hide letter
            arr[i].display();
        }else cout<<"■";
        if((i+1)%13==0) cout<<endl;
    }
}

void Keyboard::setArr(){
    //Allocate Memory
    arr=new Letter[ALPHA]; //New Array of Letters for keyboard

    //Process Data
    for(int i=0;i<ALPHA;i++){   //Initialize with alphabet
        arr[i].setChar('A'+i);
    }
}
/* File:   Letter.h
 * Author: Javier B
 * Created on December 5, 2016, 6:08 PM
 * Purpose: Class Specification File for Letter class
 */
```

```
//User Libraries
#include "Letter.h"

//Initializing static variable
int Letter::calls=0;

Letter::Letter(char a){
    //Process Data
    letter=a;
    isUsed=false;
    calls++;
}

Letter::Letter(){
    //Process Data
    letter=' ';
    isUsed=false;
    calls++;
}
/* File:   Phrase.h
 * Author: Javier B
 * Created on December 6, 2016, 12:12 PM
 * Purpose: Class Implementation File for Phrase class
 */

//User Libraries
#include "Phrase.h"
#include "Letter.h"

Phrase::Phrase(){

}

Phrase::~Phrase(){
    //Deallocate Memory
    arr=nullptr;
}

void Phrase::setArr(unsigned int n, string s){
    //Allocate Memory
    Letter *ptr=new Letter[n]; //New array of Letters for phrase
    arr=ptr;
    //Input Data
    for(int i=0;i<n;i++){               //Initialize phrase array with clue
        arr[i].setChar(s[i]);
        if(isspace(arr[i].getLetter())){//If letter is space
            arr[i].use();              //Don't hide it
        }
    }
    size=n;
}

void Phrase::display(){
    //Output Data
    for(int i=0;i<size;i++){        //Go through clue array
```

```
        if(arr[i].isLtUsed()==false){  //If letter has not been used, hide letter
            cout<<"□";
        }else{
            arr[i].display();
        }
    }
    cout<<endl;
}

char Phrase::getLetter(int n){
    return arr[n].getLetter();
}

bool Phrase::getUsed(int n){
    return arr[n].isLtUsed();
}
/* File:   Play.h
 * Author: Javier B
 * Created on December 5, 2016, 8:56 PM
 * Purpose: Class Implementation File for play class
 */

//User Libraries
#include "Play.h"

void Play::play(Game *a){
    //Variables
    win=false;
    int index=(rand()%a->counter); //Index to choose clue
    unsigned int c; //Temp char
    char   s[SIZE]; //Temp string

    //Initialize wheel array
    int *w=new int[WHEEL];
    int statArr[WHEEL]={0,0,5,5,10,15,15,20,25,30,35,40}; //Initialize wheel
    for(int i=0;i<WHEEL;i++){
        w[i]=statArr[i];
    }
    this->w=w;

    //Input Data
    a->in.seekg(a->arr[index],ios::beg); //Go to position in file to get phrase and clue
    a->in.get();
    a->in>>c;
    a->in.getline(s,SIZE,'.');
    clue.setCat(c);
    clue.setPhrase(s);

    //Create a new Phrase
    Phrase p;
    p.setArr(strlen(s),s);

    //Copy Phrase to pointer
    this->p=p;

    //Set staticCalls to amount of static calls
    a->staticCalls=p.callUse();
}

void Play::end(Game *a){
    //Output Data
    if(a->getMoney()<=0){
        cout<<"The phrase was actually: "<<endl;
        for(int i=0;i<p.size;i++){
            cout<<clue.getPhrase(i);
        }cout<<endl;
        cout<<"You have no money.\n"
            "You must restart the game to play again"<<endl;
    }else cout<<"Congrats you win!\n"
            "You have $"<<a->getMoney()*10<<".00 left in your account"<<endl;
}

void Play::spin(Game *a){
    //Variables
    int choice=rand()%WHEEL;//Random wheel choice
    char input;             //Letter input
    bool error;             //Incorrect letter input
    bool match=false;       //Did letter match?
    int money=w[choice];    //Money to add or subtract from user's money
    int points=0;           //Counter for points
    bool win=true;

    //Input Data
    cout<<"Spinning...\nPress Enter to continue";
    cin.get();
    cout<<"_______________________________________________"<<endl;
    if(money==0) cout<<"You spun a free guess"<<endl;
    else cout<<endl<<"You spun $"<<money*10<<".00"<<endl;
    display();
    do{
        try{
            error=false;
            cout<<"What letter do you want to use? ";
            cin>>input;
            cin.ignore();

    //Process Data
            if(!isalpha(input)){
                throw "Input must be part of the alphabet";
            }
            input=static_cast<char>(toupper(input)); //Make uppercase
            if(input=='A'||input=='E'||input=='I'||input=='O'||input=='U'){
                throw "You have to buy vowels";
            }
            for(int i=0;i<ALPHA;i++){
                if(k.getChar(i)==input){
                    if(k.isUsed(i)==true){
                        cout<<"You already used that letter"<<endl;
                        return;
                    }else k.use(i);
                }
            }
        }
        catch(char const* s){
            cout<<s<<endl;
            error=true;
            cout<<"Press enter to continue"<<endl;
            cin.get();
        }
    }while(error); //Keep looping until valid input
    for(int i=0;i<p.size;i++){
        if(input==p.getLetter(i)){  //If letter matches
            points+=10;             //Add ten points
            match=true;             //Match is true
            p.use(i);               //Don't hide letter anymore
        }
    }
    //Output Data
    if(match){ //If match is true
        cout<<"You have been awarded $"<<money*10<<".00"<<endl;
        a->addMoney(money);
        cout<<"You gain 10 points for each letter guessed"<<endl;
        cout<<"You gained "<<points<<" points"<<endl;
        a->addScore(points);
        for(int i=0;i<p.size;i++){
            if(p.getUsed(i)==false){
                win=false;   //Not all letters are revealed, win=false;
            }
        }
        this->win=win;       //All letters of phrase are revealed, win=true
    }else{ //Match is not true
        a->subMoney(money);
        cout<<"____________________________________________"<<endl;
        cout<<"You have lost $"<<money*10<<".00."<<endl;
    }if(a->getMoney()<=0){
        end(a);
    }
    if(this->win==true){
        end(a);
    }
}

void Play::buy(Game *a){
    //Conditions to return
    if((k.isUsed(0))&&(k.isUsed(4))&&(k.isUsed(8))&&(k.isUsed(14))&&(k.isUsed(20))){
        cout<<"You have already bought all the vowels"<<endl;
        return; //Exit
    }
    if(a->getMoney()<=50){
        cout<<"You don't have enough money!"<<endl;
        cout<<"Spin the wheel or guess the puzzle"<<endl;
        cout<<"Input a key to continue: ";
        cin.get();
        return; //Exit
    }

    //Variables
    char input;     //Input for vowel
    bool error;     //Error
    bool win=true;  //Win

    //Input Data
    do{
        try{
            error=false;
            cout<<"__________________________________________"<<endl;
            display();
            cout<<"Which vowel do you want to buy? ";
            cin>>input;
            cin.ignore();
            input=static_cast<char>(toupper(input));
            if(input=='A'||input=='E'||input=='I'||input=='O'||input=='U'){

            }
            else{
                throw "You did not choose a vowel";
            }
            for(int i=0;i<ALPHA;i++){
                if(k.getChar(i)==input){
                    if(k.isUsed(i)==true){
                        throw "You already used that letter";
                    }else k.use(i);
                }
            }
        }
        catch(char const* s){
            cout<<s<<endl;
            error=true;
            cout<<"Press enter to continue"<<endl;
            cin.get();
        }
    }while(error==true); //Loop until valid input

    //Process Data
    cout<<"You have bought a vowel for $500.00"<<endl;
    a->subMoney(50);             //Subtract money from user
    for(int i=0;i<p.size;i++){
        if(p.getLetter(i)==input) //Reveal vowels from clue phrase
            p.use(i);
    }
    for(int i=0;i<p.size;i++){
        if(p.getUsed(i)==false){
            win=false;//Not all letters are revealed, win=false;
        }
    }
    if(win==true){    //All letters of phrase are revealed,
        this->win=win;//win=true
        end(a);
    }
}

void Play::guess(Game *a){
    //Variables
    string answer; //Player answer
    int counter=0; //Amount of empty letters in keyboard array
    int score=30;  //Points=score*counter
    bool win=true;

    //Input Data
    display();
    cout<<"Input the final answer: ";
    getline(cin,answer);

    //Process Data
    for(int i=0;i<p.size;i++){ //Convert to uppercase
        answer[i]=static_cast<char>(toupper(answer[i]));
    }
    for(int i=0;i<p.size;i++){
        if((p.getLetter(i))!=(answer[i])){//Check to see if all letters match
            win=false;                    //Phrase did not match answer
        }
    }
    cout<<endl;
    for(int i=0;i<p.size;i++){         //Go through phrase array to add
        if((p.getUsed(i))==false){     //points for each letter that is not used
            counter++;
        }
    }

    //Output Data
    if(win==true){
        score*=counter;
        cout<<"You gain 30 points for each hidden letter you guessed"<<endl;
        cout<<"You gain "<<score<<" points"<<endl;
        a->addScore(score);
        this->win=win;                 //Make private member win=local win;
        end(a);                        //Go to end
    }else{
        cout<<"You did not guess correctly. You have lost $300.00\n";
        a->subMoney(30);
    }
    if(a->getMoney()<=0){
        end(a);
    }
}
void Play::display(){
    //Output Data
    clue.showCat();
    p.display();
    k.display();
    cout<<endl;
}
void Play::menu(Game *a){
    //Output Data
    cout<<"Your money = $"<<a->getMoney()*10<<".00"<<endl;
    cout<<endl<<endl<<"What would you like to do?"<<endl;
    cout<<  " 1. Spin the Wheel ✪\n"
    " 2. Buy a vowel ($500.00)\n"
    " 3. Solve the Puzzle 🖂(Bad guess lose $300.00)\n"<<endl;
}
```