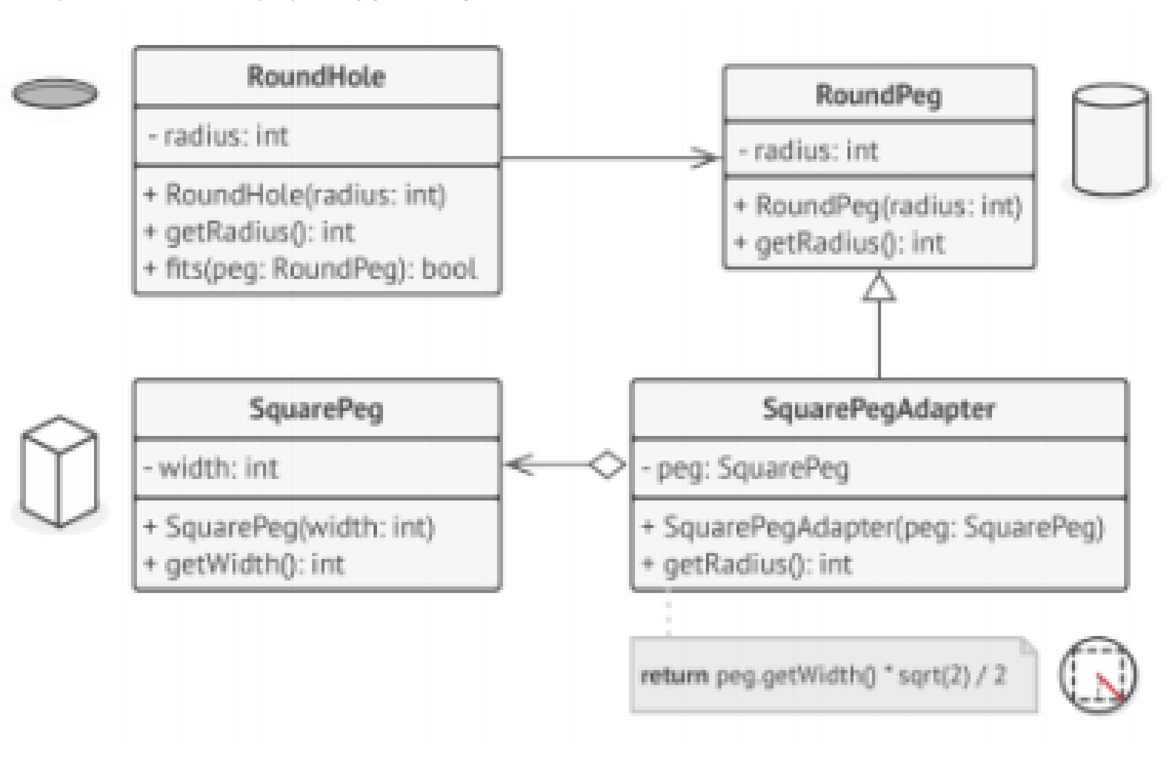
Paweł Kiełbasa, Wojciech Kosztyła

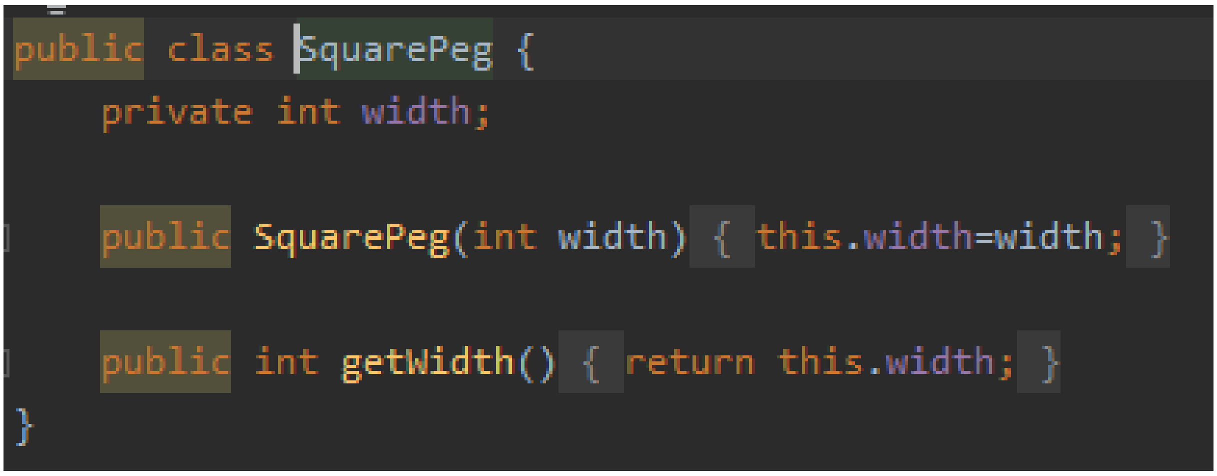
Sprawozdanie – Wzorce Projektowe 2

1. Zadanie 1
   a) Zaimplementowaliśmy aplikację według schematu:



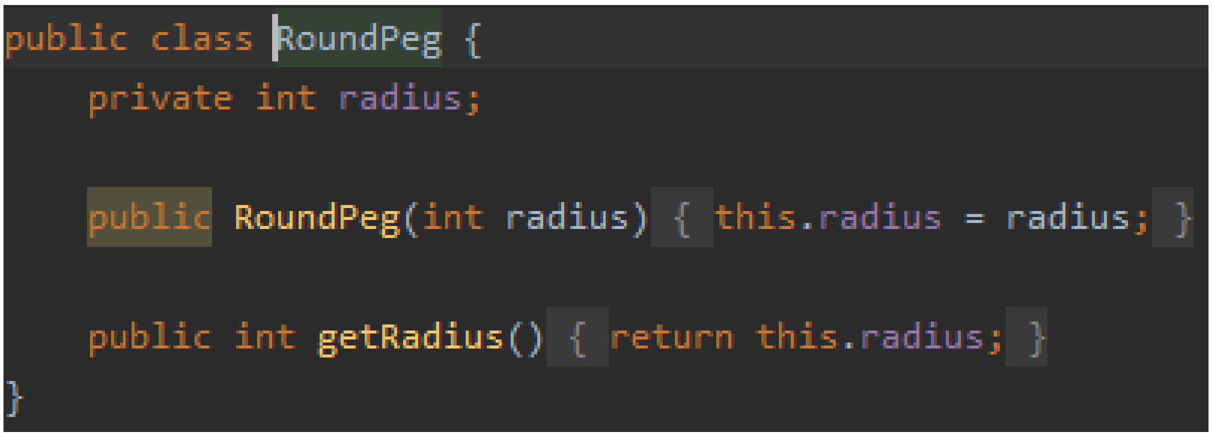
Rysunek 1: Adapting square pegs to round holes.

   b) Klasa SquarePeg



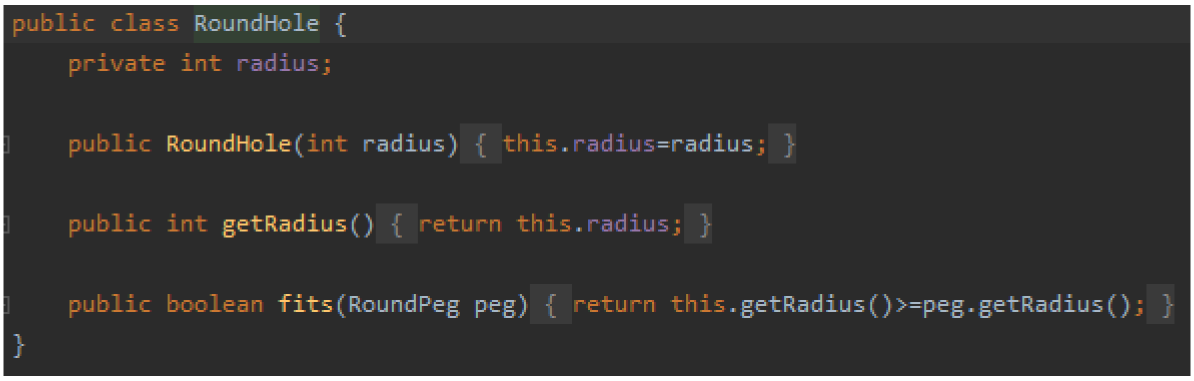
```
public class SquarePeg {
    private int width;

    public SquarePeg(int width) { this.width=width; }

    public int getWidth() { return this.width; }
}
```

   c) Klasa RoundPeg



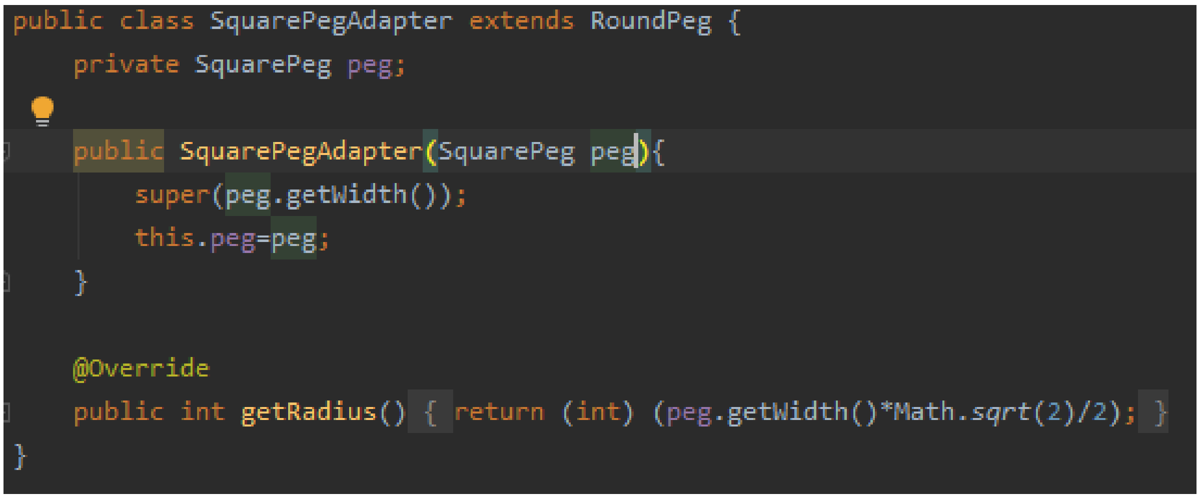
```
public class RoundPeg {
    private int radius;

    public RoundPeg(int radius) { this.radius = radius; }

    public int getRadius() { return this.radius; }
}
```

