

Design Patterns

MSc in Communications Software

Produced
by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



Undo / Redo Discussion

Issues

- How do we know if command is 'undoable' or not?
- Does our solution in fact work at all? (as currently implemented)

'undoable'

```
public boolean dispatchCommand(String commandName, Object [] parameters)
{
    boolean dispatched = false;
    Command command = commands.get(commandName);

    if (command != null)
    {
        dispatched = true;
        command.doCommand(parameters);
        undoBuffer.push(command);
    }
    return dispatched;
}
```

- How do we know if command is 'undoable' or not?
- Eg. if command is 'lu' what happens when next command is 'undo'?

One Solution

- Only push commands onto undo stack that make sense
- Is this solution feasible?

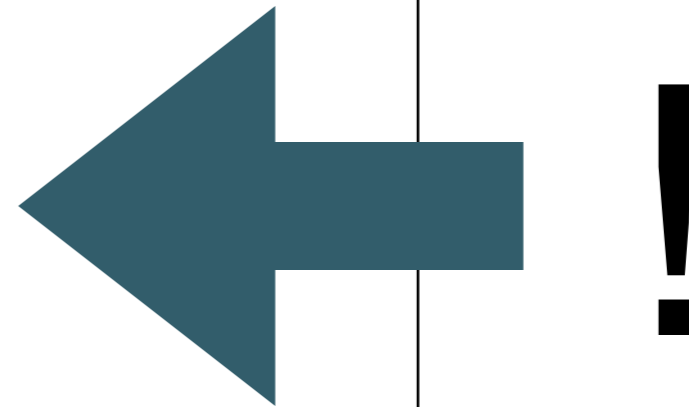
```
public boolean dispatchCommand(String commandName, Object [] parameters)
{
    boolean dispatched = false;
    Command command = commands.get(commandName);

    if (command != null)
    {
        dispatched = true;
        command.doCommand(parameters);
        if ((command instanceof CreateUserCommand)
            || (command instanceof DeleteUserCommand))
        {
            undoBuffer.push(command);
        }
    }
    return dispatched;
}
```

One Solution

```
public boolean dispatchCommand(String commandName, Object [] parameters)
{
    boolean dispatched = false;
    Command command = commands.get(commandName);

    if (command != null)
    {
        dispatched = true;
        command.doCommand(parameters);
        if ((command instanceof CreateUserCommand)
            || (command instanceof DeleteUserCommand)
            || (command instanceof UpdateUserCommand)
            || (command instanceof AddLocationCommand)
            || (command instanceof StoreCommand)
            || (command instanceof LoadCommand)
            || (command instanceof ChangeFormatCommand))
        {
            undoBuffer.push(command);
        }
    }
    return dispatched;
}
```



Sample Transcript

- add user a
- add user b
- add user c
- 1st undo - should remove user c
- 2nd undo - should remove user b
- Error?

```
Welcome to pacemaker-console - ?help for instructions
pm> cu a a a a
+-----+-----+-----+-----+
| ID | FIRSTNAME | LASTNAME | EMAIL | PASSWORD |
+-----+-----+-----+-----+
| 1 |          a |          a |      a |          a |
+-----+-----+-----+-----+

pm> cu b b b b
+-----+-----+-----+-----+
| ID | FIRSTNAME | LASTNAME | EMAIL | PASSWORD |
+-----+-----+-----+-----+
| 2 |          b |          b |      b |          b |
+-----+-----+-----+-----+

pm> cu c c c c
+-----+-----+-----+-----+
| ID | FIRSTNAME | LASTNAME | EMAIL | PASSWORD |
+-----+-----+-----+-----+
| 3 |          c |          c |      c |          c |
+-----+-----+-----+-----+

pm> undo
pm> undo
Error executing command
pm>
```

Debug

▼ ● this	CommandDispatcher (id=25)
▶ ■ commands	HashMap<K,V> (id=42)
▶ ■ redoBuffer	Stack<E> (id=46)
▼ ■ undoBuffer	Stack<E> (id=49)
◆ capacityIncrement	0
◆ elementCount	3
▼ ◆ elementData	Object[10] (id=75)
▶ ▲ [0]	CreateUserCommand (id=31)
▶ ▲ [1]	CreateUserCommand (id=31)
▶ ▲ [2]	CreateUserCommand (id=31)
◆ modCount	3

- Breakpoint and inspect of undo stack after the third add command.
- Notice anything unusual?

Diagnosis

- Each contact in the AddCommand stored on the stack is in fact the same object.

undoBuffer	Stack<E> (id=49)
capacityIncrement	0
elementCount	3
elementData	Object[10] (id=75)
[0]	CreateUserCommand (id=31)
pacemaker	PacemakerAPI (id=36)
parser	AsciiParser (id=38)
user	User (id=76)
activities	HashMap<K,V> (id=77)
email	"c" (id=78)
firstname	"c" (id=79)
id	Long (id=80)
lastname	"c" (id=83)
password	"c" (id=84)
[1]	CreateUserCommand (id=31)
pacemaker	PacemakerAPI (id=36)
parser	AsciiParser (id=38)
user	User (id=76)
activities	HashMap<K,V> (id=77)
email	"c" (id=78)
firstname	"c" (id=79)
id	Long (id=80)
lastname	"c" (id=83)
password	"c" (id=84)
[2]	CreateUserCommand (id=31)
pacemaker	PacemakerAPI (id=36)
parser	AsciiParser (id=38)
user	User (id=76)
activities	HashMap<K,V> (id=77)
email	"c" (id=78)
firstname	"c" (id=79)
id	Long (id=80)
lastname	"c" (id=83)
password	"c" (id=84)
modCount	3

Resolution

- We should in fact take a copy of the Command object push the copy onto the undo stack.
- The original will be reused in subsequent command dispatches.
- How do we copy an object at runtime?

```
public boolean dispatchCommand(String commandName, Object [] parameters)
{
    boolean dispatched = false;
    Command command = commands.get(commandName);

    if (command != null)
    {
        dispatched = true;
        command.doCommand(parameters);
        if ((command instanceof CreateUserCommand)
            || (command instanceof DeleteUserCommand)
            || (command instanceof UpdateUserCommand)
            || (command instanceof AddLocationCommand)
            || (command instanceof StoreCommand)
            || (command instanceof LoadCommand)
            || (command instanceof ChangeFormatCommand))
        {
            undoBuffer.push(command);
        }
    }
    return dispatched;
}
```

Copy a Command?

- use instanceof to determine type
- construct and push appropriate object.

```
...
{
  if (command instanceof CreateUserCommand)
  {
    Command newcommand = new new CreateUserCommand(paceApi, parser, command.user)
    undoBuffer.push(newcommand);
  }
  else if (command instanceof DeleteUserCommand)
  {
    Command newcommand = new new DeleteUserCommand(paceApi, parser, command.user)
    undoBuffer.push(newcommand);
  }
  else ....
  ....
}
...
```

Rethink

```
public PacemakerShell()
{
    Parser parser = new AsciiParser();
    paceApi      = new PacemakerAPI();
    dispatcher   = new CommandDispatcher();
    dispatcher.addCommand("list-users", new ListUsersCommand(paceApi, parser));
    dispatcher.addCommand("create-user", new CreateUserCommand(paceApi, parser));
    dispatcher.addCommand("delete-user", new DeleteUserCommand(paceApi, parser));
}
```

- Core design needs a rethink.
- Command objects are created in CommandDispatcher
- Consider these to be “Prototypes” of Command objects.
- We generate new commands from these prototypes in some manner.



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE

