

1) Figure.java

```
1 package decorator.e3;
2
3 public abstract class Figure implements Cloneable {
4     int x, y;
5     boolean _handle, _shadow;
6
7     public Figure() {
8         _handle = false;
9         _shadow = false;
10    }
11
12    public void move() {
13        x += 10;
14        y += 10;
15    }
16
17    public boolean isHandled() {
18        return _handle;
19    }
20
21    public boolean isShadowed() {
22        return _shadow;
23    }
24
25    public void setHandle(boolean value) {
26        _handle = value;
27    }
28
29    public void setShadow(boolean value) {
30        _shadow = value;
31    }
32
33    public abstract void drawFigure(int indent);
34
35    public void draw(int indent) {
36        drawFigure(indent);
37        if (_handle)
38            drawHandle(indent);
39        if (_shadow)
40            drawShadow(indent);
41    }
42
43    public void drawHandle(int indent) {
44        System.out.printf("%s Handle %d %dWn", " ".repeat(indent), x, y);
45    }
46
47    public void drawShadow(int indent) {
48        System.out.printf("%s Shadow %d %dWn", " ".repeat(indent), x, y);
49    }
50
51    @Override
52    public Figure clone() throws CloneNotSupportedException {
53        return (Figure) super.clone();
54    }
55 }
```

2) Rectangle.java

```
1 package decorator.e3;
2
3 public class Rectangle extends Figure {
4     String label;
5
6     public Rectangle(String label) {
7         this.label = label;
8     }
9
10    @Override
11    public void drawFigure(int indent) {
12        String padding = " ".repeat(indent);
13        System.out.printf("%sRectangle(%s, %d, %d)\n", padding, label, x, y);
14    }
15 }
```

3) Line.java

```
1 package decorator.e3;
2
3 public class Line extends Figure {
4     String label;
5
6     public Line(String label) {
7         this.label = label;
8     }
9
10    @Override
11    public void drawFigure(int indent) {
12        String padding = " ".repeat(indent);
13        System.out.printf("%sLine(%s, %d, %d)\n", padding, label, x, y);
14    }
15 }
```

4) FigureGroup.java

```
1 package decorator.e3;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class FigureGroup extends Figure {
7     private List<Figure> figures = new ArrayList<Figure>();
8
9     @Override
10    public void move() {
11        for (Figure figure : figures)
12            figure.move();
13    }
14
15    @Override
16    public Figure clone() throws CloneNotSupportedException {
17        FigureGroup group = new FigureGroup();
18        for (Figure figure : figures)
19            group.add(figure.clone());
20        return group;
21    }
22
23    @Override
24    public void drawFigure(int indent) {
25        String padding = " ".repeat(indent);
26        System.out.printf("%sGroup\n", padding);
27        for (Figure figure : figures)
28            figure.draw(indent + 1);
29        System.out.printf("%s\n", padding);
30    }
31
32    public void add(Figure f) {
33        figures.add(f);
34    }
35
36    public int getCount() { return figures.size(); }
37    public Figure get(int index) { return figures.get(index); }
38 }
```

5) Example3.java

```
1 package decorator.e3;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class Example3 {
8
9     static List<Figure> figures = new ArrayList<Figure>();
10
11     static void drawFigures() {
12         System.out.println();
13         for (int i = 0; i < figures.size(); ++i) {
14             System.out.printf("%d: ", i);
15             figures.get(i).draw();
16         }
17         System.out.println();
18     }
19
20     static void execute(String cmd) {
21         try {
22             int index;
23             Figure figure;
24             String[] a = cmd.split("[, ]+");
25             switch (a[0].toLowerCase()) {
26                 case "rectangle": figures.add(new Rectangle(a[1])); break;
27                 case "line": figures.add(new Line(a[1])); break;
28                 case "move":
29                     figure = figures.get(Integer.valueOf(a[1]));
30                     figure.move();
31                     break;
32                 case "remove":
33                     index = Integer.valueOf(a[1]);
34                     figures.remove(index);
35                     break;
36                 case "duplicate":
37                     figure = figures.get(Integer.valueOf(a[1]));
38                     figures.add(figure.clone());
39                     break;
40                 case "group":
41                     FigureGroup group = new FigureGroup();
42                     for (int i = 1; i < a.length; ++i)
43                         group.add(figures.get(Integer.valueOf(a[i])));
44                     figures.add(group);
45                     for (int i = 0; i < group.getCount(); ++i)
46                         figures.remove(group.get(i));
47                     break;
48                 case "shadow":
49                     index = Integer.valueOf(a[1]);
50                     figure = figures.get(index);
51                     figure.setShadow(!figure.isShadowed());
52                     break;
53                 case "handle":
54                     index = Integer.valueOf(a[1]);
55                     figure = figures.get(index);
56                     figure.setHandle(!figure.isHandled());
57                     break;
58                 case "quit": System.exit(0); break;
59             }
60         } catch (Exception e) {
61         }
62     }
63
64     static void prompt() {
65         System.out.printf(" 사각형 : rectangle 레이블\n");
66         System.out.printf(" 선 : line 레이블\n");
67         System.out.printf(" 이동 : move 번호\n");
68         System.out.printf(" 삭제 : remove 번호\n");
```