d) Klasa RoundHole



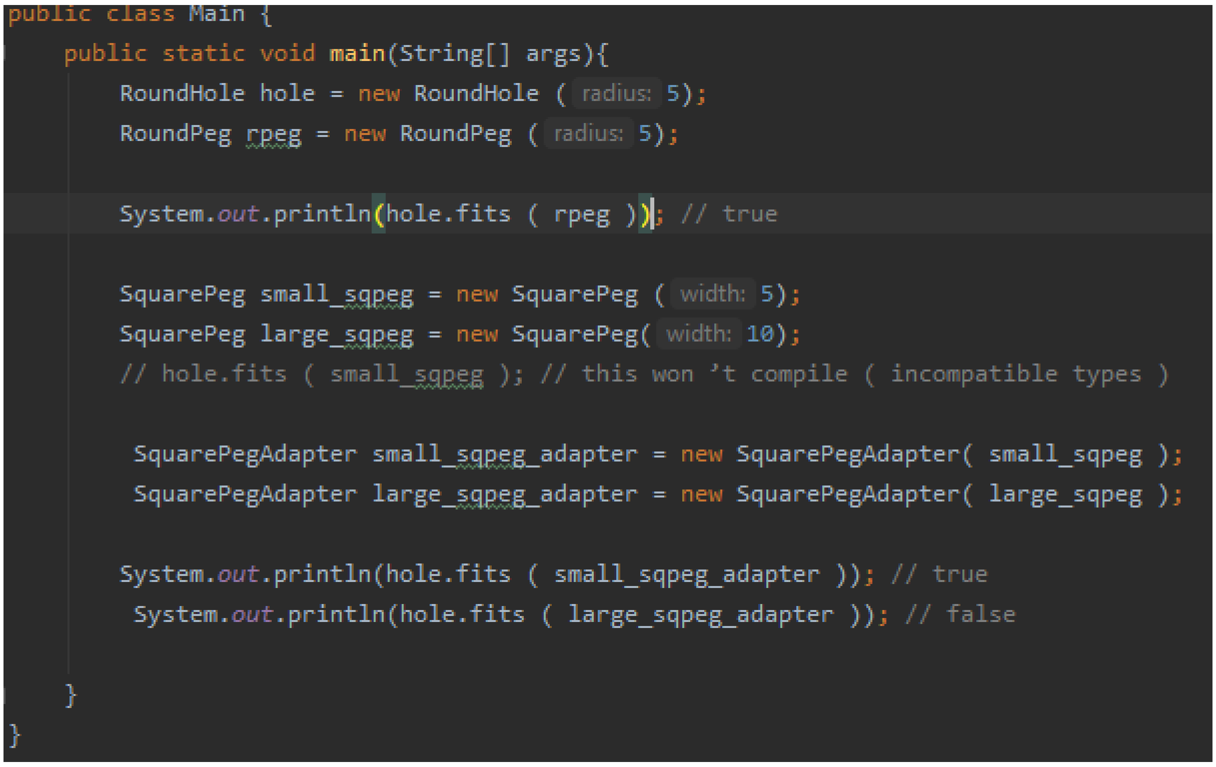
```java
public class RoundHole {
    private int radius;

    public RoundHole(int radius) { this.radius=radius; }

    public int getRadius() { return this.radius; }

    public boolean fits(RoundPeg peg) { return this.getRadius()>=peg.getRadius(); }
}
```

e) Klasa SquarePegAdapter



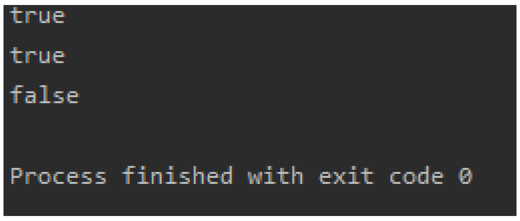
```java
public class SquarePegAdapter extends RoundPeg {
    private SquarePeg peg;

    public SquarePegAdapter(SquarePeg peg){
        super(peg.getWidth());
        this.peg=peg;
    }

    @Override
    public int getRadius() { return (int) (peg.getWidth()*Math.sqrt(2)/2); }
}
```

f) Klasa Main



```java
public class Main {
    public static void main(String[] args){
        RoundHole hole = new RoundHole ( radius: 5);
        RoundPeg rpeg = new RoundPeg ( radius: 5);

        System.out.println(hole.fits ( rpeg )); // true

        SquarePeg small_sqpeg = new SquarePeg ( width: 5);
        SquarePeg large_sqpeg = new SquarePeg( width: 10);
        // hole.fits ( small_sqpeg ); // this won ’t compile ( incompatible types )

         SquarePegAdapter small_sqpeg_adapter = new SquarePegAdapter( small_sqpeg );
         SquarePegAdapter large_sqpeg_adapter = new SquarePegAdapter( large_sqpeg );

        System.out.println(hole.fits ( small_sqpeg_adapter )); // true
         System.out.println(hole.fits ( large_sqpeg_adapter )); // false

    }
}
```

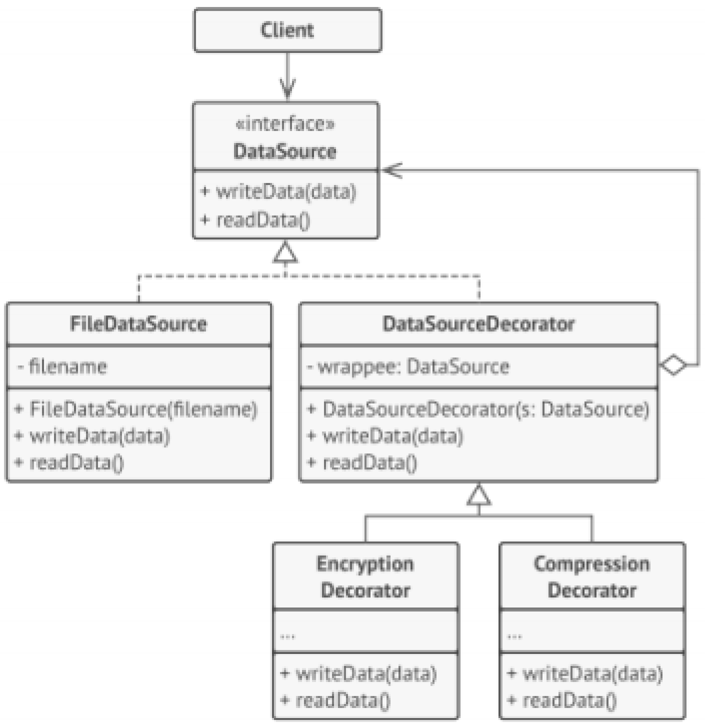
g) Efekt wykonania



```
true
true
false

Process finished with exit code 0
```

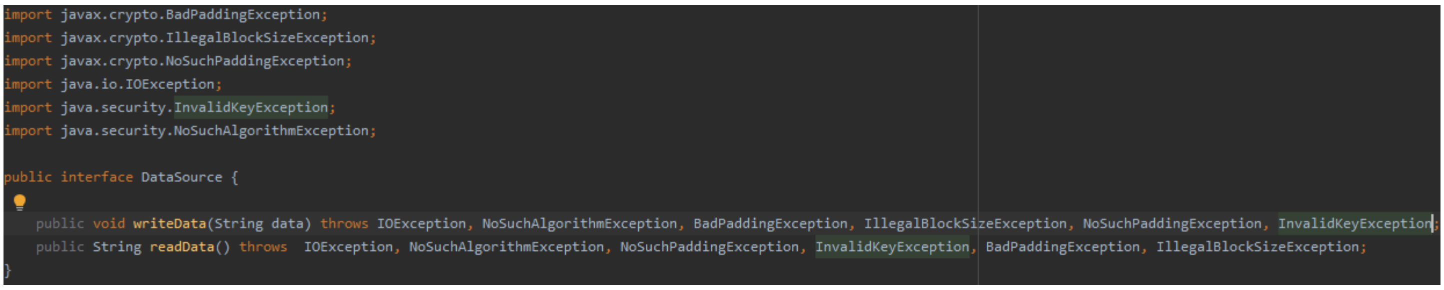
2. Zadanie 2

a) Zaimplementowaliśmy aplikację według poniższego schematu



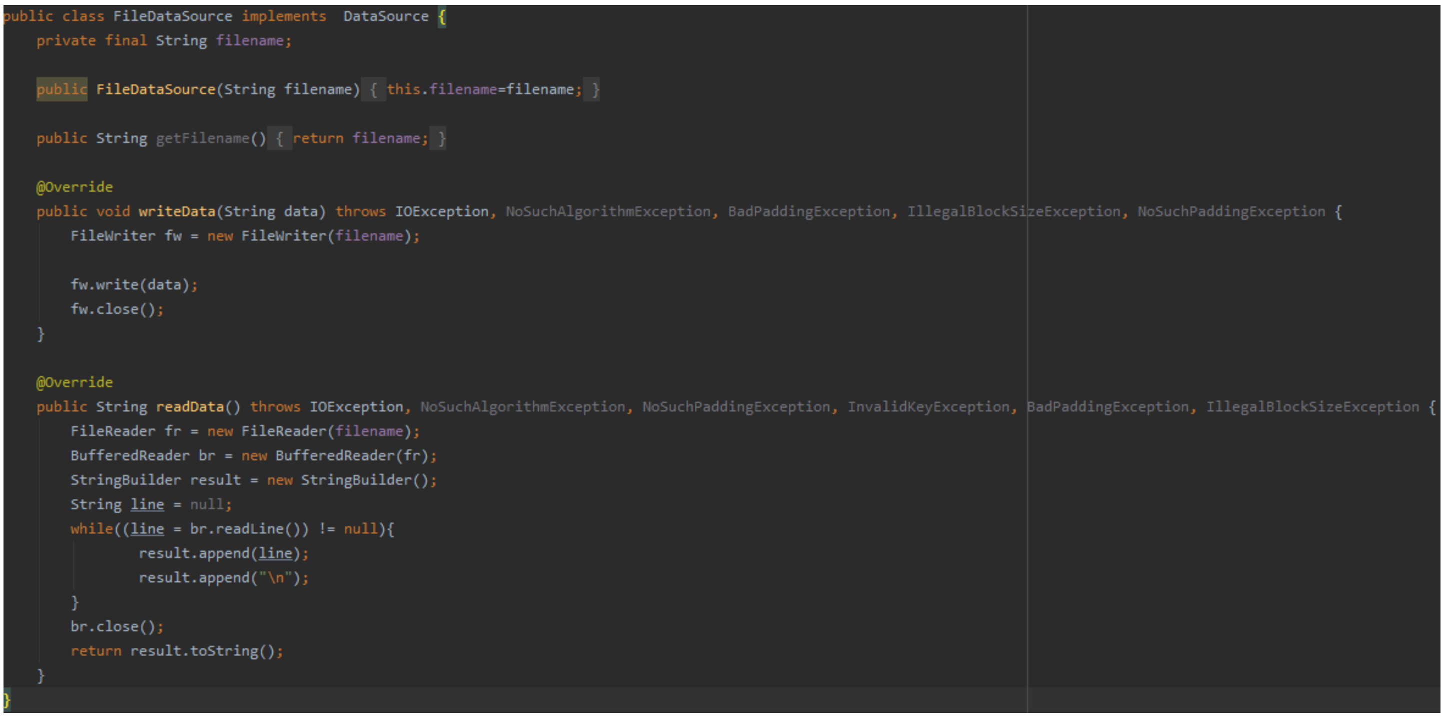
Rysunek 2: The encryption and compression decorators example.

