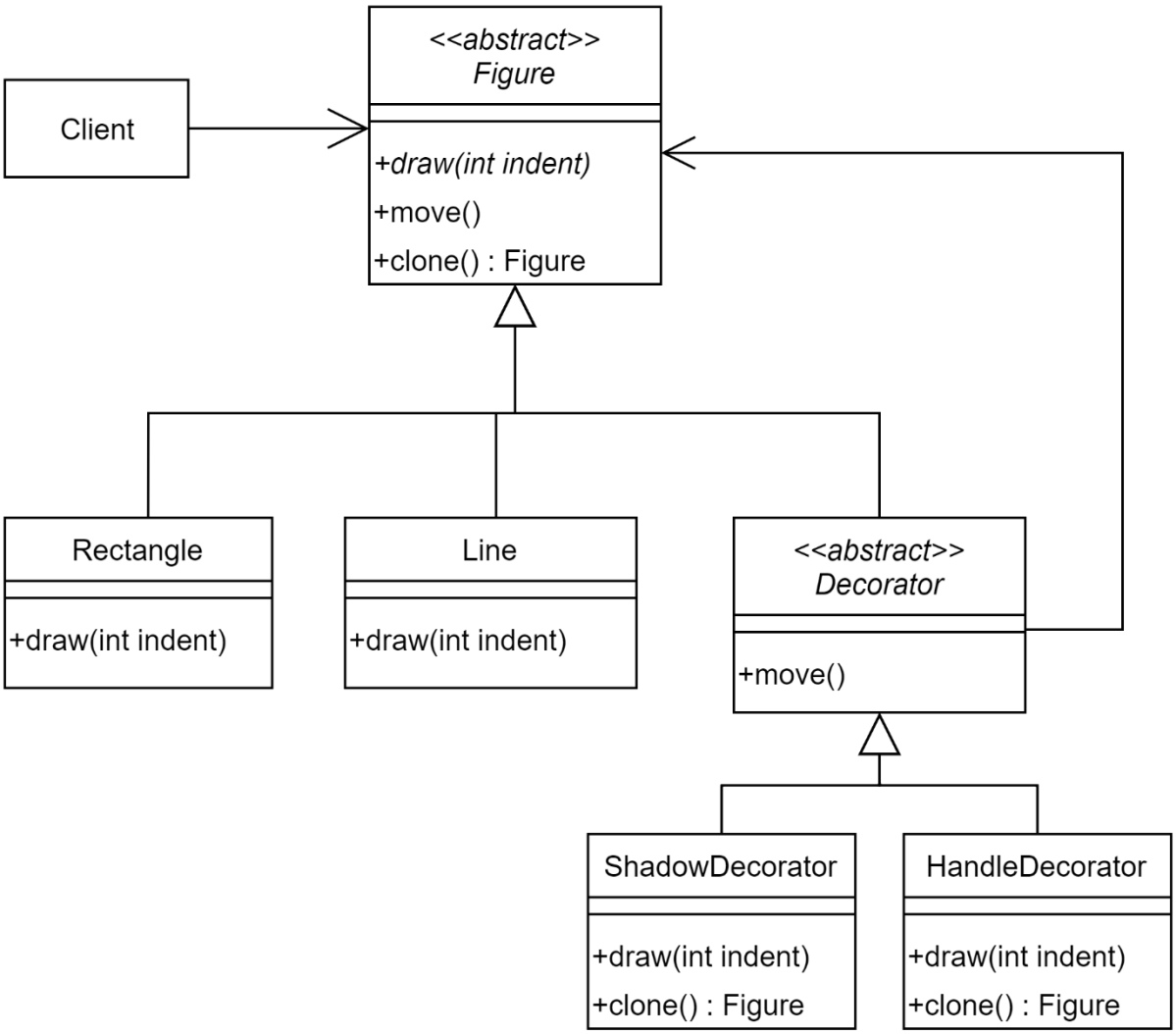
1) 개요


<<abstract>>
Figure
+draw(int indent)
+move()
+clone() : Figure
Client
Rectangle
+draw(int indent)
Line
+draw(int indent)
<<abstract>>
Decorator
+move()
ShadowDecorator
+draw(int indent)
+clone() : Figure
HandleDecorator
+draw(int indent)
+clone() : Figure

2) Figure.java

```java
package decorator.e2;

public abstract class Figure implements Cloneable {
    int x, y;

    public abstract void draw(int indent);

    public void move() {
        x += 10;
        y += 10;
    }

    @Override
    public Figure clone() throws CloneNotSupportedException {
        return (Figure)super.clone();
    }

    int getX() {
        return x;
    }

    int getY() {
        return y;
    }

}
```