69	System.out.printf("복제 : duplicate 번호\n");
70	System.out.printf("그룹 : group 번호1, 번호2,...\n");
71	System.out.printf("그림자 : shadow <객체번호>\n");
72	System.out.printf("핸들 : handle <객체번호>\n");
73	System.out.printf("종료 : quit\n");
74	System.out.printf(" ? ");
75	}
76	
77	public static void main(String[] args) {
78	try (Scanner scanner = new Scanner(System.in)) {
79	while (true) {
80	prompt();
81	String cmd = scanner.nextLine();
82	execute(cmd);
83	drawFigures();
84	}
85	}
86	}
87	}

실행 사례

사각형 : rectangle 레이블
선 : line 레이블
이동 : move 번호
삭제 : remove 번호
복제 : duplicate 번호
그룹 : group 번호1, 번호2,...
그림자 : shadow <객체번호>
핸들 : handle <객체번호>
종료 : quit
? rectangle A
0: Rectangle(A, 0, 0)
사각형 : rectangle 레이블
선 : line 레이블
이동 : move 번호
삭제 : remove 번호
복제 : duplicate 번호
그룹 : group 번호1, 번호2,...
그림자 : shadow <객체번호>
핸들 : handle <객체번호>
종료 : quit
? move 0
0: Rectangle(A, 10, 10)
사각형 : rectangle 레이블
선 : line 레이블
이동 : move 번호
삭제 : remove 번호
복제 : duplicate 번호
그룹 : group 번호1, 번호2,...
그림자 : shadow <객체번호>
핸들 : handle <객체번호>
종료 : quit
? shadow 0
0: Rectangle(A, 10, 10)
Shadow 10 10
사각형 : rectangle 레이블
선 : line 레이블
이동 : move 번호
삭제 : remove 번호
복제 : duplicate 번호
그룹 : group 번호1, 번호2,...
그림자 : shadow <객체번호>

핸들 : handle <객체번호>

종료 : quit

? handle 0

0: Rectangle(A, 10, 10)

Handle 10 10

Shadow 10 10

사각형 : rectangle 레이블

선 : line 레이블

이동 : move 번호

삭제 : remove 번호

복제 : duplicate 번호

그룹 : group 번호1, 번호2,...

그림자 : shadow <객체번호>

핸들 : handle <객체번호>

종료 : quit

? move 0

0: Rectangle(A, 20, 20)

Handle 20 20

Shadow 20 20

사각형 : rectangle 레이블

선 : line 레이블

이동 : move 번호

삭제 : remove 번호

복제 : duplicate 번호

그룹 : group 번호1, 번호2,...

그림자 : shadow <객체번호>

핸들 : handle <객체번호>

종료 : quit

? duplicate 0

0: Rectangle(A, 20, 20)

Handle 20 20

Shadow 20 20

1: Rectangle(A, 20, 20)

Handle 20 20

Shadow 20 20

사각형 : rectangle 레이블

선 : line 레이블

이동 : move 번호

삭제 : remove 번호

복제 : duplicate 번호

그룹 : group 번호1, 번호2,...

그림자 : shadow <객체번호>

핸들 : handle <객체번호>

종료 : quit

? group 0, 1

0: Group(

Rectangle(A, 20, 20)

Handle 20 20

Shadow 20 20

Rectangle(A, 20, 20)

Handle 20 20

Shadow 20 20

)

사각형 : rectangle 레이블

선 : line 레이블

이동 : move 번호

삭제 : remove 번호

복제 : duplicate 번호

그룹 : group 번호1, 번호2,...

그림자 : shadow <객체번호>

핸들 : handle <객체번호>

종료 : quit

? move 0

0: Group(

Rectangle(A, 30, 30)

Handle 30 30

Shadow 30 30

Rectangle(A, 30, 30)

Handle 30 30

Shadow 30 30

)