Test Case Omega

Introduction

This is an advanced testcase set for proficient developers in any oops based language(c#, java or any alternates) . please select one of the testcase and publish your code in GITHUB. for more information, please refer the instructions for testcases.

> Please read the Publishing and evaluation details before sending us your response.

File Serialize Test Case (Test Case 01)

Question:

Create a C# project to do the following

- Read the given file using a file stream.
- Convert the file and serialize the data to a user-defined class object.
  - **Type cast the objects to datatype**s such as strings, floats and other standard datatypes used in C#.
  - Use of user defined datatypes such as new sub classes or structures during the serialization process is also allowed.
  - The main aim of this object is to be an serialized object. So, use of any methods (if any) for any other purpose other than reading and writing data from and to the object is **not allowed.**
- After the Object is serialized in a class object. Write the data into a file with a user defined File Type (not .txt or any existing file types but a new file extension).
- This file with a user defined file extension should be able to save in a dynamic path using a save file dialog box.
- The File should be created with the following in mind
  - The file should be stored in a format that could be easily serialized and de-serialized into a class object and physical file.

    - The file should not use an already serialized format such as .json or .xml.
- Open the created file and convert it into a serialized class object using a serialization technique.

Note:

Use a custom serializing and de-serializing technique for creating a class object or the new file type. this technique should be standard irrespective of change in datatypes.

Test Case 02 - Dll Creation and Access

This Testcase will be evaluated by the completion of 2 separate components. use the following information to implement the components.

DLL Project

- Create a '.dll' file with basic arithmetic functions such as
    - Add
    - Substact
    - Multiply
    - Divide
    - Power
- All the functions in the DLL should be returning the result of the operation and should not directly involve the console.

Output Class

Create an application that will be able to

- import the created .dll file to this project.
- create references to the .dll classes for accessing the operations inside them
- act as a basic calculator application with the features to input operands and view results of the operations.

Test Case 03 - GUI Data Control

- Define an Abstract Class and create a basic function.
- Create 2 child classes for the above class.
- Override the function of the parent class with different behaviours.
- Use the following question for implementing the above
- Design a form with following 3 buttons,
    1. New Valve
    2. New Pump
    3. Show All
- Clicking the above of the buttons would,
    1. New Valve - Opens a new dialog box/form/any alternative method for recording user's inputs for a new Valve object.
    2. New Pump - Opens a new dialog box/form/any alternative method for recording user's inputs for a new Pump object.
    3. Show All - Shows all objects created with its values in a dialog box.

Question :

Implement the Following setup

- Equipment : base abstract class with abstract method to show and write properties of the equipment.
- Valve: Derived class with custom properties which should be shown and updated using the abstract method from the base class.
- Pump: Derived class with custom properties which should be shown and updated using abstract method from base class.

Note: The show and write functions should be unique for both the derived classes(Valve and Pump).