الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

مقرر تصميم نظم البرمجيات	كلية الهندسة المعلوماتية	المحاضرة 10

Design Patterns: Command Pattern

د. رياض سنبل

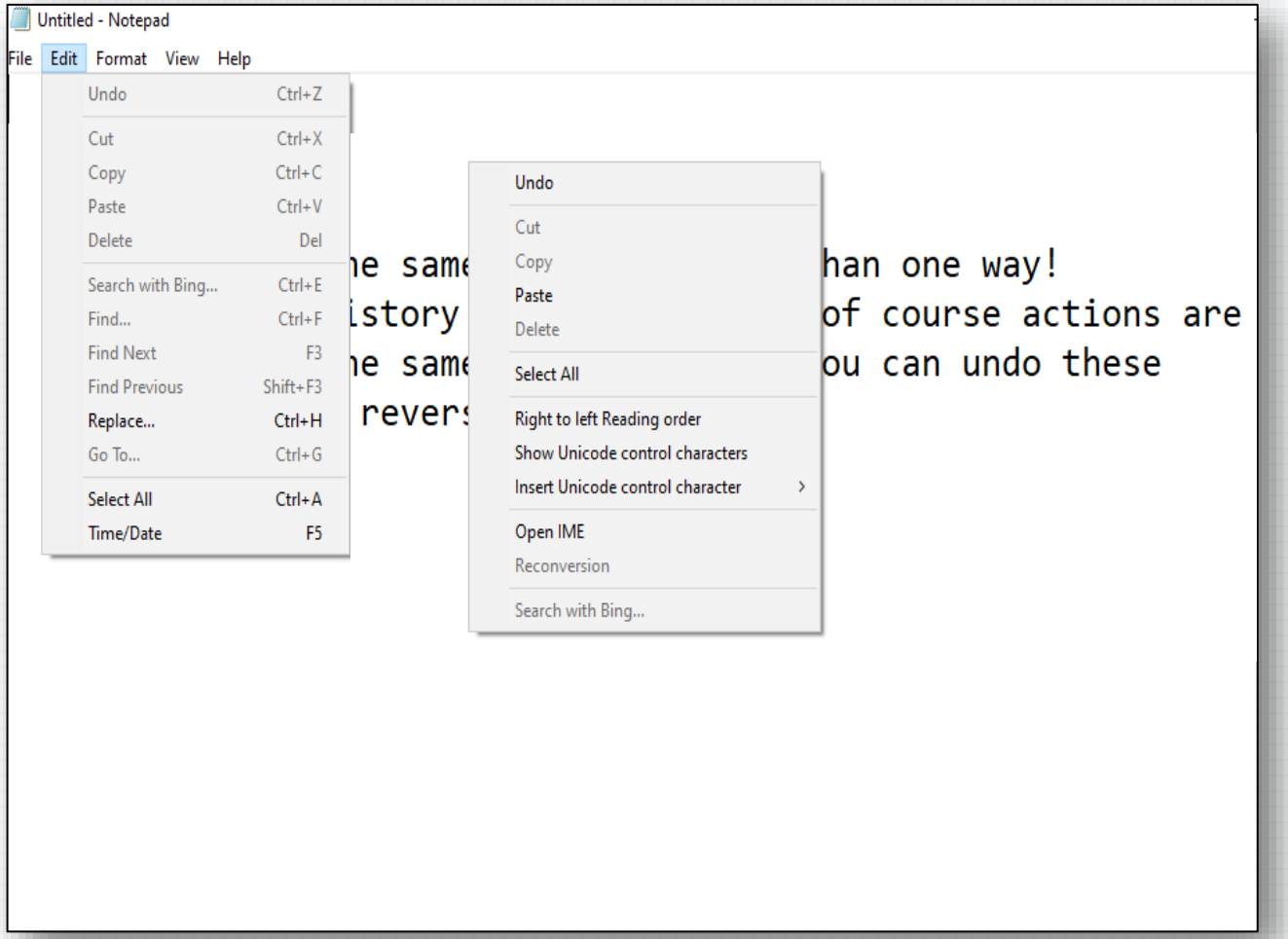
Behavioral Patterns

COMMAND PATTERN

The Problem

- In Text editors,
  - I can do the same action in more than one way!
  - I have a history for all actions (of course actions are not from the same type :) ,
  - also you can undo these actions in reverse order

- How can you implement these features!



Any Suggestion

The Solution

Encapsulate actions as objects, providing a way to parameterize clients with different requests, queue requests, and support undoable operations.

