**Unit 05 Develop: Eternal Quest Program**

***Step 4: Review the Program Specification***

Refer to the Eternal Quest program specification. As a team, review the program requirements and how it is supposed to work.

1. What does the program do?

#1 Creating 3 different types of goals. #2 List the goals

#3 Save the goals #4 Load the goals #5 Record event #6 Gives and keeps track of points for goals accomplished.

2. What user inputs does it have?

#1 Goals #2 Goal point value #3 Textfile name #4 Goal completions #5 Quit #6 Awards #7 Date goal was completed

3. What output does it produce?

#1 Levels, Points, Congratulations statements, Awards #2 List of goals #3 List of completed goals #4 Date goal is completed #5 Reminder to do goals

4. How does the program end.

#1 When the user enters “Quit” option

***Co-requirements for the program***

1. **What does the program do?**

\*\*1 displays on opening menu

\*\*2 provides goals for running a marathon, reading scriptures and visiting the temple.

\*\*3 displays opening message

\*\*4 asks the user to create new goals of any type.

\*\*5 starts the program and leads the user through it with prompts

\*\*6 asks the user to record an event.

\*\*7 diplays the user’s score.

\*\*8 displays closing message

1. **What user inputs does it have?**

\*\*1 user’s score.

\*\*2 choosing a goal

\*\*3 answers to prompts

\*\*4 recording events

1. **What output does it produce?**

\*\*1 menu

\*\*2 opening message

\*\*3 prompts

\*\*4 scores

\*\*5 closing message

1. **How does the program end?**

\*\*1. at the end of program, it tells the user the total score of the goals completed as for a closing message. Tells what goals they completed and how long they did it. returns to the main menu

\*\*2 end of program enter quit to end the program

***Step 5: Determine the classes***

\*\*1) **Menu class**: to display the menu

\*\*2) **Goal class**: base class for goals

\*\*3) **OneOffGoal class**: goal that gets checked off upon completion

\*\*4) **NeverEndingGoal class**: goal that continues forever

\*\*5) **XTimesGoal class**: goal that is to be completed X amount of times

6) **ReminderGoal:** type of goal that includes a reminder

***Steps 6: Define class behaviors , 7: Define class attributes &***

***8: Define Constructors***

#1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: **Menu (- means private while + means public)**

Attributes:

* \_programList : List<Activity>

Constructors:

+ Menu()

Behaviors:

+ DisplayMenu() : string

How it will work:

The Menu class will display the choices for the user for the program and record that choice as according to what the user chooses.

#2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: **Goal**

Attributes:

* \_name : string
* \_description : string
* \_startDate: DateTime
* \_completionDate : DateTime
* ­\_completion : bool
* \_possiblePoints : int
* \_earnedPoints : int
* \_failed : bool

Constructors:

+ Goal(string name, string description, int possiblePoints)

Uses the date created as the start date

+ Goal(DateTime \_startDate , string name, string description, int possiblePoints)

Allows user to create the start date

Behaviors:

+ CompleteGoal (string goal) : virtual void

Description: called when a goal is completed/ uses polymorphism – changes depending on goal type/ if you check it off, check it off, or is a never-ending goal

+ GetName() : string

+ SetName() : void

+ GetDescription() : string

+ SetDescription() : void

+ GetStartDate() : DateTime

+ SetStartDate() : void

+ GetCompletionDate() : DateTime

+ SetCompletionDate() : void

+ GetCompletion() : bool

+ SetCompletion() : void

+ GetPossiblePoints() : int

+ SetPossiblePoints() : void

+ GetEarnedPoints() : int

+ SetEarnedPoints() : void

+ GetFailed(): bool

+ SetFailed() : void

+ DisplayMessage(string message) : void

+DisplayPoints(): void

How it will work:

It provides the base class for all goal types

#3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: **OneOffGoal : Goal**

Attributes:

Constructors:

+ OneOffGoal()(string name, string description, int possiblePoints) : base(name, description, possiblePoints)

Uses the date created as the start date

+ OneOffGoal(DateTime \_startDate , string name, string description, int possiblePoints) : base(startDate, name, description, possiblePoints)

Allows user to create the start date

Behaviors:

+

How it will work:

It will inherit its attributes and methods from the goal base class and change the methods as needed with polymorphism

#4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: **NeverEndingGoal**

Attributes:

Constructors:

+ NeverEndingGoal()(string name, string description, int possiblePoints) : base(name, description, possiblePoints)

Uses the date created as the start date

+ NeverEndingGoal(DateTime \_startDate , string name, string description, int possiblePoints) : base(startDate, name, description, possiblePoints)

Allows user to create the start date

Behaviors:

+ CompleteGoal() : override int

add possiblePoints to the total earnedPoints variable

How it will work:

It will inherit its attributes and methods from the goal base class and change the methods as needed with polymorphism. Never truly completed instead it adds points as completion is continued.

#5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: **XTimesGoal**

Attributes:

* \_xTimes : int
* \_numTimesCompleted : int

Constructors:

+ XTimesGoal()(int \_xTimes, string name, string description, int possiblePoints) : base(name, description, possiblePoints)

Uses the date created as the start date

+ XTimesGoal(int \_xTimes, DateTime \_startDate , string name, string description, int possiblePoints) : base(startDate, name, description, possiblePoints)

Allows user to create the start date

Behaviors:

+ Iterate(int numTimesCompleted) : int

+ CompleteGoal() : override void

How it will work:

It will inherit its attributes and methods from the goal base class and change the methods as needed with polymorphism. Has an amount of times to be completed and displays completion as numTimesCompleted over xTimes.

#6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: **Scores**

Attributes:

* \_name : string
* \_points : int

Constructors:

+Score()

Behaviors(methods):