b) Interfejs DataSource



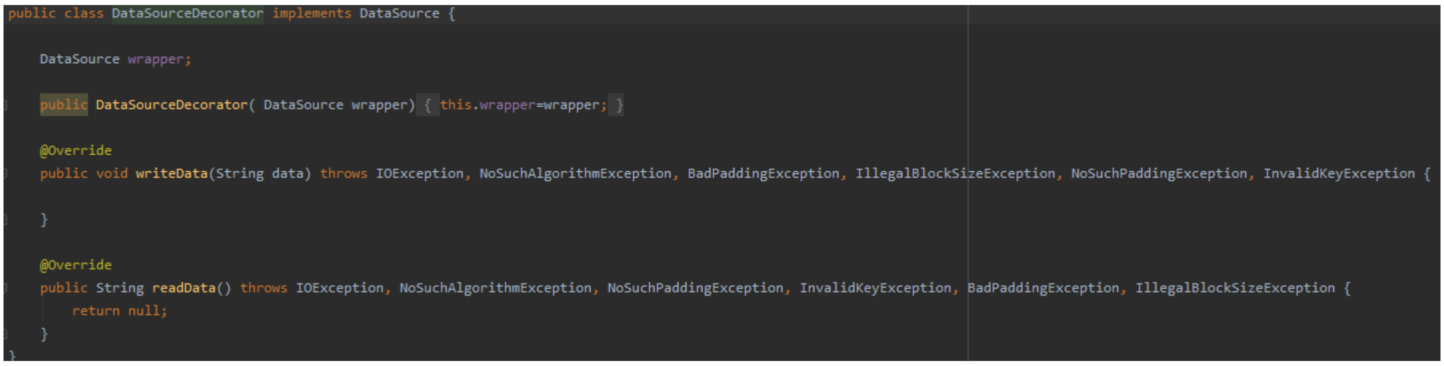
```java
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;

public interface DataSource {

    public void writeData(String data) throws IOException, NoSuchAlgorithmException, BadPaddingException, IllegalBlockSizeException, NoSuchPaddingException, InvalidKeyException;
    public String readData() throws  IOException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, BadPaddingException, IllegalBlockSizeException;
}
```

c) Klasa FileDataSource



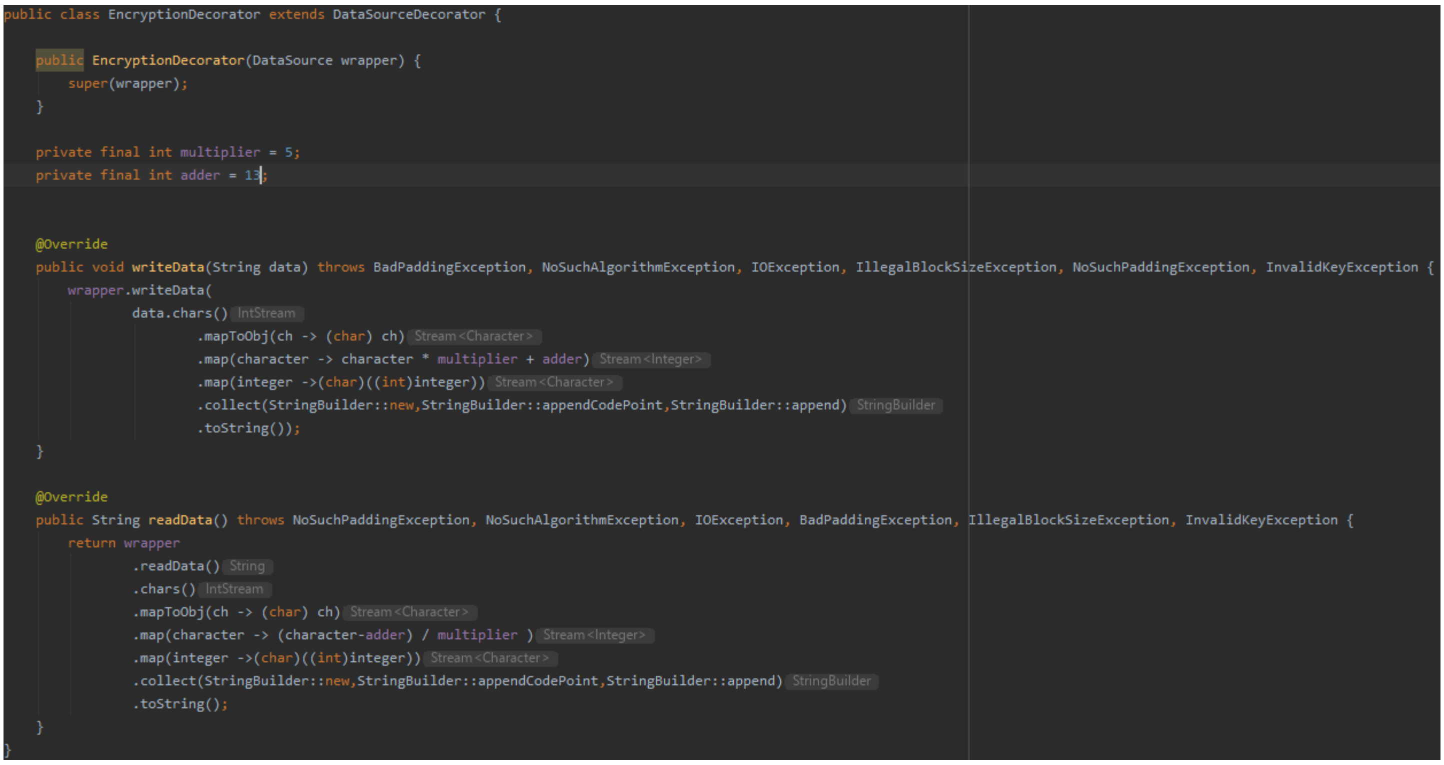
```java
public class FileDataSource implements  DataSource {
    private final String filename;

    public FileDataSource(String filename) { this.filename=filename; }

    public String getFilename() { return filename; }

    @Override
    public void writeData(String data) throws IOException, NoSuchAlgorithmException, BadPaddingException, IllegalBlockSizeException, NoSuchPaddingException {
        FileWriter fw = new FileWriter(filename);

        fw.write(data);
        fw.close();
    }

    @Override
    public String readData() throws IOException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, BadPaddingException, IllegalBlockSizeException {
        FileReader fr = new FileReader(filename);
        BufferedReader br = new BufferedReader(fr);
        StringBuilder result = new StringBuilder();
        String line = null;
        while((line = br.readLine()) != null){
                result.append(line);
                result.append("\n");
        }
        br.close();
        return result.toString();
    }
}
```

d) Klasa DataSourceDecorator



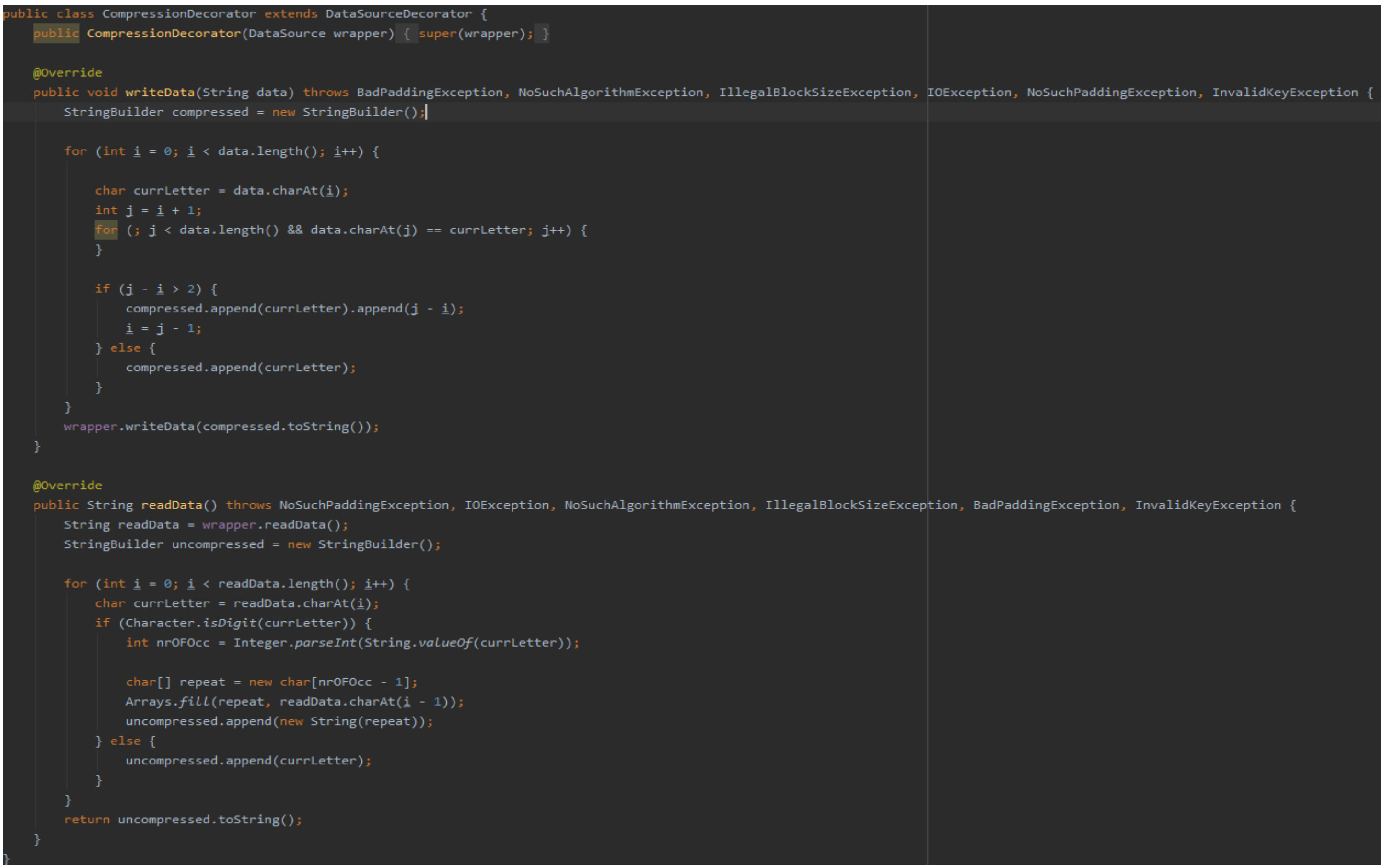
```java
public class DataSourceDecorator implements DataSource {

    DataSource wrapper;

    public DataSourceDecorator( DataSource wrapper) { this.wrapper=wrapper; }

    @Override
    public void writeData(String data) throws IOException, NoSuchAlgorithmException, BadPaddingException, IllegalBlockSizeException, NoSuchPaddingException, InvalidKeyException {

    }

    @Override
    public String readData() throws IOException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, BadPaddingException, IllegalBlockSizeException {
        return null;
    }
}
```

e) Klasa EncryptionDecorator – ze względu na problemy z gotowymi rozwiązaniami postanowiliśmy stworzyć własny algorytm. Algorytm ten przyjmuje tylko jako argumenty wyrazy nieposiadające żadnej liczby.



