TodoIt

Description:

Create a “Todo app” in Net. Core using knowledge you gained so far. It involves a concept of central storage and the relation between Todo and Person.

Recommended:

- 1-2 Students per group.
- Pair-programming teamwork.

Requirements:

- You follow instructions in the correct order. How you work is just as important as the result.
- Communication between group members.
- All group members have to contribute.
- Code comments – why and NOT how.
- Do not add excess methods or fields.

Objectives:

- Arrays
- Object oriented programming
- Encapsulation
- Relations between objects (Aggregation)
- Testing
- Agile workflow
- Collaboration

Instructions

1. Create a Core Console project
    a. Push empty project to GitHub.

2. Create a new folder called **Model**

3. Create **Person** class in folder Model.
    a. Required private fields are **personId** (int and readonly), **firstName** and **lastName** (String).
    b. Make a constructor that can build the object.
    c. Use Properties to prevent names from saving NULL & Empty
    d. Unit test **Person** class.
    e. Commit changes.

4. Create **Todo** class in the model folder
    a. Required private fields are **todoId** (int and readonly), **description** (String), **done** (bool) and **assignee** (Person).
    b. Make a constructor that take in **todoId** (int) and a **description** (String).
    c. Unit test Todo class.
    d. Commit changes.

5. Create a new folder called **Data**.

6. Create a new class **PersonSequencer** in Data folder.
    a. In PersonSequencer creates a private static int variable called personId.
    b. Add a static method called nextPersonId that increment and return the next personId value.
    c. Add a static method called reset() that sets the personId variable to 0.
    d. Unit test PersonSequencer.
    e. Commit changes.

7. Create a new class **TodoSequencer** in Data folder that have the same behaviour as PersonSequencer but different method names.
    a. Unit test TodoSequencer.
    b. Commit changes.

8. Create a new class called **People** inside the Data folder.
    a. Have a private static **Person array** declared and instantiated as empty and **not null** (new Person[0]).
    b. Add a method **public int Size()** that return the length of the array.
    c. Add a method **public Person[] FindAll()** that return the Person array.
    d. Add a method **public Person FindById(int personId)** that return the person that has a matching personId as the passed in parameter.
    e. Add a method that creates a new Person, adds the newly created object in the array and then return the created object. You have to "expand" the Person array. (tip: send in parameters needed to create the Person object and use the PersonSequencer to give a unique personId)
    f. Add a method **public void Clear()** that clears all Person objects from the Person array.
    g. Unit test People class.
    h. Commit changes.

9. Create a new class called **TodoItems** inside the Data folder.
    a. TodoItems should have the **same functionality as the People class**.
    b. Unit test TodoItems class
    c. Commit changes.

10. Add the following methods to **TodoItems** class
    a. **public Todo[] FindByDoneStatus(bool doneStatus)** – Returns array with objects that has a matching done status.
    b. **public Todo[] FindByAssignee(int personId)** – Returns array with objects that has an assignee with a personId matching.
    c. **public Todo[] FindByAssignee(Person assignee)** – Returns array with objects that has sent in Person.
    d. **public Todo[] FindUnassignedTodoItems()** – Returns an array of objects that does not have an assignee set.
    e. Unit test changes
    f. Commit.

11. Add the following to **TodoItems AND People** class.
    a. Functionality to **remove object from array**. (not nulling)
    First: you need to find the correct **array index of the object**.
    Second: You need to rebuild array by **excluding the object on found index**.
    b. Unit test changes
    c. Commit and Push to GitHub