Exercise 7

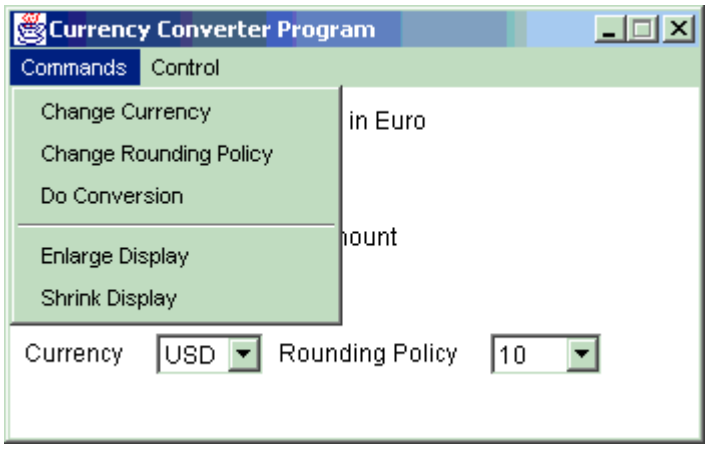
29.11.2001 **Due date: 12.11.2002**

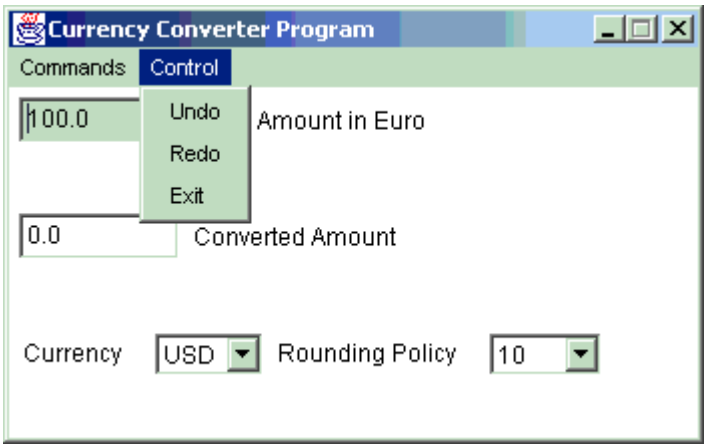
Assignment 7.1.

This assignment introduces the use of the command pattern from Gamma?s book on an extended version of the currency converter program (CCP) discussed in previous exercises.
Object of the exercise is to implement a CCP application with the following functionalities:

- conversion from one of the following currencies: Euro to
  - US Dollar,
  - Japanese Yen, and
  - Russian Rubel
- use of one of the following rounding policies:
  - RoundingPolicyDefault,
  - RoundingPolicy10 and
  - RoundingPolicy100
- undo and redo capabilities for up to 10 user commands

The user interface for the CCP application should be as in the figure:





User commands are entered through a pull-down menu. Foreseen commands are:

- Change Currency: use input currency indicated by the string in the ?currency? input field.
- Change Rounding Policy: take the rounding policy indicated by the string in the ?rounding policy? input field.
- Do Conversion: convert the amount in the Euro input field to the selected currency.
- Enlarge Display: increase the size of the window containing the CCP GUI by 10%.
- Shrink Display: decrease the size of the window containing the CCP GUI by 10%.
- Exit: terminate the CCP application

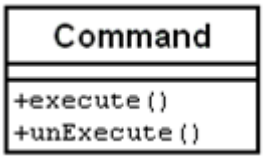
The command design pattern proposes the encapsulation of a request or an action in an object. The command pattern allows commands to be treated in a uniform manner independent of their content.

- Command histories can be easily set up and maintained.
- Undo operations are easily implemented.

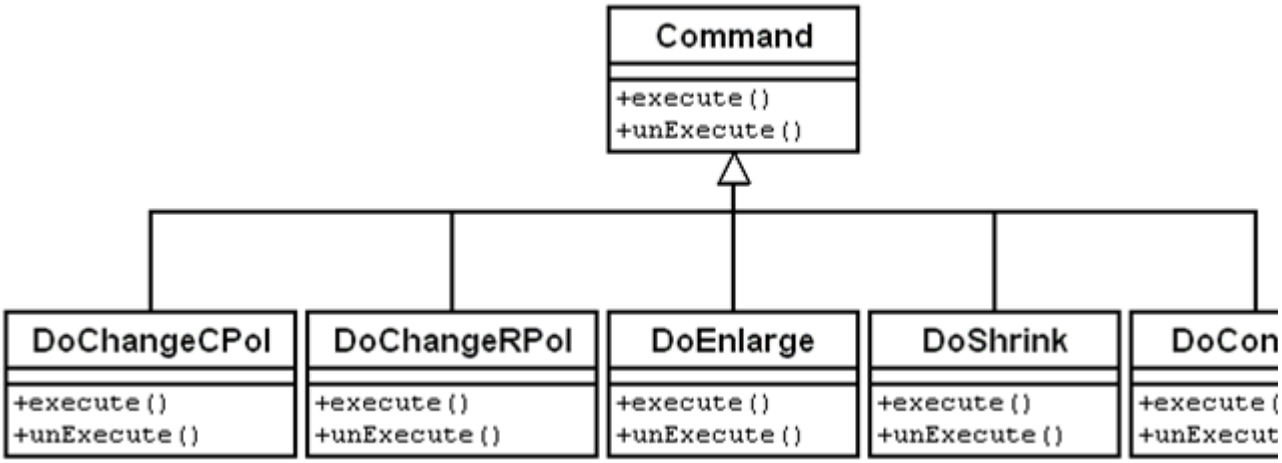
The command pattern allows management of commands to be separated from their implementation.

The abstract class Command:

- A call to execute causes the command action to be implemented,
- a call to unExecute brings the receiver of the command to its state prior to the execution of the command.
- Concrete commands are instantiated from subclasses of Command

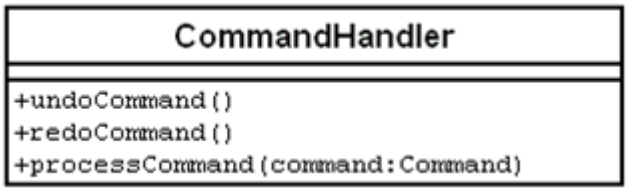


Concrete command subclasses for this exercise could be:



The Command Handler performs the following tasks:

- Execute commands it receives from the user interface
- Keeps a list of the 10 most recently executed commands (for undo)
- Executes undo and redo requests using the commands in the list
- The command list could be implemented as a circular buffer:



- Method processCommand calls execute on its argument and stores in the command list
- Methods undoCommand and redoCommand navigate the command list back and forth each time executing or un-executing a command

Thanks to the use of the command design pattern, the command handler is independent of the content of the commands and could be used in context other than the this application