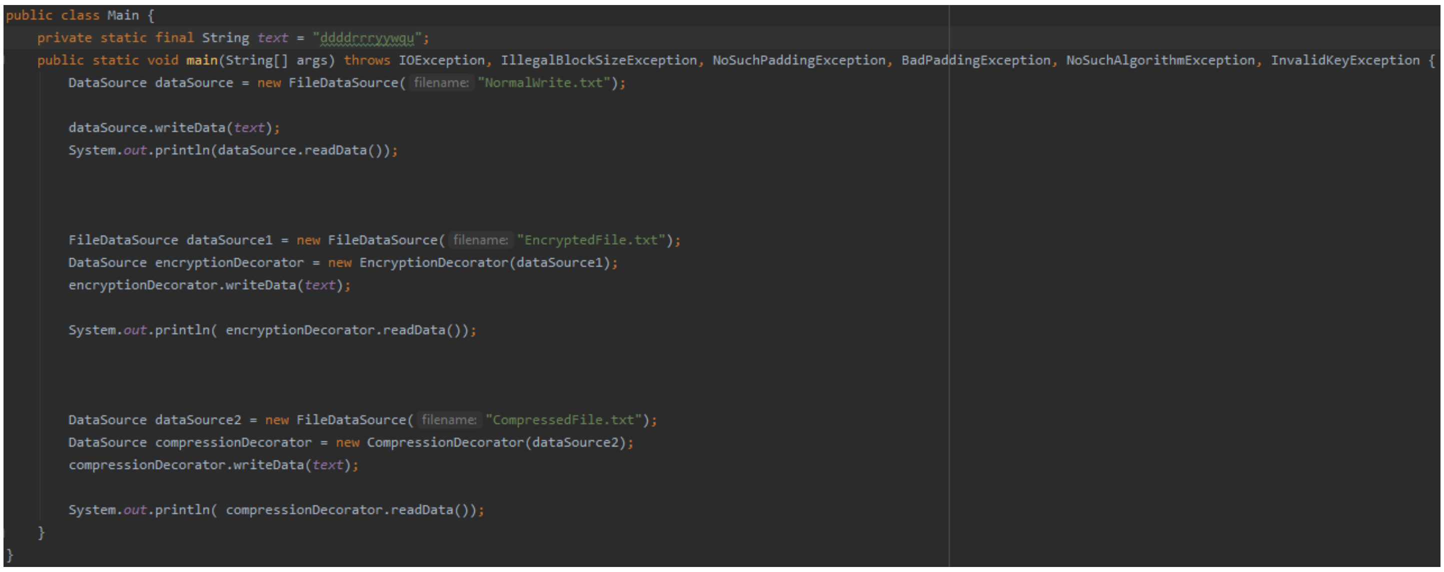
```java
public class EncryptionDecorator extends DataSourceDecorator {

    public EncryptionDecorator(DataSource wrapper) {
        super(wrapper);
    }

    private final int multiplier = 5;
    private final int adder = 13;


    @Override
    public void writeData(String data) throws BadPaddingException, NoSuchAlgorithmException, IOException, IllegalBlockSizeException, NoSuchPaddingException, InvalidKeyException {
        wrapper.writeData(
                data.chars() IntStream
                        .mapToObj(ch -> (char) ch) Stream<Character>
                        .map(character -> character * multiplier + adder) Stream<Integer>
                        .map(integer ->(char)((int)integer)) Stream<Character>
                        .collect(StringBuilder::new,StringBuilder::appendCodePoint,StringBuilder::append) StringBuilder
                        .toString());
    }

    @Override
    public String readData() throws NoSuchPaddingException, NoSuchAlgorithmException, IOException, BadPaddingException, IllegalBlockSizeException, InvalidKeyException {
        return wrapper
                .readData() String
                .chars() IntStream
                .mapToObj(ch -> (char) ch) Stream<Character>
                .map(character -> (character-adder) / multiplier ) Stream<Integer>
                .map(integer ->(char)((int)integer)) Stream<Character>
                .collect(StringBuilder::new,StringBuilder::appendCodePoint,StringBuilder::append) StringBuilder
                .toString();
    }
}
```

f) Klasa CompressionDecorator



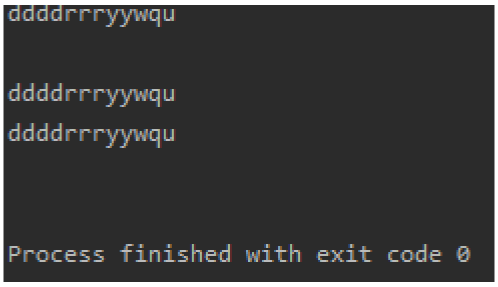
```java
public class CompressionDecorator extends DataSourceDecorator {
    public CompressionDecorator(DataSource wrapper) { super(wrapper); }

    @Override
    public void writeData(String data) throws BadPaddingException, NoSuchAlgorithmException, IllegalBlockSizeException, IOException, NoSuchPaddingException, InvalidKeyException {
        StringBuilder compressed = new StringBuilder();

        for (int i = 0; i < data.length(); i++) {

            char currLetter = data.charAt(i);
            int j = i + 1;
            for (; j < data.length() && data.charAt(j) == currLetter; j++) {
            }

            if (j - i > 2) {
                compressed.append(currLetter).append(j - i);
                i = j - 1;
            } else {
                compressed.append(currLetter);
            }
        }
        wrapper.writeData(compressed.toString());
    }

    @Override
    public String readData() throws NoSuchPaddingException, IOException, NoSuchAlgorithmException, IllegalBlockSizeException, BadPaddingException, InvalidKeyException {
        String readData = wrapper.readData();
        StringBuilder uncompressed = new StringBuilder();

        for (int i = 0; i < readData.length(); i++) {
            char currLetter = readData.charAt(i);
            if (Character.isDigit(currLetter)) {
                int nrOFOcc = Integer.parseInt(String.valueOf(currLetter));

                char[] repeat = new char[nrOFOcc - 1];
                Arrays.fill(repeat, readData.charAt(i - 1));
                uncompressed.append(new String(repeat));
            } else {
                uncompressed.append(currLetter);
            }
        }
        return uncompressed.toString();
    }
}
```

g) Klasa Main

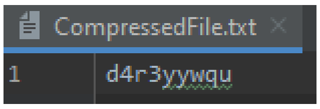


```java
public class Main {
    private static final String text = "ddddrrryywqu";
    public static void main(String[] args) throws IOException, IllegalBlockSizeException, NoSuchPaddingException, BadPaddingException, NoSuchAlgorithmException, InvalidKeyException {
        DataSource dataSource = new FileDataSource( filename: "NormalWrite.txt");

        dataSource.writeData(text);
        System.out.println(dataSource.readData());



        FileDataSource dataSource1 = new FileDataSource( filename: "EncryptedFile.txt");
        DataSource encryptionDecorator = new EncryptionDecorator(dataSource1);
        encryptionDecorator.writeData(text);

        System.out.println( encryptionDecorator.readData());



        DataSource dataSource2 = new FileDataSource( filename: "CompressedFile.txt");
        DataSource compressionDecorator = new CompressionDecorator(dataSource2);
        compressionDecorator.writeData(text);

        System.out.println( compressionDecorator.readData());
    }
}
```

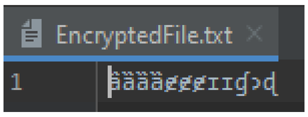
h) Efekt wykonania



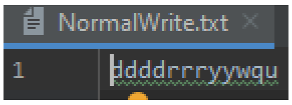
```
ddddrrryywqu

ddddrrryywqu
ddddrrryywqu



Process finished with exit code 0
```



```
CompressedFile.txt
d4r3yywqu
```



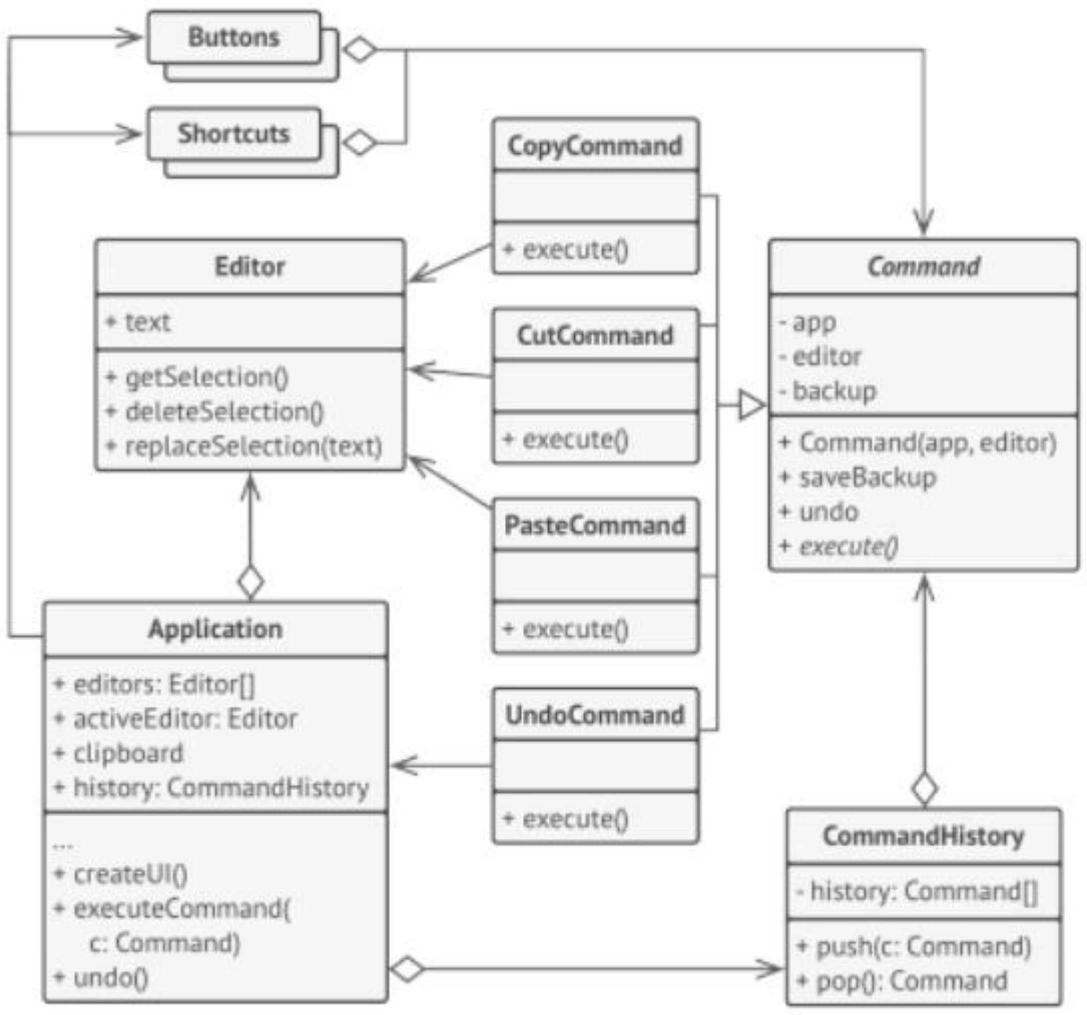
```
EncryptedFile.txt
ăăăăęęęɪɪɡ>d
```



```
NormalWrite.txt
ddddrrryywqu
```

