Who Wants to Be a Millionaire contain 2 class

****Class** Program**: This class contains the main logic for the game, including displaying questions, handling user input, and managing lifelines.

static List<Question> questions = new List<Question>()

{

new Question("What is the capital of the Philippines?", new string[]{"a. Manila", "b. Cebu", "c. Davao", "d. Quezon City"}, "a", 1000),

new Question("What is the national language of the Philippines?", new string[]{"a. Filipino", "b. English", "c. Tagalog", "d. Visayan"}, "a", 5000),

// ... more questions

};

This initializes a list of questions, each with the text, options, correct answer, and prize money.

****Class** Question**: This class represents a quiz question with its text, options, the correct answer, and the prize money associated with it.

class Question

{

public string QuestionText { get; }

public string[] Options { get; }

public string CorrectAnswer { get; }

public int Prize { get; }

public Question(string questionText, string[] options, string correctAnswer, int prize)

{

QuestionText = questionText;

Options = options;

CorrectAnswer = correctAnswer;

Prize = prize;

}

}

* **QuestionText**: The text of the question.
* **Options**: An array of strings containing the answer choices.
* **CorrectAnswer**: The correct answer option.
* **Prize**: The prize money for the correct answer.

****Game State Variables****:

static int currentQuestionIndex = 0;

static int prize = 0;

static bool isGameOver = false;

static bool fiftyFiftyUsed = false;

static bool phoneAFriendUsed = false;

static bool askTheAudienceUsed = false;

These variables keep track of the game's current state, such as the current question index, the prize amount, and whether each lifeline has been used.

****Main Method****

**static void Main(string[] args)**

**{**

**Console.WriteLine("Welcome to Who Wants to Be a Millionaire - Philippines Edition!");**

**while (!isGameOver)**

**{**

**DisplayQuestion();**

**string userAnswer = Console.ReadLine();**

**if (userAnswer.Trim().ToLower() == "l")**

**{**

**UseLifeline();**

**}**

**else if (userAnswer.ToLower() == "quit")**

**{**

**Console.WriteLine($"You earned ₱{prize:N0}. Thanks for playing!");**

**isGameOver = true;**

**}**

**else**

**{**

**CheckAnswer(userAnswer);**

**}**

**}**

**}**

This is the entry point of the program. It displays the welcome message and enters a loop where it keeps displaying questions until the game is over. The user can input answers or choose to use a lifeline.

****Display Question****:

static void DisplayQuestion()

{

Console.WriteLine($"Question {currentQuestionIndex + 1}: {questions[currentQuestionIndex].QuestionText}");

foreach (string option in questions[currentQuestionIndex].Options)

{

Console.WriteLine(option);

}

Console.Write("Your answer (Type L if need lifeline or type 'quit' to end the game): ");

}

This method displays the current question and its options to the user.

