Todo – Collaborative Assignment

Description:

Create a “Todo application” in Java using knowledge you gained so far.
It involves a concept of central storage and the relation between Todo and Person.
This assignment put your coding knowledge as much as your collaborative skills to a test.
In his project we do not create a console application, we test it with JUint instead.

Recommended:

- 1-3 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:

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

Teamwork suggestions:

1. Thoroughly read and talk through the assignment before starting to code.
2. Setup project structure together before implementing features.

Instructions

1. Create a maven project
    a. Choose the **"*maven-archetype-quickstart*"** as the archetype.
    b. Group id should be **`se.lexicon.[groupname]`**.
    c. Artifact id should be **`todo_it`**.
    d. When generated, navigate to **"`~/pom.xml`"**
    and make sure that the project uses ***java 8 +*** and ***JUnit 4.12 +***.
    e. Run your program and make sure it compiles.
    f. Push the empty project to GitHub.

2. Create a new package called **`model`**
3. Create **`Person`** class in package model.
    a. Required fields are **`personId`** (int and final), **`firstName`** and **`lastName`** (String).
    b. Make a constructor that can build the object.
    c. Create needed getters and setters.
    d. Unit test with Junit.
    e. Commit changes.

4. Create **`Todo`** class in the model package
    a. Required fields are **`todoId`** (int and final), **`description`** (String), **`done`** (boolean) and **`assignee`** (Person).
    b. Make a constructor that take in **`todoId`** (int) and a **`description`** (String).
    c. Create needed getters and setters.
    d. Unit test Todo class with Junit.
    e. Commit changes.

5. Create new package called **`data`**.
6. Create a new class **`PersonSequencer`** in data package.
    a. In PersonSequencer create 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 with Junit.
    e. Commit changes.

7. Create a new class **`TodoSequencer`** in data package that have the same behaviour as PersonSequencer but different method names.
    a. Unit test TodoSequencer with Junit.
    b. Commit changes.

8. Create a new class called **`People`** inside the data package.
    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 mathing personId as the passed in parameter.
    e. Add a method that creates a new Person, adds 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 you a unique personId)
    f. Add a method **`public void clear()`** that clears all Person objects from the Person array.
    g. Unit test People class with Junit.
    h. Commit changes.

9. Create a new class called **`TodoItems`** inside the data package.
    a. TodoItems should have the **same functionality as the People class but tailored for TodoItems.**
    b. Unit test TodoItems class.
    c. Commit changes.

10. Add the following methods to **`TodoItems`** class
    a. **`public Todo[] findByDoneStatus(boolean 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 & 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.