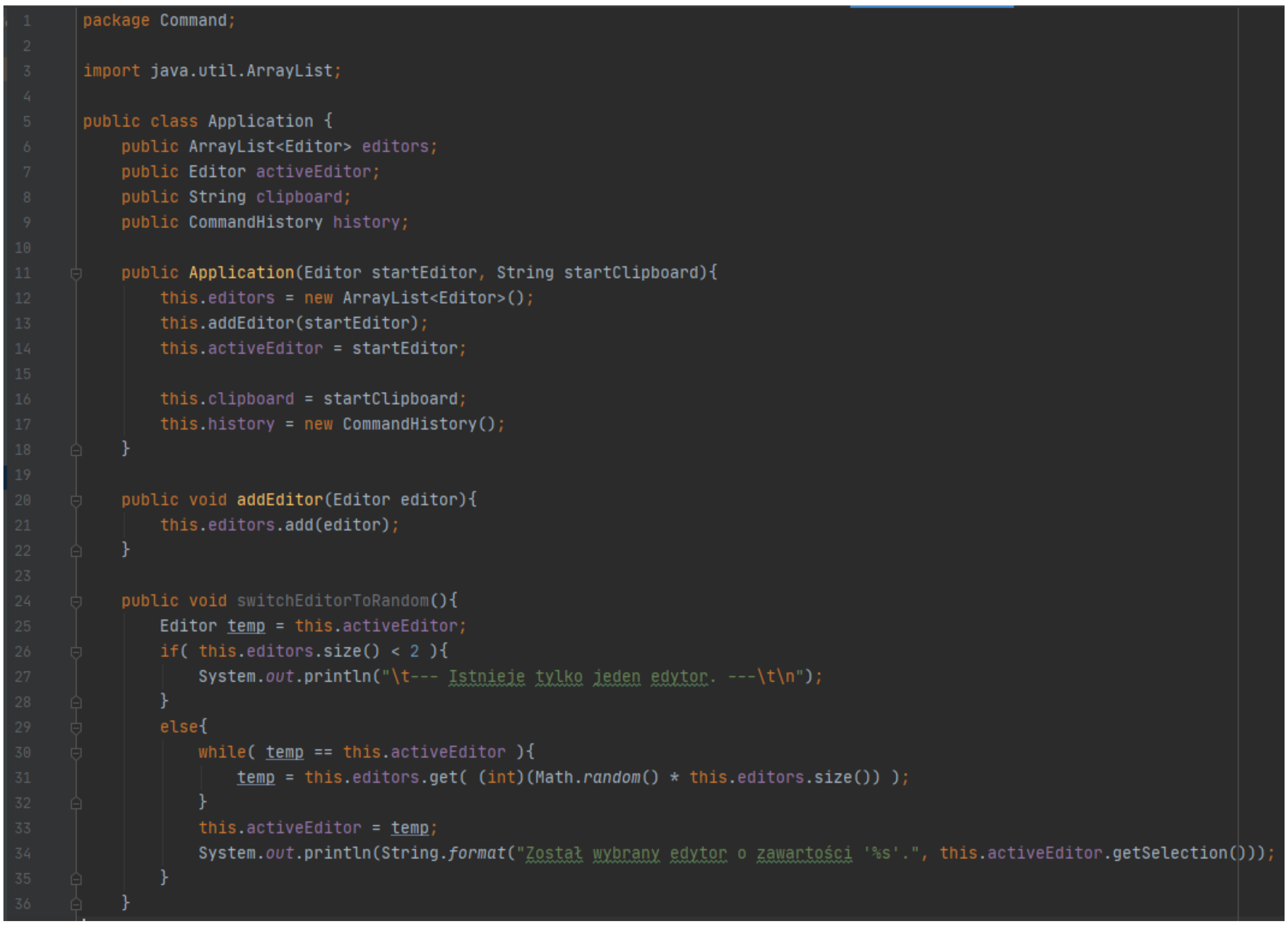
3. Zadanie 3

a) Zaimplementowaliśmy aplikację według schematu:

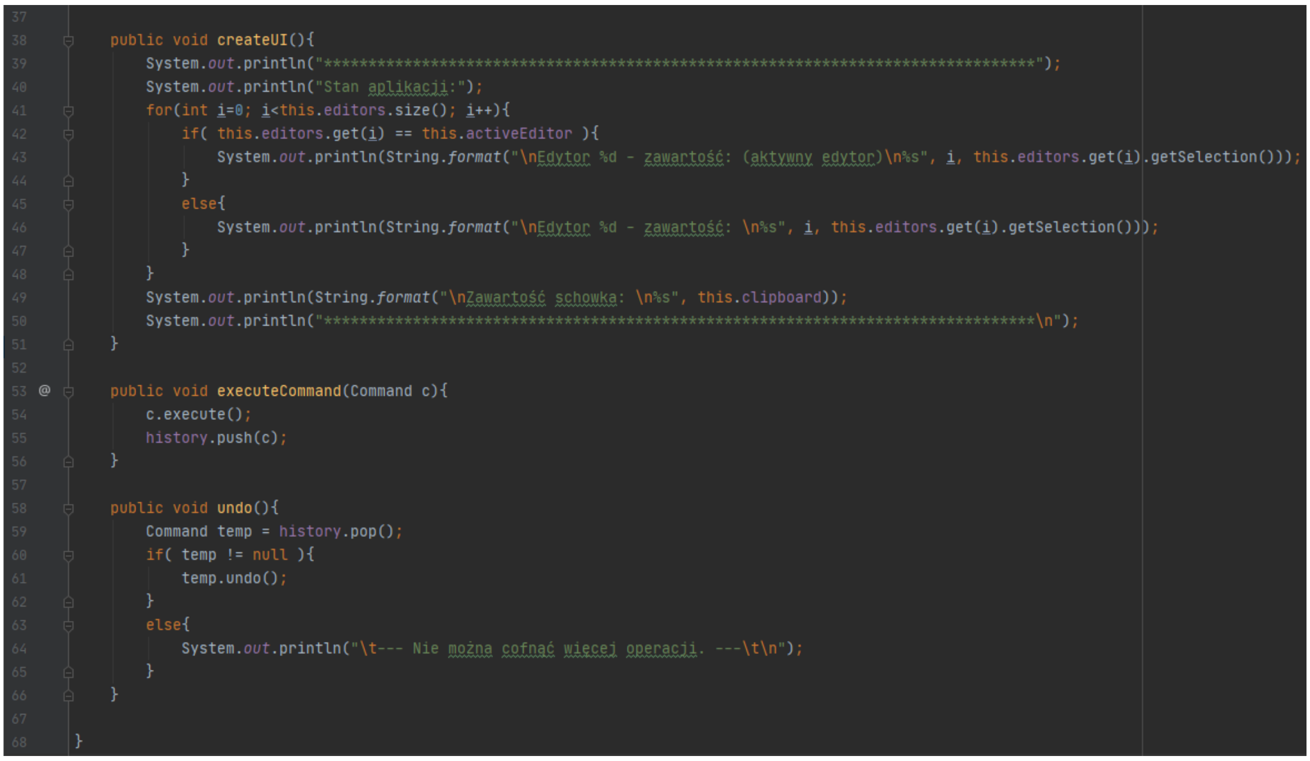


Rysunek 3: Undoable operations in a text editor.

b) Klasa Application:

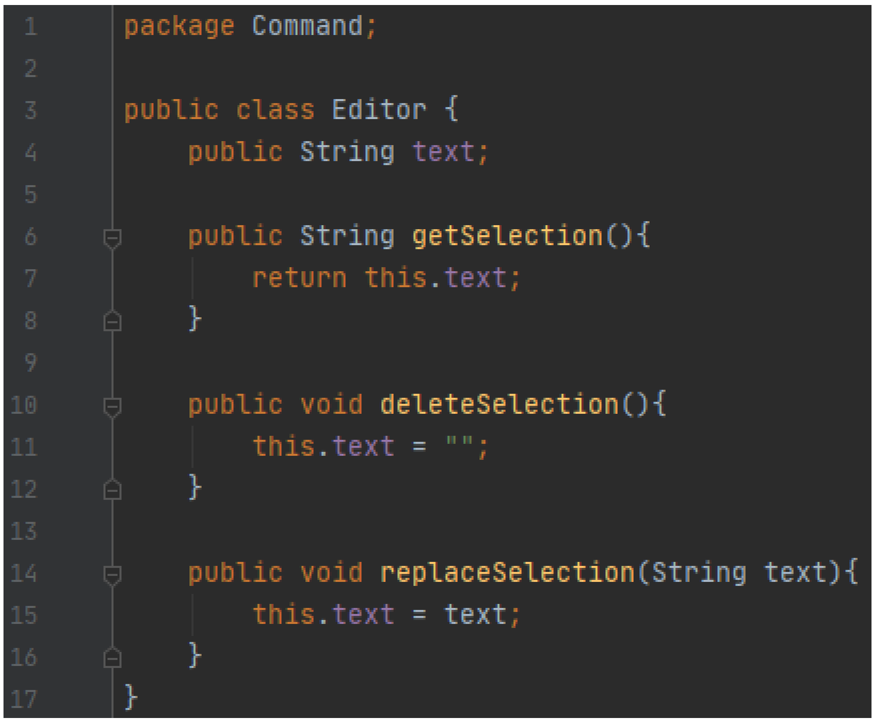


```java
package Command;

import java.util.ArrayList;

public class Application {
    public ArrayList<Editor> editors;
    public Editor activeEditor;
    public String clipboard;
    public CommandHistory history;

    public Application(Editor startEditor, String startClipboard){
        this.editors = new ArrayList<Editor>();
        this.addEditor(startEditor);
        this.activeEditor = startEditor;

        this.clipboard = startClipboard;
        this.history = new CommandHistory();
    }

    public void addEditor(Editor editor){
        this.editors.add(editor);
    }

    public void switchEditorToRandom(){
        Editor temp = this.activeEditor;
        if( this.editors.size() < 2 ){
            System.out.println("\t--- Istnieje tylko jeden edytor. ---\t\n");
        }
        else{
            while( temp == this.activeEditor ){
                temp = this.editors.get( (int)(Math.random() * this.editors.size()) );
            }
            this.activeEditor = temp;
            System.out.println(String.format("Został wybrany edytor o zawartości '%s'.", this.activeEditor.getSelection()));
        }
    }
```



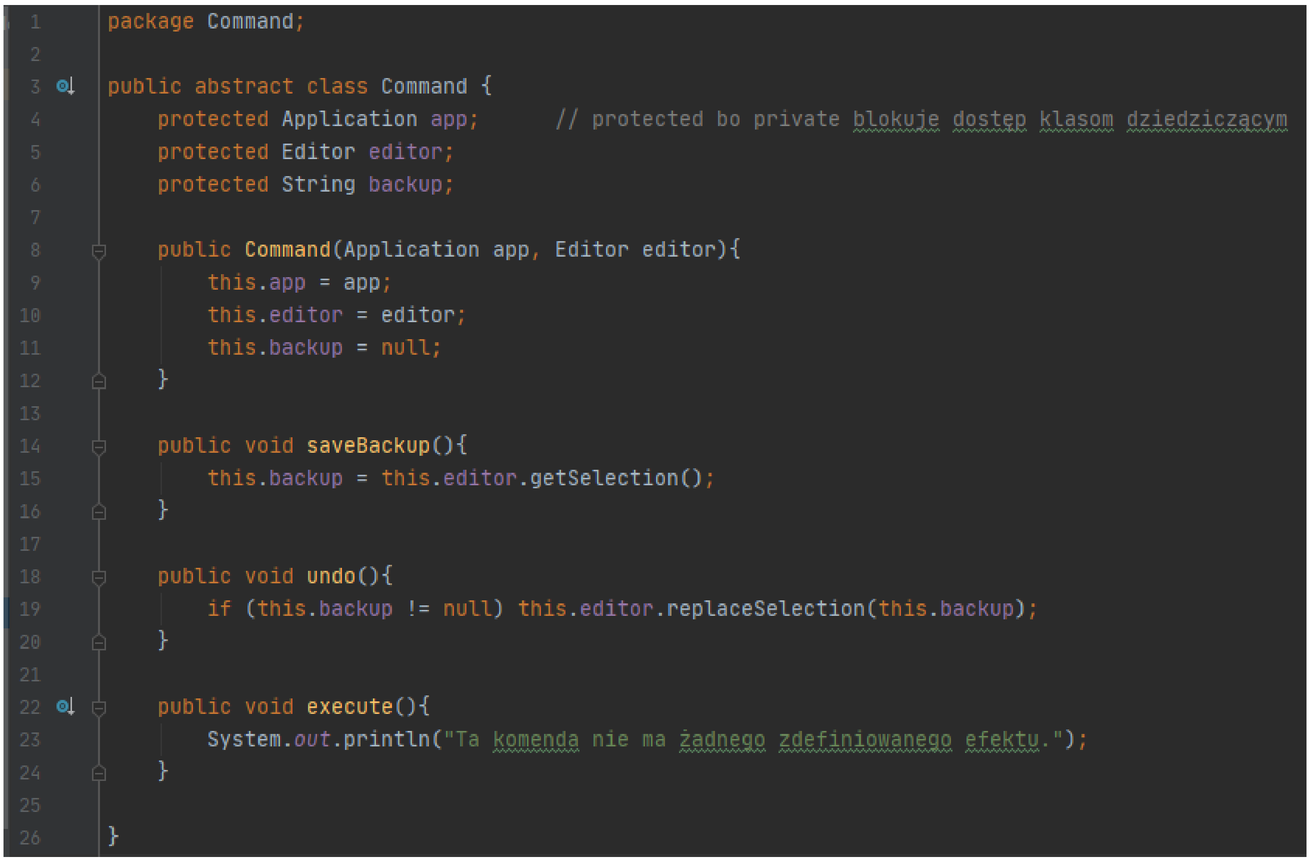
```java

    public void createUI(){
        System.out.println("***********************************************************************************");
        System.out.println("Stan aplikacji:");
        for(int i=0; i<this.editors.size(); i++){
            if( this.editors.get(i) == this.activeEditor ){
                System.out.println(String.format("\nEdytor %d - zawartość: (aktywny edytor)\n%s", i, this.editors.get(i).getSelection()));
            }
            else{
                System.out.println(String.format("\nEdytor %d - zawartość: \n%s", i, this.editors.get(i).getSelection()));
            }
        }
        System.out.println(String.format("\nZawartość schowka: \n%s", this.clipboard));
        System.out.println("***********************************************************************************\n");
    }

    public void executeCommand(Command c){
        c.execute();
        history.push(c);
    }

    public void undo(){
        Command temp = history.pop();
        if( temp != null ){
            temp.undo();
        }
        else{
            System.out.println("\t--- Nie można cofnąć więcej operacji. ---\t\n");
        }
    }

}
```

