COMP 304B – Object-Oriented Software Design
Assignment 4 – Command Pattern

Due date: Friday April 9, 2004 before 23:55

Practical information

- Team size == 2 (pair design) !
- Each team submits only *one full solution*. Use the `index.html` template provided on the assignments page. Use **exactly** this format to specify names and IDs of the team members. The other team member *must* submit a single `index.html` file containing only the coordinates of both team members. This will allow us to put in grades for both team members in WebCT. Beware: after the submission deadline there is no way of adding the other team member's `index.html` file and thus no way of entering a grade !
- Your submission must be in the form of a simple HTML file (`index.html`) with explicit references to *all submitted files* as well as inline inclusion of images. See the general assignments page for an `index.html` template.
- The submission medium is WebCT.

The assignment

In this assignment you will use UML *Class Diagrams*, Pseudocode and UML *Sequence Diagrams* to describe a simple design based on the *Command Pattern*. The design uses a highly simplified version of part of the DSheet design.

A `DSheet` has references

- named `subject` to a single `SSheetData` object;
- named `cmds` to 0 or more `Command` objects.

`SSheetData` has a private attribute `cellData`, a dictionary mapping a given coordinate (a tuple `(col,row)`) to a SSheetCell object . Its first public method is `getCellValue(col,row)`, which returns the evaluated value of the formula stored at coordinate `(col,row)`. The method `setCell(col,row, formulaStr)` instantiates and adds to the dictionary a SSheetCell object having formula value `formulaStr` and coordinates `(col,row)`. A method `removeCell(col,row)` is also available to remove a given cell (for instance if its formula value becomes empty). It also has a constructor which initializes the datastructures.

`Command` is an *abstract* class. Its constructor (used in concrete subclasses) takes a reference to `DSheet`'s `SSheetData` object as an argument and uses that to provide a local reference named `ref`. `Command` declares (and provides dummy, empty implementations) for methods `execute()` and `unexecute()`.

`Command` has two concrete subclasses, `DisplayCommand` and `SetCommand`.

`DisplayCommand`'s `execute()` method uses the `ref` reference to loop over all cells and print their values to the screen. `DisplayCommand` does not override the default, dummy `unexecute()` method in `Command` (as `DisplayCommand`'s `execute()` method does not modify the state of the `subject`).

`SetCommand` has an `askUser()` method which prompts the user for row, column, and formula to set. It returns the tuple `(row, column, formula)`. `SetCommand`'s `execute()` method calls `askUser()` and subsequently uses the `ref` reference to set the appropriate cell to the value given by the user.

`SetCommand` has a `memory` attribute as it is necessary to *remember the effects* of `execute()`. Using this `memory` information, `unexecute()` will be able to undo previous `execute()`s. Note how you must support an *arbitrary* number of undo levels.

Your assignment solution should contain:

1. A Class Diagram depicting all relevant classes, their attributes and methods, as well as all relationships between the classes.

   Pseudocode (in sticky notes) must be provided for *all* important methods (in particular, for `execute()` and `unexecute()`).

2. Pseudocode for a *Use Case* in which

   (a) a `DSheet` object gets created;

   (b) this also creates a `SSheetData` object;

   (c) two `Command` objects get created, one `DisplayCommand` and one `SetCellCommand` object. They get stored in the first and second position of the `DSheet`'s `cmds` list;

   (d) the `execute()` method of the `DSheet`'s first command (in its `cmds` list) gets called;

   (e) the `execute()` method of the `DSheet`'s second command (in its `cmds` list) gets called. Assume the user decides to set the cell at row 5 and column 5 to 5;

   (f) the `execute()` method of the `DSheet`'s first command (in its `cmds` list) gets called;

   (g) the `execute()` method of the `DSheet`'s second command (in its `cmds` list) gets called. Assume the user decides to set the cell at row 5 and column 5 to 10;

   (h) the `execute()` method of the `DSheet`'s second command (in its `cmds` list) gets called. Assume the user decides to set the cell at row 1 and column 1 to 1;

   (i) the `unexecute()` method of the `DSheet`'s second command (in its `cmds` list) gets called.

   (j) the `execute()` method of the `DSheet`'s first command (in its `cmds` list) gets called;

   (k) the `unexecute()` method of the `DSheet`'s second command (in its `cmds` list) gets called.

   (l) the `execute()` method of the `DSheet`'s first command (in its `cmds` list) gets called;

   For each operation, describe the new state of the relevant objects.

3. A Sequence Diagram depicting the above Use Case.

The Class Diagram, Pseudocode, and Sequence Diagram must be *consistent*.

Add short explanations where necessary.