LabE

Scenario

Luna is an employee at Calgary Coding Solutions. She is a web developer. Luna has many tasks to do on daily basis. She needs a ToDoList Application to manage her daily tasks. Throughout the day, Luna needs to be able to add several tasks, mark some as complete, and delete others. However, at one point, she might accidentally delete an important task, as she is so busy.

The application needs to provide the undo feature, so Luna can revert this deletion, restoring the task to the list. Similarly, if she edits a task but decide against the changes, she can use the undo to revert to the task's original state.

Learning Objectives

- Apply OOP principles to design and implement the application.
- Use interfaces to define behavior for task management.
- Implement object copying to safely edit tasks.
- Implement an undo feature that utilizes object copying.
- Practice using data structures.
- Follow TDD practices.

Submission

- Upload your code before the end of the lab period to the associated dropbox folder. It is important to get the submission format correct in order to facilitate future labs. Only one person in the group needs to upload the code.
- The code should be submitted as #.zip, where # corresponds to your group number, e.g., group 12 should submit a file called 12.zip which contains the required files when unzipped.
- You should not include your student ID or name in group submissions.
- Your submission should contain the following files:
    - Task.java
    - IToDoList.java
    - ToDoList.java
- You should **not** include any other files, such as ToDoListTest.java

Instructions

- Download ToDoListTest.java
- Download the stub code for Task.java and use this as your starting point
- Write code to pass the tests, using the tips given below:

Tips

1. Task Class:

- The Task class should include properties for *id, title*, and *isCompleted*.
- You must implement a *copy* method for deep copying task objects.

2. IToDoList Interface:

- Define an interface with methods for adding, completing, deleting, editing, undoing, and listing tasks.

3. ToDoList Class:

- Implement the *IToDoList* interface.
- Implement an undo feature that allows users to revert the list to its previous state before the last change (add, complete, delete, edit).
- Use a *List<Task>* to manage tasks and a Stack<*List<Task>*> to track the history of operations (add, complete, delete, edit).
- For each operation, before making a change, push the current state or relevant information onto the stack.