c) Klasa Editor:



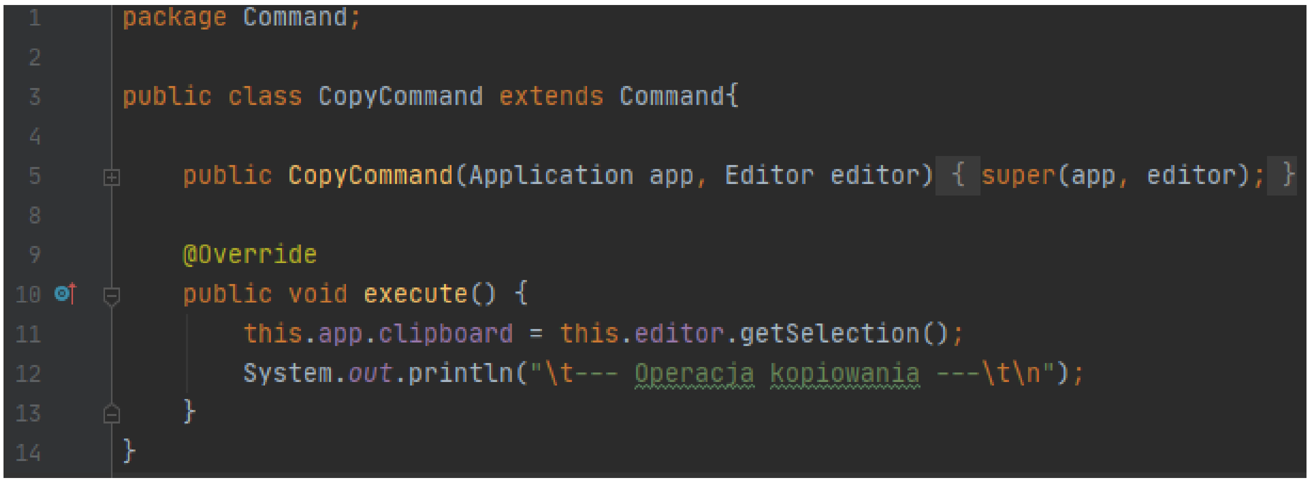
```java
package Command;

public class Editor {
    public String text;

    public String getSelection(){
        return this.text;
    }

    public void deleteSelection(){
        this.text = "";
    }

    public void replaceSelection(String text){
        this.text = text;
    }
}
```

d) Klasa Command:



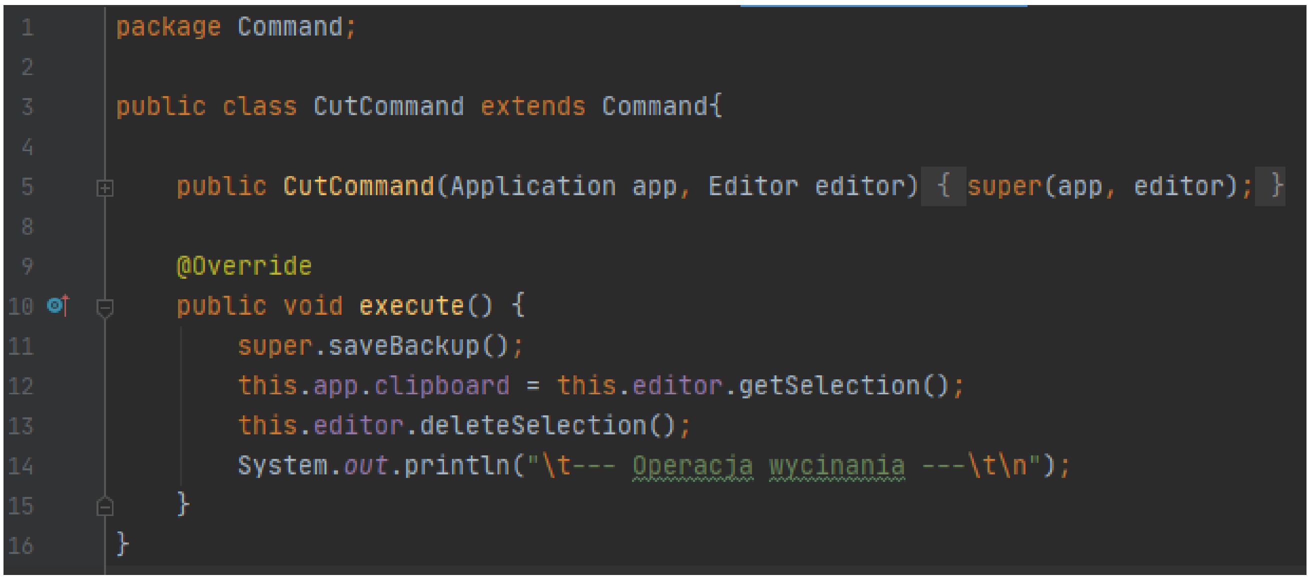
```java
package Command;

public abstract class Command {
    protected Application app;      // protected bo private blokuje dostep klasom dziedziczacym
    protected Editor editor;
    protected String backup;

    public Command(Application app, Editor editor){
        this.app = app;
        this.editor = editor;
        this.backup = null;
    }

    public void saveBackup(){
        this.backup = this.editor.getSelection();
    }

    public void undo(){
        if (this.backup != null) this.editor.replaceSelection(this.backup);
    }

    public void execute(){
        System.out.println("Ta komenda nie ma żadnego zdefiniowanego efektu.");
    }

}
```

e) Podklasa CopyCommand:



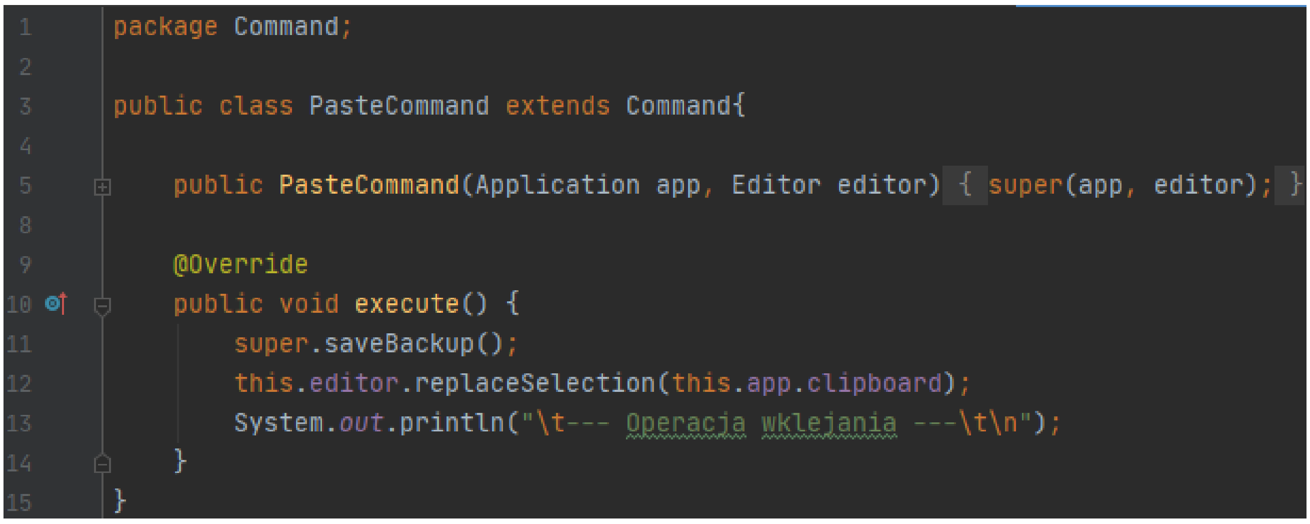
```java
package Command;

public class CopyCommand extends Command{

    public CopyCommand(Application app, Editor editor) { super(app, editor); }

    @Override
    public void execute() {
        this.app.clipboard = this.editor.getSelection();
        System.out.println("\t--- Operacja kopiowania ---\t\n");
    }
}
```

f) Podklasa CutCommand:



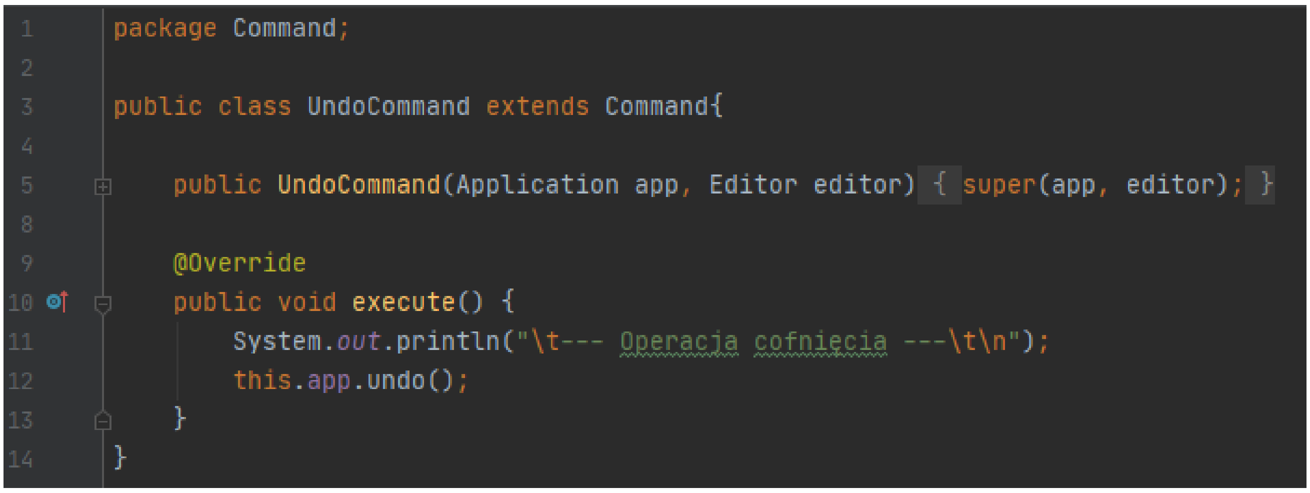
```java
package Command;

public class CutCommand extends Command{

    public CutCommand(Application app, Editor editor) { super(app, editor); }

    @Override
    public void execute() {
        super.saveBackup();
        this.app.clipboard = this.editor.getSelection();
        this.editor.deleteSelection();
        System.out.println("\t--- Operacja wycinania ---\t\n");
    }
}
```

g) Podklasa PasteCommand:



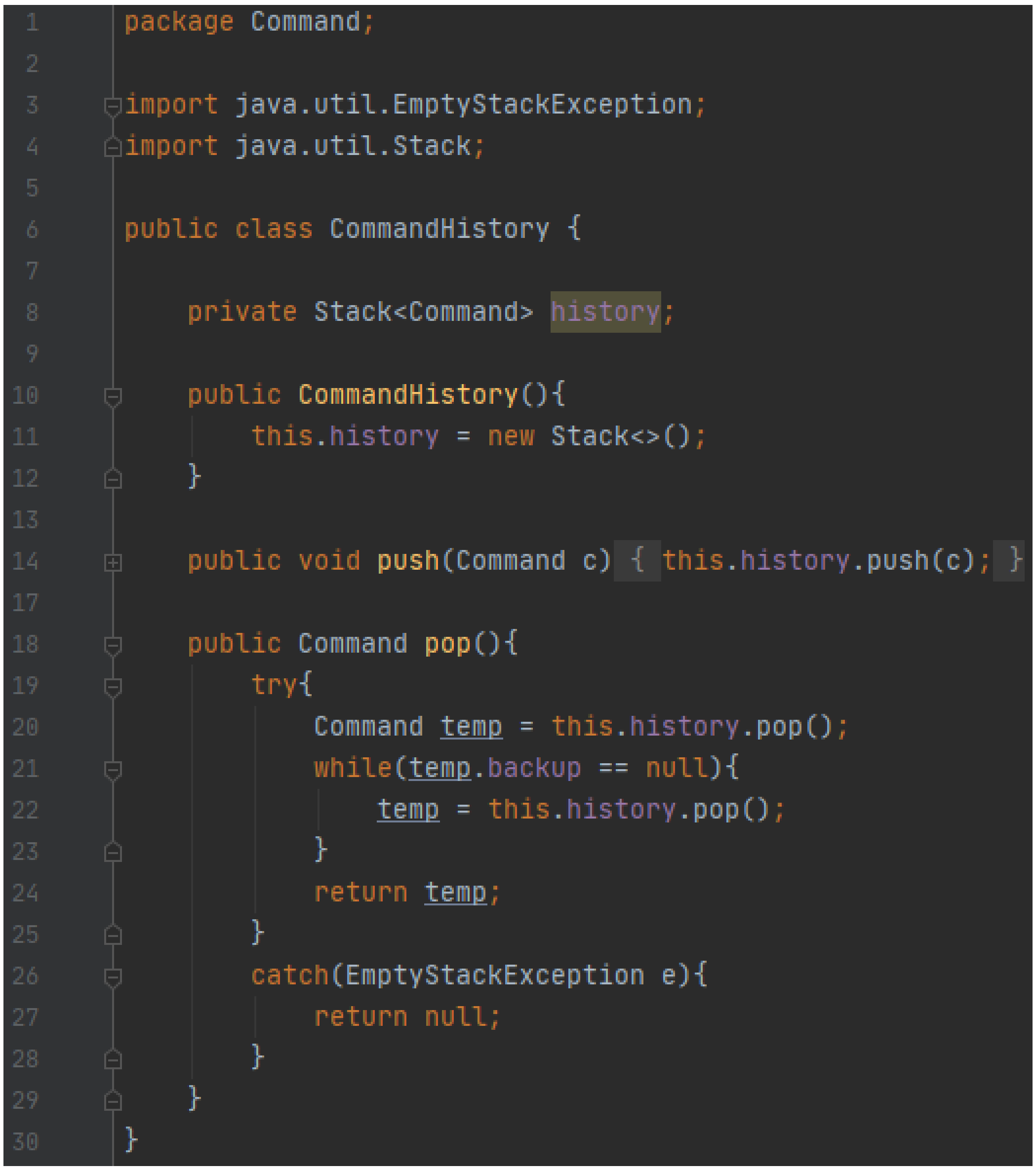
```java
package Command;

public class PasteCommand extends Command{

    public PasteCommand(Application app, Editor editor) { super(app, editor); }

    @Override
    public void execute() {
        super.saveBackup();
        this.editor.replaceSelection(this.app.clipboard);
        System.out.println("\t--- Operacja wklejania ---\t\n");
    }
}
```

h) Podklasa UndoCommand:



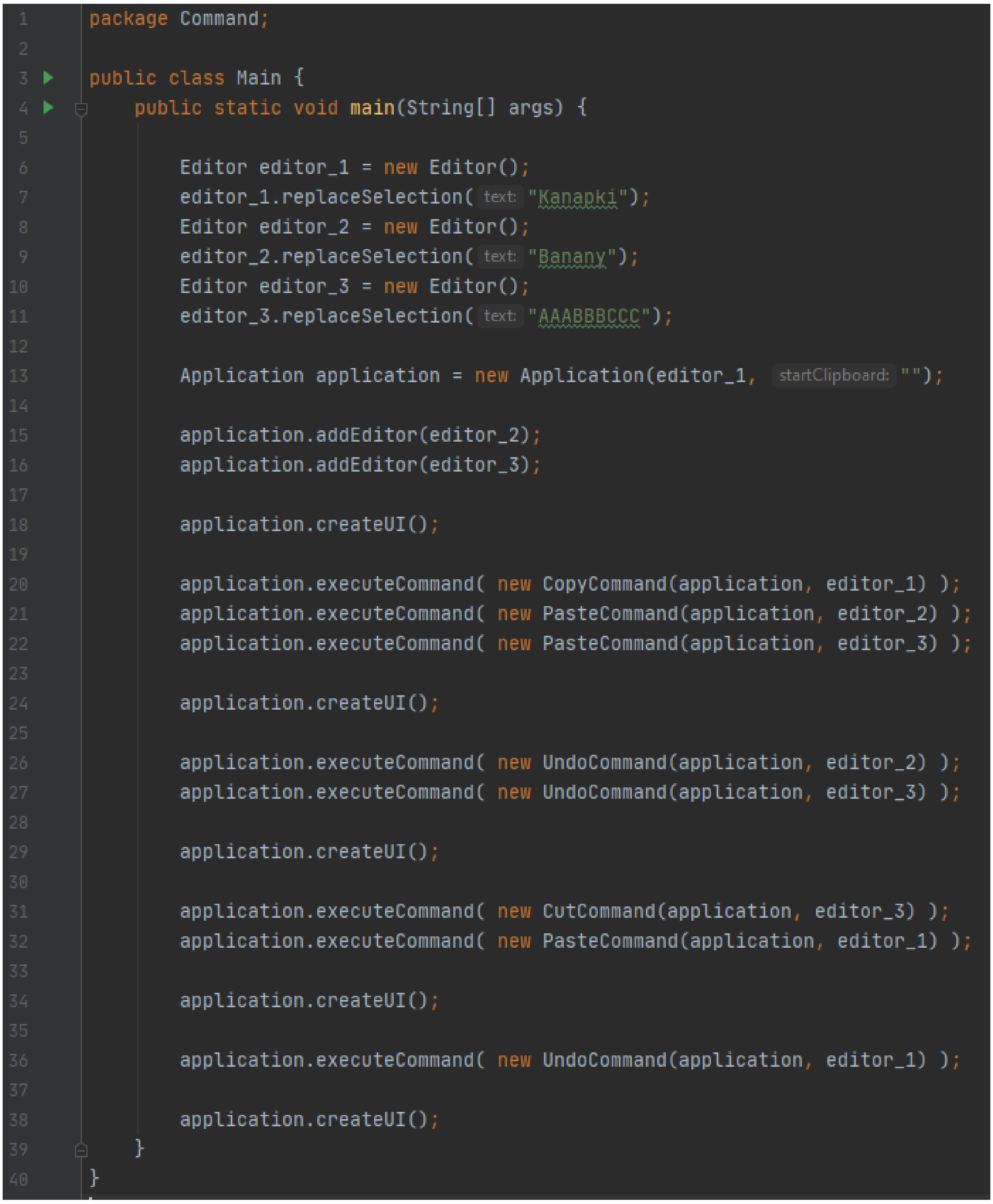
```java
package Command;

public class UndoCommand extends Command{

    public UndoCommand(Application app, Editor editor) { super(app, editor); }

    @Override
    public void execute() {
        System.out.println("\t--- Operacja cofniecia ---\t\n");
        this.app.undo();
    }
}
```

i) Klasa CommandHistory:



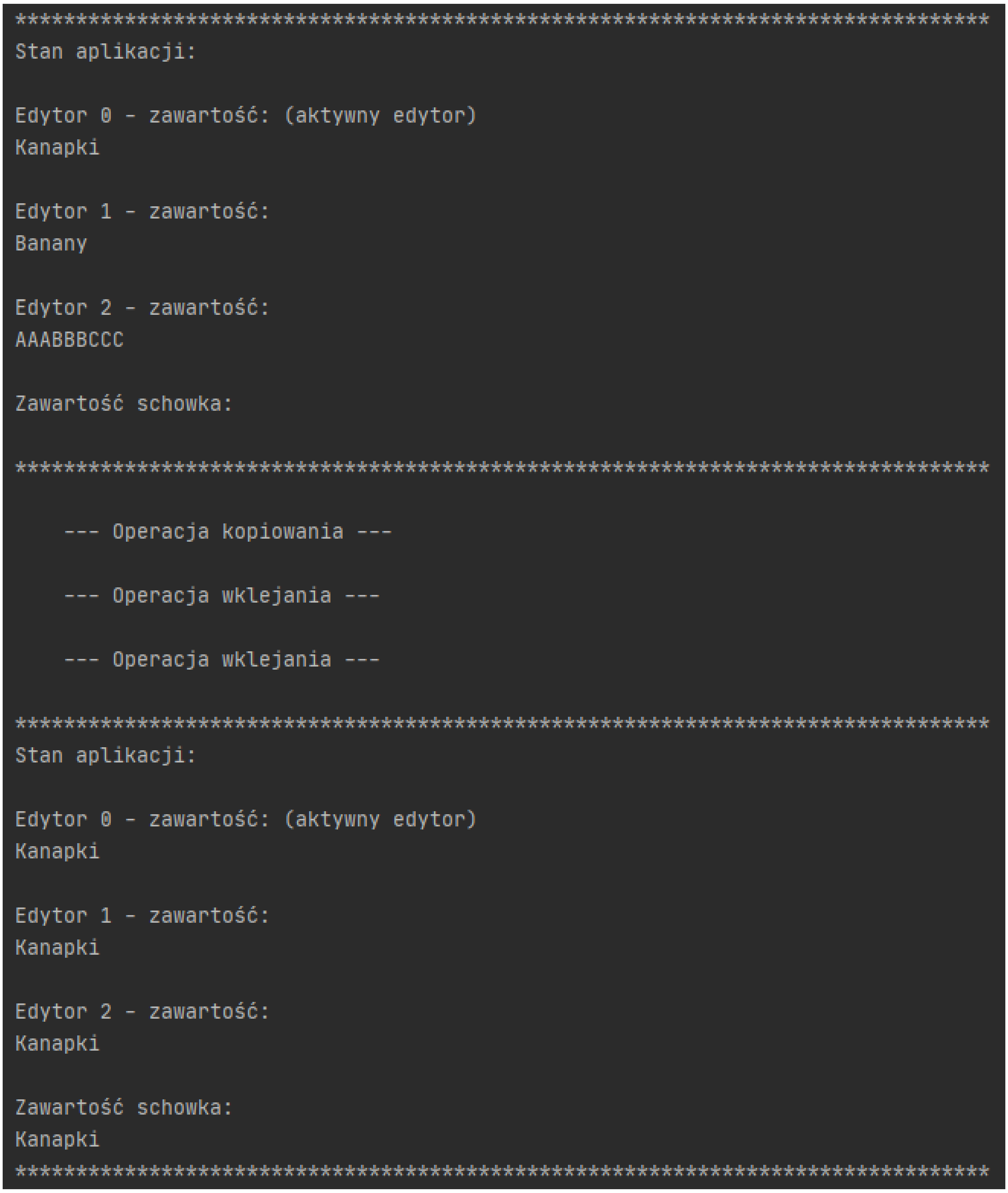
```java
package Command;

import java.util.EmptyStackException;
import java.util.Stack;

public class CommandHistory {

    private Stack<Command> history;

    public CommandHistory(){
        this.history = new Stack<>();
    }

    public void push(Command c) { this.history.push(c); }

    public Command pop(){
        try{
            Command temp = this.history.pop();
            while(temp.backup == null){
                temp = this.history.pop();
            }
            return temp;
        }
        catch(EmptyStackException e){
            return null;
        }
    }
}
```

j) I klasa Main wywołująca mockupowy program.

