HW4

1분반

2024년 10월 09일

32211792

박재홍

IDoll 인터페이스)

```java
public interface IDoll {
    String describe(); // 인형에 대한 설명을 제공하는 메소드

}

```

인형에 대한 설명을 제공하는 메소드를 선언해주었다.

Doll 클래스)

```java
import javax.swing.*;
import java.awt.*;

public abstract class Doll extends JPanel implements IDoll { // Doll 클래스는 JPanel을 상속받고 IDoll을 구현

    protected Image image;  // 인형의 이미지를 저장하기 위한 변수

    public Doll() { // JPanel 의 기본 크기 설정
        this.setPreferredSize(new Dimension(width:200,height:200)); // JPanel 의 기본크기를 200,200 으로 설정
    }

    @Override
    public void paintComponent(Graphics g) { // 인형을 그리기 위한 메소드
        if(image != null){
            super.paintComponent(g); // 부모 클래스의 paintComponent 호출
            g.drawImage(image,x:0, y:0, this); // 이미지를 좌표(0,0)에 그린다.
        }
    }

    @Override
    public abstract String describe(); // 인형에 대한 설명을 추가하는 메소드

}

```

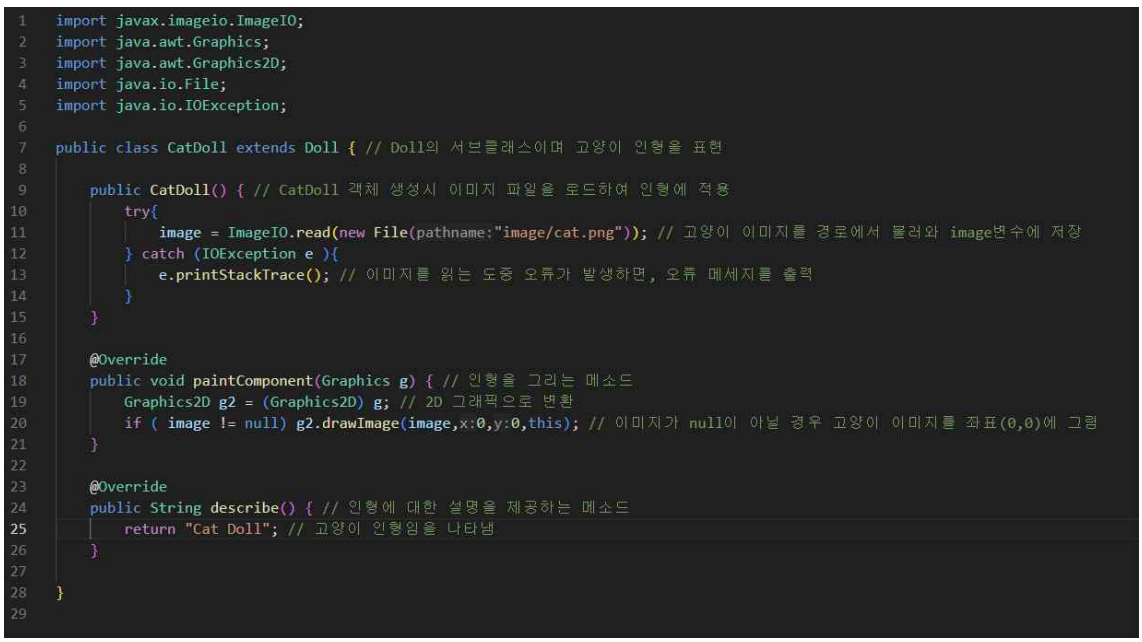
Doll 클래스는 JPanel을 상속받고 IDoll을 구현하는 추상 클래스이고 인형의 이미지를 저장하기 위해 변수 image를 선언해주었고 Doll 메소드를 통해 Jpanel의 기본크기를 200,200 으로 설정해주었다. 그 후 아래에 paintComponent 메소드를 오버라이드 하여 이미지를 좌표 (0,0) 에 그릴 수 있도록 하게 해주었다.