****Use Lifeline****

**static void UseLifeline()**

**{**

**List<int> availableLifelines = new List<int>();**

**Console.WriteLine("Available lifelines:");**

**if (!fiftyFiftyUsed)**

**{**

**Console.WriteLine("1. 50:50");**

**availableLifelines.Add(1);**

**}**

**if (!phoneAFriendUsed)**

**{**

**Console.WriteLine("2. Phone a Friend");**

**availableLifelines.Add(2);**

**}**

**if (!askTheAudienceUsed)**

**{**

**Console.WriteLine("3. Ask the Audience");**

**availableLifelines.Add(3);**

**}**

**if (availableLifelines.Count == 0)**

**{**

**Console.WriteLine("No lifelines available. Please answer the question without using a lifeline.");**

**return;**

**}**

**Console.Write("Enter lifeline number: ");**

**int lifelineChoice;**

**bool isValidChoice = int.TryParse(Console.ReadLine(), out lifelineChoice);**

**if (!isValidChoice || !availableLifelines.Contains(lifelineChoice))**

**{**

**Console.WriteLine("Invalid lifeline choice. Please try again.");**

**return;**

**}**

**switch (lifelineChoice)**

**{**

**case 1:**

**if (!fiftyFiftyUsed)**

**{**

**FiftyFifty();**

**fiftyFiftyUsed = true;**

**}**

**else**

**{**

**Console.WriteLine("50:50 lifeline already used.");**

**}**

**break;**

**case 2:**

**if (!phoneAFriendUsed)**

**{**

**PhoneAFriend();**

**phoneAFriendUsed = true;**

**}**

**else**

**{**

**Console.WriteLine("Phone a Friend lifeline already used.");**

**}**

**break;**

**case 3:**

**if (!askTheAudienceUsed)**

**{**

**AskTheAudience();**

**askTheAudienceUsed = true;**

**}**

**else**

**{**

**Console.WriteLine("Ask the Audience lifeline already used.");**

**}**

**break;**

**}**

**}**

This method handles the lifeline usage. It checks which lifelines are available and prompts the user to choose one. It then executes the corresponding lifeline method.

****Check Answer****:

static void CheckAnswer(string userAnswer)

{

if (userAnswer.Trim().ToLower() == questions[currentQuestionIndex].CorrectAnswer.ToLower())

{

Console.WriteLine("Correct!");

prize = questions[currentQuestionIndex].Prize;

currentQuestionIndex++;

if (currentQuestionIndex >= questions.Count)

{

Console.WriteLine("Congratulations! You've won ₱1,000,000!");

isGameOver = true;

}

else

{

Console.WriteLine($"Your prize: ₱{prize:N0}");

Console.WriteLine();

}

}

else

{

Console.WriteLine("Incorrect! Game Over.");

Console.WriteLine($"You earned ₱{prize:N0}. Thanks for playing!");

isGameOver = true;

}

}

****50:50 Lifeline****:

static void FiftyFifty()

{

List<string> options = new List<string>(questions[currentQuestionIndex].Options);

int correctIndex = options.FindIndex(option => option.EndsWith(questions[currentQuestionIndex].CorrectAnswer));

Random rnd = new Random();

int index1, index2;

do

{

index1 = rnd.Next(0, options.Count);

} while (index1 == correctIndex);

options.RemoveAt(index1);

do

{

index2 = rnd.Next(0, options.Count);

} while (index2 == correctIndex);

options.RemoveAt(index2);

Console.WriteLine("The remaining options are:");

foreach (string option in options)

{

Console.WriteLine(option);

}

}

This lifeline removes two incorrect options, leaving one correct and one incorrect option.

****Phone a Friend Lifeline****:

static void PhoneAFriend()

{

Console.WriteLine($"Your friend suggests the answer is: {questions[currentQuestionIndex].CorrectAnswer}");

}

This lifeline simulates calling a friend who always suggests the correct answer.

****Ask the Audience Lifeline****:

static void AskTheAudience()

{

Random rnd = new Random();

int correctIndex = Array.IndexOf(questions[currentQuestionIndex].Options, questions[currentQuestionIndex].CorrectAnswer);

int[] audienceVotes = new int[questions[currentQuestionIndex].Options.Length];

// Simulate audience votes

for (int i = 0; i < audienceVotes.Length; i++)

{

if (i == correctIndex)

{

// Give higher probability for the correct answer

audienceVotes[i] = rnd.Next(60, 101);

}

else

{

audienceVotes[i] = rnd.Next(1, 61);

}

}

Console.WriteLine("Audience votes:");

for (int i = 0; i < audienceVotes.Length; i++)

{

Console.WriteLine($"{questions[currentQuestionIndex].Options[i]}: {audienceVotes[i]}%");

}

}

This lifeline simulates asking the audience, with a higher probability that the audience suggests the correct answer.

***Who Wants to Be a Millionaire". It provides a series of questions, tracks the player's progress, and allows the use of lifelines to help answer difficult questions. The game loop continues until the player either answers incorrectly, wins by answering all questions, or chooses to quit. The lifelines add an interesting element by providing help when the player is unsure of the correct answer.***