COSC 3P91

Lab 4

In this lab you are going to use the design patterns **Command** and **Composite** in an example.

1. First, we consider three operations on mutable strings (class `StringBuilder`). These operations are append, insert, and replace. Use the **Command** design pattern creating commands for each of these operations. All classes are supposed to be a `StringCommand` (interface), and individual classes are called `StringCommandXXX`, e.g., `StringCommandAppend`. A `StringCommand` has an execute and an undo method. Both methods should have only the receiver of the command (`StringBuilder`) as a parameter.
2. Test your implementation by creating several commands, executing and undoing them on a given mutable string.
3. Implement a class called `StringManipulator`. This class has a `StringBuilder str` that can be manipulated. The class provides methods `append`, `insert`, and `replace` for that purpose. These methods create a corresponding `StringCommand` and execute it on `str`. Furthermore, the class provides two methods `undo()` and `redo()` for undoing resp. redoing the last/previous operation. Multiple calls of `undo()` and `redo()` will undo resp. redo the corresponding number of previous operations.
4. Now use the **Composite** design pattern for extending the `StringCommand`s to a macro language. The leaf constructions are the `StringCommand`s from above. Add two composite `StringCommand`s called `StringCommandSequence` and `StringCommandRepeat`. A `StringCommandSequence` has a list of `StringCommand`s that will be executed (and undone) at once. A `StringCommandRepeat` has a `numRepeat` (type `int`) and a `StringCommand body`. When executed the `body` is executed `numRepeat` times.
5. Extend the `StringManipulator` by a method `addMacro(StringCommand macro)` that will add `macro` to the manipulator. As above that means that `macro` is executed and can be undone and/or redone later.