Eternal Quest Program Design

Tracy Mann

CSE210

Bro. Parrish

Section A8

6/17/2023

v1.0

# What does the program do?

For this assignment you will write a program to help people follow through with their goals focusing on the following three types of goals.:

* Simple Goal - Goals that are one and done.
* Eternal Goal - Goals that repeat and are never finished, such as studying the scriptures.
* Checklist Goal - Goals have to be completed a number of times before they can be finished, such as attending the temple 10 times. While we recognize the value of our goals in the eternal sense, sometimes we need some shorter term "rewards" to help excite us along the way.

The program will keep track of various kinds of goals that people have, which are goals in their real life, such as studying the scriptures each day, finishing a project, or even stopping a bad habit. This program will keep track of your progress on these goals and offer points, awards, or other celebrations to keep you encouraged to keep working.

## Functional Requirements:

Your program must do the following:

1. Provide for simple goals that can be marked complete and the user gains some value.
   1. For example, if you run a marathon you get 1000 points.
2. Provide for eternal goals that are never complete, but each time the user records them, they gain some value.
   1. For example, every time you read your scriptures you get 100 points.
3. Provide for a checklist goal that must be accomplished a certain number of times to be complete. Each time the user records this goal they gain some value, but when they achieve the desired amount, they get an extra bonus.
   1. For example, if you set a goal to attend the temple 10 times, you might get 50 points each time you go, and then a bonus of 500 points on the 10th time.
4. Display the user's score.
5. Allow the user to create new goals of any type.
6. Allow the user to record an event (meaning they have accomplished a goal and should receive points).
7. Show a list of the goals. This list should show indicate whether the goal has been completed or not (for example **[ ]** compared to **[X]**), and for checklist goals it should show how many times the goal has been completed (for example **Completed 2/5 times**).
8. Allow the user's goals and their current score to be saved and loaded.

## Design Requirements

In addition, your program must:

1. Use inheritance by having a separate class for each kind of activity with a base class to contain any shared attributes or behaviors.
2. Use polymorphism by having derived classes override base class methods where appropriate.
3. Follow the principles of encapsulation and abstraction by having private member variables and putting related items in the same class.

## Stretch Requirements

1. I will save the file in JSON format while maintaining the encapsulation of the main objects to the outer users.
2. I will read configuration information for the messages to display from an XML config file.
3. I will allow a display of current goals that will not show the goals already completed.
4. I will allow a user to reuse a completed goal, leaving the original as completed and making an exact copy that is not completed for the original.

# What user inputs does it have?

1. Menu prompt response.
2. Select goal response.
3. Enter name response.
4. Enter description response.
5. Enter points response.
6. Enter number of times to complete response.
7. Enter bonus points response.

# What output does it produce?

1. Display menu (with each option display if available based on usage):
   * + 1. Add a simple goal.
       2. Add an eternal goal.
       3. Add a checklist goal.
       4. Reuse completed goal.
       5. List current goals.
       6. List all goals.
       7. Save goals.
       8. Load goals.
       9. Report Event.

Press Enter to quit!

>

1. Add Simple Goal Message: Add simple goal!
2. Add Eternal Goal Message: Add eternal goal!
3. Add Checklist Goal Message: Add checklist goal!
4. Reuse Completed Message: Reuse Completed Goal!
5. List Current Goals Message: List current goals!
6. List All Goals Message: List all goals!
7. Save Goals Message: Save goals.
8. Load Goals Message: Load goals.
9. Report Event Message: Report an event.
10. Score Message where {0} is the score: Your score is {0}.
11. Request Goal Message: Please enter the number of your goal.
12. Award Message where {0} is the points earned: Congratulations, you earned {0} points.
13. Request Name Message: Please enter the name of your goal.
14. Request Description Message: Please enter the description of your goal.
15. Request Point Value Message: Please enter the point value of your goal.
16. Request Repeat Point Value Message: Please enter the point value for each completion of your goal.
17. Request Checklist Complete Count Message: Please enter the number times required to complete your overall goal.
18. Request Checklist Bonus Point Value Message: Please enter the bonus point value of your overall goal.
19. Request File Name Message: Please enter the file name.
20. Simple and Eternal Goal Indexed Display where {0} is the index, {1} is the completed status, {2} is the name, and {3} is the description: {0}) [{1}] {2}({3})
21. Simple and Eternal Goal Non-Indexed Display where {0} is the completed status, {1} is the name, and {2} is the description: [{0}] {1}({2})
22. Checklist Goal Indexed Display where {0} is the index, {1} is the completed status, {2} is the name, {3} is the description, {4} is the number of time completed, and {5} is the number of times needed to complete this goal: {0}) [{1}] {2}({3}) {4}/{5}
23. Checklist Goal Non-Indexed Display where {0} is the completed status, {1} is the name, {2} is the description, {3} is the number of time completed, and {4} is the number of times needed to complete this goal: [{0}] {1}({2}) {3}/{4}

# How does the program end?

# The user will press enter to exit from the menu.

# Class Diagram:

# 

Notes: the classes that start with JSON in the name will all be internal classes because their descriptors are public in order to maintain encapsulation outside this namespace.

The only static methods I do not use all caps in the naming for are the casting operators used in conversions for JSON. This was maintained in camel case to retain consistent casting to built-in C# casting.

For that JSON serialization, I am following the same method used in my previous assignments where I wrote files out to JSON. <https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/how-to?pivots=dotnet-7-0>

In order to serialize the configuration for XML, I found the notes to do so at the follow site: <https://learn.microsoft.com/en-us/dotnet/api/system.xml.serialization.xmlserializer.deserialize?view=net-7.0>

For the ability to serialize a Dictionary object, allowing me to write the configuration out to XML I found the solution on the following site: <https://stackoverflow.com/questions/34316613/cannot-serialize-member-of-type-system-collections-generic-dictionary2-beca>

To clean up the XML and make it human readable I used details from the following site: <https://stackoverflow.com/questions/1123718/format-xml-string-to-print-friendly-xml-string>