3) Rectangle.java

```java
package decorator.e2;

public class Rectangle extends Figure {
    String label;

    public Rectangle(String label) {
        this.label = label;
    }

    @Override
    public void draw(int indent) {
        String padding = "  ".repeat(indent);
        System.out.printf("%sRectangle(%s, %d, %d)Wn", padding, label, x, y);
    }
}
```

4) Line.java

```java
package decorator.e2;

public class Line extends Figure {
    String label;

    public Line(String label) {
        this.label = label;
    }

    @Override
    public void draw(int indent) {
        String padding = "  ".repeat(indent);
        System.out.printf("%sLine(%s, %d, %d)Wn", padding, label, x, y);
    }
}
```

5) FigureGroup.java

```java
package decorator.e2;

import java.util.ArrayList;
import java.util.List;

public class FigureGroup extends Figure {
    private List<Figure> figures = new ArrayList<Figure>();

    @Override
    public void move() {
        for (Figure figure : figures)
            figure.move();
    }

    @Override
    public Figure clone() throws CloneNotSupportedException {
        FigureGroup group = new FigureGroup();
        for (Figure figure : figures)
            group.add(figure.clone());
        return group;
    }

    @Override
    public void draw(int indent) {
        String padding = "  ".repeat(indent);
        System.out.printf("%sGroup(Wn", padding);
        for (Figure figure : figures)
            figure.draw(indent + 1);
        System.out.printf("%s)Wn", padding);
    }

    public void add(Figure f) {
        figures.add(f);
    }

    public int getCount() { return figures.size(); }
    public Figure get(int index) { return figures.get(index); }
}
```

6) Decorator.java

```java
package decorator.e2;

public abstract class Decorator extends Figure {
    Figure figure;

    public Decorator(Figure figure) {
        this.figure = figure;
        this.x = figure.getX(); // decorator는 figure 위치에 생성된다
        this.y = figure.getY();
    }

    @Override
    public void move() {
        x += 10; // 먼저 decorator가 이동하고
        y += 10;
        figure.move(); // 그 다음 figure도 이동한다
    }
}
```

7) ShadowDecorator.java

```java
package decorator.e2;

public class ShadowDecorator extends Decorator {

    public ShadowDecorator(Figure figure) {
        super(figure);
    }

    @Override
    public void draw(int indent) {
        figure.draw(indent); // 먼저 도형을 그리고
        System.out.printf("%s Shadow %d %dWn", "  ".repeat(indent), x, y); // 그 다음 decorator를 그린다
    }

    @Override
    public Figure clone() throws CloneNotSupportedException {
        return new ShadowDecorator(figure.clone());
    }
}
```

8) HandleDecorator.java

```java
package decorator.e2;

public class HandleDecorator extends Decorator {

    public HandleDecorator(Figure figure) {
        super(figure);
    }

    @Override
    public void draw(int indent) {
        figure.draw(indent); // 먼저 도형을 그리고
        System.out.printf("%s Handle %d %dWn", "  ".repeat(indent), x, y); // 그 다음 decorator를 그린다
    }

    @Override
    public Figure clone() throws CloneNotSupportedException {
        return new HandleDecorator(figure.clone());
    }
}
```

9) Example2.java

```
1  package decorator.e2;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Scanner;
6
7  public class Example2 {
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(0);
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                 figures.set(index, new ShadowDecorator(figure));
52                 break;
53             case "handle":
54                 index = Integer.valueOf(a[1]);
55                 figure = figures.get(index);
56                 figures.set(index, new HandleDecorator(figure));
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 레이블Wn");
66         System.out.printf("  선     : line 레이블Wn");
67         System.out.printf("  이동   : move 번호Wn");
68         System.out.printf("  삭제   : remove 번호Wn");
```

```
69          System.out.printf("  복제    : duplicate 번호Wn");
70          System.out.printf("  그룹    : group 번호1, 번호2,...Wn");
71          System.out.printf("  그림자 : shadow <객체번호>Wn");
72          System.out.printf("  핸들    : handle <객체번호>Wn");
73          System.out.printf("  종료    : quitWn");
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)
 Shadow 10 10
 Handle 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)
 Shadow 20 20
 Handle 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)
 Shadow 20 20
 Handle 20 20
1: Rectangle(A, 20, 20)
 Shadow 20 20
 Handle 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)
   Shadow 20 20
   Handle 20 20
  Rectangle(A, 20, 20)
   Shadow 20 20
   Handle 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)
   Shadow 30 30
   Handle 30 30
  Rectangle(A, 30, 30)
   Shadow 30 30
   Handle 30 30
)
```