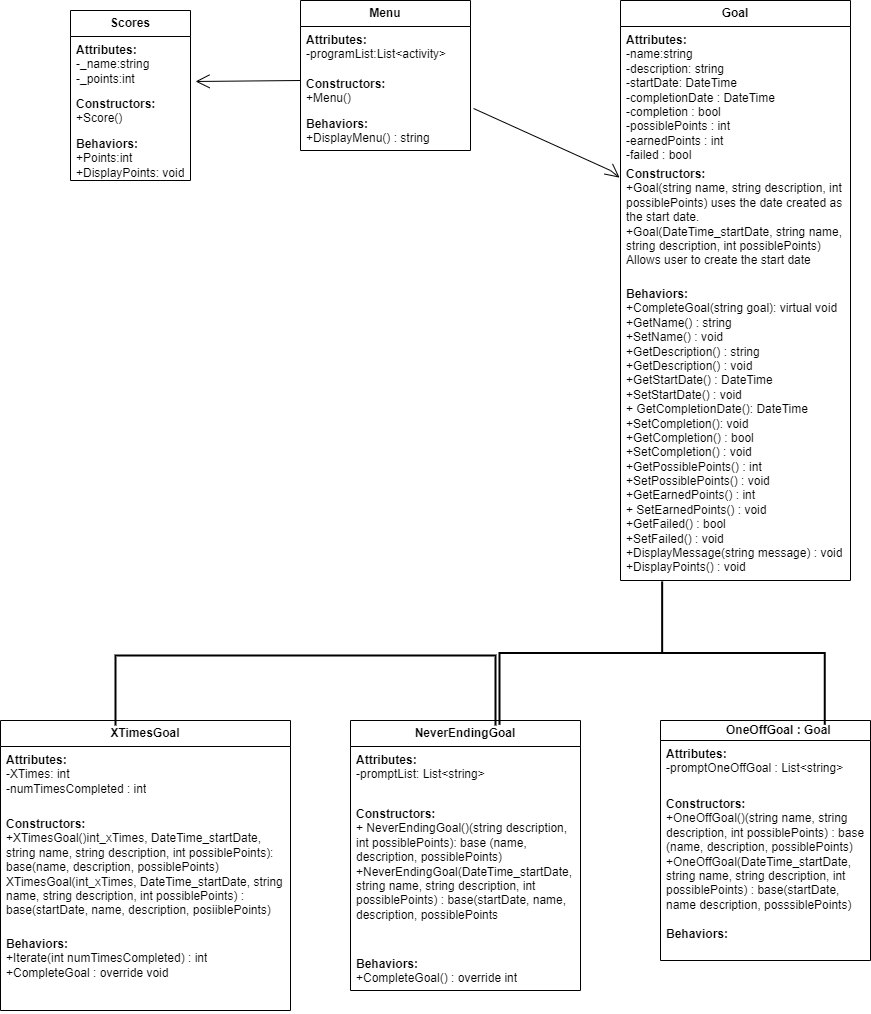
+ Points() : int

+ DisplayPoints() : void

How it will work:

Displays the user’s total scores in points.

Here is how my program will flow with this flow chart diagram below::



1. Create a c# eternal quest program that:

#1 Creates 3 different types of goals. #2 List the goals

#3 Save the goals #4 Load the goals #5 Record event #6 Gives and keeps track of points for goals accomplished.

2. What user inputs does it have?

#1 Goals #2 Goal point value #3 Textfile name #4 Goal completions #5 Quit #6 Awards #7 Date goal was completed

3. What output does it produce?

#1 Levels, Points, Congratulations statements, Awards #2 List of goals #3 List of completed goals #4 Date goal is completed #5 Reminder to do goals

4. How does the program end.

#1 When the user enters “Quit” option

using System;

using System.Collections.Generic;

namespace Classes

{

public class Menu

{

// Private attribute to store the list of activities

private List<Activity> \_programList;

// Public constructor that initializes the list

public Menu()

{

\_programList = new List<Activity>();

}

// Public behavior that displays the menu and returns the user's choice

public string DisplayMenu()

{

Console.WriteLine("Welcome to the program menu!");

Console.WriteLine("Please choose one of the following activities:");

// Loop through the list and display each activity

for (int i = 0; i < \_programList.Count; i++)

{

Console.WriteLine($"{i + 1}. {\_programList[i].Name}");

}

// Get the user's input and validate it

string choice = Console.ReadLine();

while (!IsValidChoice(choice))

{

Console.WriteLine("Invalid input. Please enter a valid number.");

choice = Console.ReadLine();

}

// Return the user's choice

return choice;

}

// Private helper method that checks if the user's input is a valid number within the list range

private bool IsValidChoice(string input)

{

int number;

if (int.TryParse(input, out number))

{

return number > 0 && number <= \_programList.Count;

}

return false;

}

}

// A simple class to represent an activity

public class Activity

{

// Public property to store the name of the activity

public string Name { get; set; }

// Public constructor that takes a name as a parameter

public Activity(string name)

{

Name = name;

}

}

}

using System;

namespace Classes

{

public class Goal

{

// Private attributes to store the goal properties

private string \_name;

private string \_description;

private DateTime \_startDate;

private DateTime \_completionDate;

private bool \_completion;

private int \_possiblePoints;

private int \_earnedPoints;

private bool \_failed;

// Public constructor that takes a name, a description, and possible points as parameters

// Uses the date created as the start date

public Goal(string name, string description, int possiblePoints)

{

\_name = name;

\_description = description;

\_possiblePoints = possiblePoints;

\_startDate = DateTime.Now; // Use the current date and time as the start date

\_completion = false; // Initialize the completion status to false

\_earnedPoints = 0; // Initialize the earned points to zero

\_failed = false; // Initialize the failed status to false

}

// Public constructor that takes a start date, a name, a description, and possible points as parameters

// Allows user to create the start date

public Goal(DateTime startDate, string name, string description, int possiblePoints)