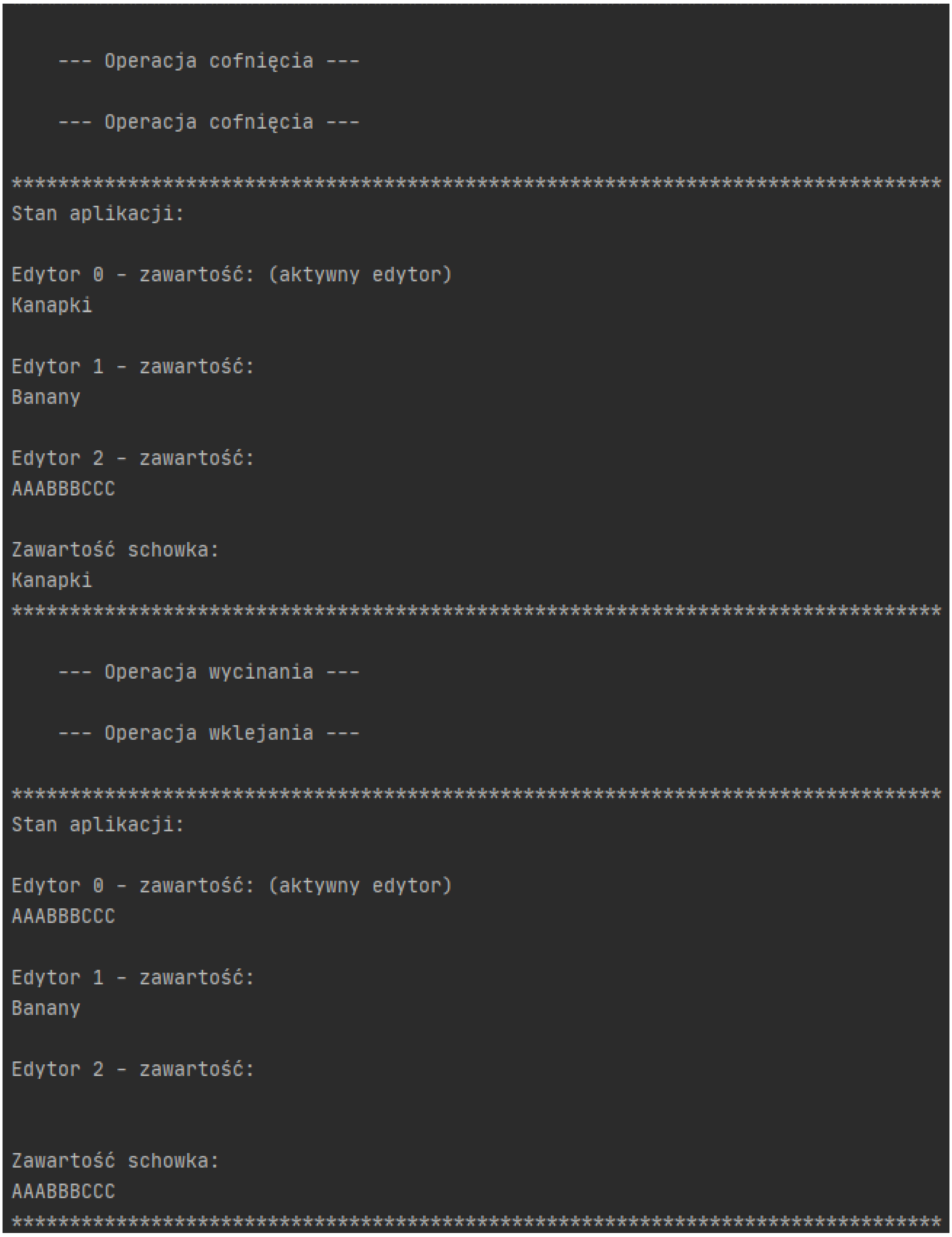


```java
package Command;

public class Main {
    public static void main(String[] args) {

        Editor editor_1 = new Editor();
        editor_1.replaceSelection( text: "Kanapki");
        Editor editor_2 = new Editor();
        editor_2.replaceSelection( text: "Banany");
        Editor editor_3 = new Editor();
        editor_3.replaceSelection( text: "AAABBBCCC");

        Application application = new Application(editor_1,  startClipboard: "");

        application.addEditor(editor_2);
        application.addEditor(editor_3);

        application.createUI();

        application.executeCommand( new CopyCommand(application, editor_1) );
        application.executeCommand( new PasteCommand(application, editor_2) );
        application.executeCommand( new PasteCommand(application, editor_3) );

        application.createUI();

        application.executeCommand( new UndoCommand(application, editor_2) );
        application.executeCommand( new UndoCommand(application, editor_3) );

        application.createUI();

        application.executeCommand( new CutCommand(application, editor_3) );
        application.executeCommand( new PasteCommand(application, editor_1) );

        application.createUI();

        application.executeCommand( new UndoCommand(application, editor_1) );

        application.createUI();
    }
}
```

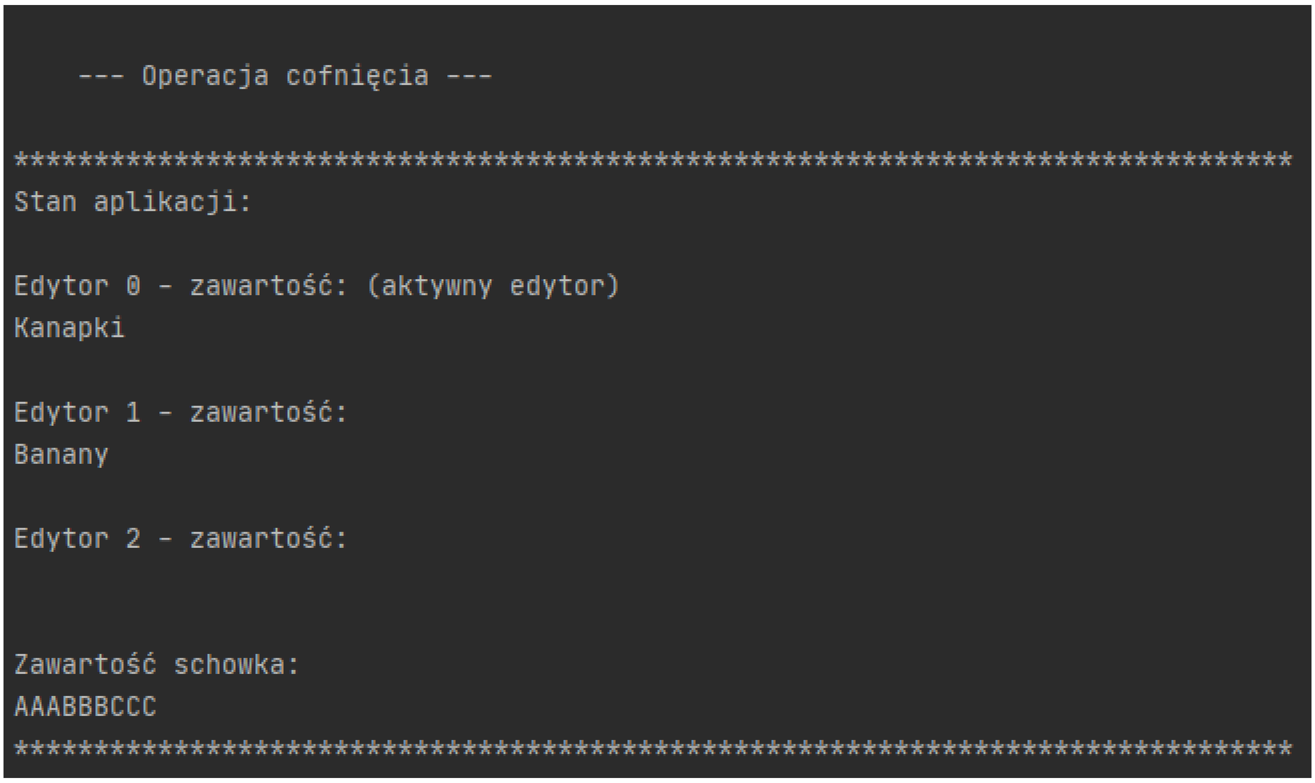
(i) Efekt wywołania:



```
********************************************************************************
Stan aplikacji:

Edytor 0 - zawartość: (aktywny edytor)
Kanapki

Edytor 1 - zawartość:
Banany

Edytor 2 - zawartość:
AAABBBCCC

Zawartość schowka:

********************************************************************************

    --- Operacja kopiowania ---

    --- Operacja wklejania ---

    --- Operacja wklejania ---

********************************************************************************
Stan aplikacji:

Edytor 0 - zawartość: (aktywny edytor)
Kanapki

Edytor 1 - zawartość:
Kanapki

Edytor 2 - zawartość:
Kanapki

Zawartość schowka:
Kanapki
********************************************************************************
```



```
    --- Operacja cofnięcia ---

    --- Operacja cofnięcia ---

********************************************************************************
Stan aplikacji:

Edytor 0 - zawartość: (aktywny edytor)
Kanapki

Edytor 1 - zawartość:
Banany

Edytor 2 - zawartość:
AAABBBCCC

Zawartość schowka:
Kanapki
********************************************************************************

    --- Operacja wycinania ---

    --- Operacja wklejania ---

********************************************************************************
Stan aplikacji:

Edytor 0 - zawartość: (aktywny edytor)
AAABBBCCC

Edytor 1 - zawartość:
Banany

Edytor 2 - zawartość:


Zawartość schowka:
AAABBBCCC
********************************************************************************
```



```
    --- Operacja cofnięcia ---

*********************************************************************************
Stan aplikacji:

Edytor 0 - zawartość: (aktywny edytor)
Kanapki

Edytor 1 - zawartość:
Banany

Edytor 2 - zawartość:



Zawartość schowka:
AAABBBCCC
*********************************************************************************
```