Use the Command Design Pattern

- *Decouples the invoker from the receiver of a request.*
- *Turns a request into a stand-alone object that contains all information about the request.*

The Receiver Class

```
public class TextFile {

    private String name;

    // constructor

    public String open() {
        return "Opening file " + name;
    }

    public String save() {
        return "Saving file " + name;
    }

    // additional text file methods (editing, writing, copying, pasting)
}
```

Command Classes

```java
@FunctionalInterface
public interface TextFileOperation {
    String execute();
}
```

```java
public class OpenTextFileOperation implements TextFileOperation {

    private TextFile textFile;

    // constructors

    @Override
    public String execute() {
        return textFile.open();
    }
}
```

```java
public class SaveTextFileOperation implements TextFileOperation {

    // same field and constructor as above

    @Override
    public String execute() {
        return textFile.save();
    }
}
```

The Invoker Class

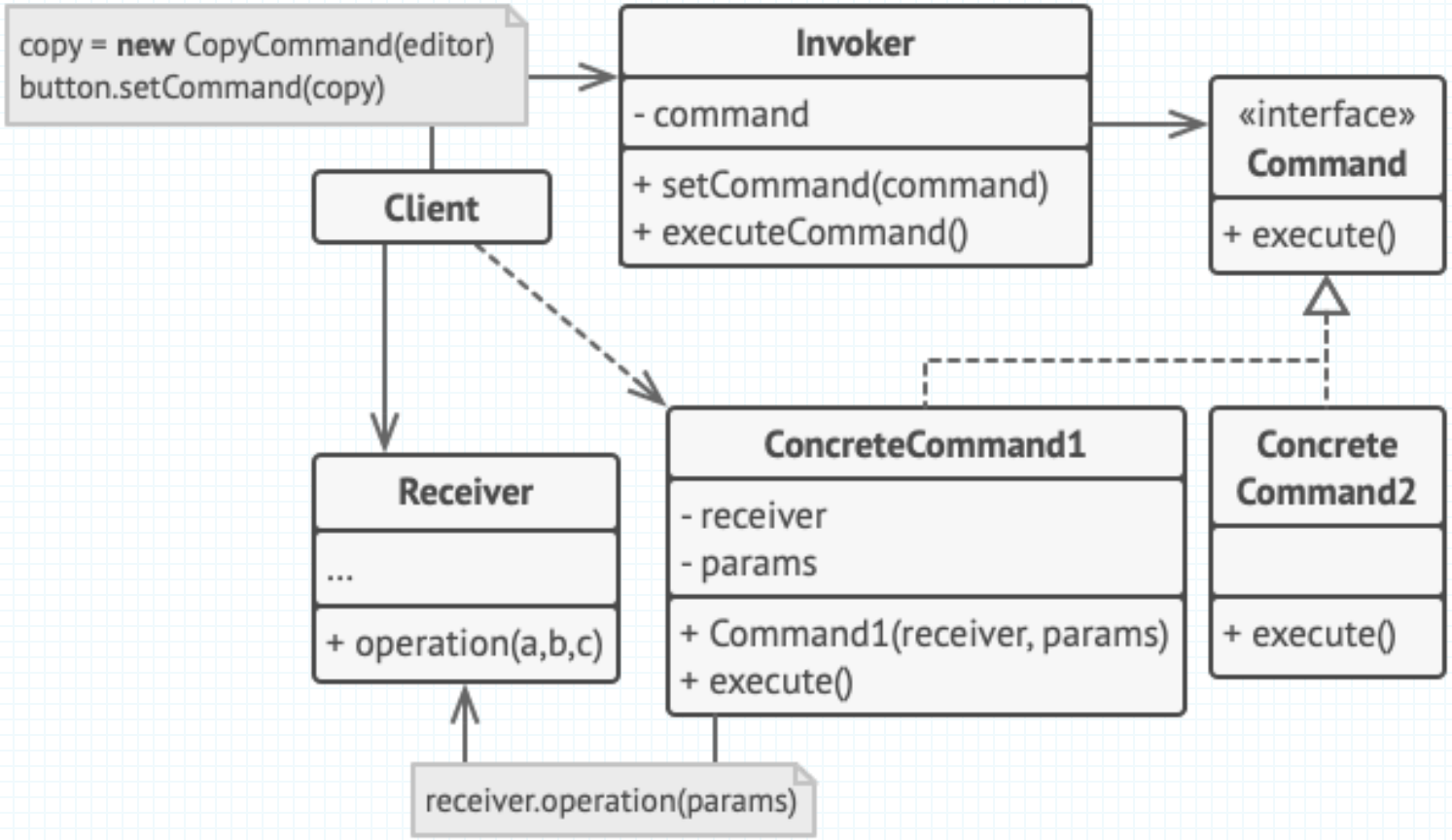
```
public class TextFileOperationExecutor {

    private final List<TextFileOperation> textFileOperations
     = new ArrayList<>();

    public String executeOperation(TextFileOperation textFileOperation) {
        textFileOperations.add(textFileOperation);
        return textFileOperation.execute();
    }
}
```

The Client Class

```
public static void main(String[] args) {
    TextFileOperationExecutor textFileOperationExecutor
      = new TextFileOperationExecutor();
    textFileOperationExecutor.executeOperation(
      new OpenTextFileOperation(new TextFile("file1.txt"))));
    textFileOperationExecutor.executeOperation(
      new SaveTextFileOperation(new TextFile("file2.txt"))));
}
```

Command Pattern

- The Command Design Pattern is a behavioral design pattern and helps to **decouples the invoker from the receiver of a request**.
- Command is a behavioral design pattern that **turns a request into a stand-alone object** that contains all information about the request.



Other Use cases for Command Pattern

- Transaction Management.
- Remote Control Systems.
- Networking and Message Handling (different types of images).
- Smart Home Automation Systems.
- etc