DollDecorator 클래스)

```java
import java.awt.*;

public abstract class DollDecorator extends Doll{ // Doll 클래스를 상속받아 데코레이터 역할을 수행하는 추상 클래스

    protected Doll decorateDoll; // 데코레이터 대상이 되는 Doll 객체를 저장하는 변수

    public DollDecorator(Doll decorateDoll){ // 데코레이터 대상인 인형을 받아서 초기화
        this.decorateDoll = decorateDoll;
    }

    @Override
    public void paintComponent(Graphics g){ // 인형을 그리기 위한 메소드
        super.paintComponent(g); // 부모 클래스의 paintComponent 호출
        decorateDoll.paintComponent(g); // 데코레이터 대상 인형의 paintComponent 호출
    }

    @Override
    public String describe(){ // 인형에 대한 설명을 제공하는 메소드
        return decorateDoll.describe();
    }

}

```

DollDecorator 클래스는 Doll 클래스를 상속받아서 데코레이터 역할을 수행해주는 추상클래스이다. DollDecorator 객체를 생성해 데코레이터 대상인 인형을 받아 초기화 시켜주었고 paintComponent 메소드를 오버라이드하여 인형을 그리기 위한 메소드를 선언해주었다. 그 후 인형에 대한 설명을 제공하는 메소드인 describe 메소드도 오버라이드 해주었다.

CatDoll 클래스)



```java
import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class CatDoll extends Doll { // Doll의 서브클래스이며 고양이 인형을 표현

    public CatDoll() { // CatDoll 객체 생성시 이미지 파일을 로드하여 인형에 적용
        try{
            image = ImageIO.read(new File(pathname:"image/cat.png")); // 고양이 이미지를 경로에서 불러와 image변수에 저장
        } catch (IOException e ){
            e.printStackTrace(); // 이미지를 읽는 도중 오류가 발생하면, 오류 메세지를 출력
        }
    }

    @Override
    public void paintComponent(Graphics g) { // 인형을 그리는 메소드
        Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환
        if ( image != null) g2.drawImage(image,x:0,y:0,this); // 이미지가 null이 아닐 경우 고양이 이미지를 좌표(0,0)에 그림
    }

    @Override
    public String describe() { // 인형에 대한 설명을 제공하는 메소드
        return "Cat Doll"; // 고양이 인형임을 나타냄
    }

}
```

CatDoll 클래스는 Doll 클래스의 서브 클래스이며 고양이 인형을 표현한다. CatDoll 객체를 생성하여 try-catch문을 통해 고양이 이미지를 경로에서 읽어와 image 변수에 저장할 수 있게 하고 읽는 도중 오류 발생시 오류메세지를 출력 할 수 있도록 해주었다. 그 후 인형을 그리는 메소드인 paintComponent를 오버라이드 하여 2D 그래픽으로 변환 해 준 후 image가 null 이 아니라면 좌표(0,0)에 고양이 이미지를 그릴 수 있게 해주었고 describe 메소드를 통해 고양이 인형임을 설명해주었다.

HatDecorator 클래스)

```java
import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class HatDecorator extends DollDecorator { // DollDecorator의 서브클래스이며 인형에 모자를 표현

public HatDecorator(Doll decoratorDoll){ // // 인형에 모자를 추가하는 역할을 함
    super(decoratorDoll);
    try {
        image = ImageIO.read(new File(pathname:"image/hat.png")); // hat이미지를 읽어와 image 변수에 저장
    } catch (IOException e){
        e.printStackTrace(); // 이미지 파일을 읽는 도중 오류가 발생하면 오류 메세지 출력
    }

}

@Override
public String describe() { // 인형에 대한 설명을 제공하는 메소드
    return decorateDoll.describe() + "with a hat";
}


@Override
public void paintComponent(Graphics g) { // 인형을 그리는 메소드
    Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환
    if ( image != null) g2.drawImage(image,-50,-150,this); // image가 null이 아니라면 좌표(-50,-150)에 그림
}
}

```

BallDecorator 클래스)

```java
import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class BallDecorator extends DollDecorator { // DollDecorator 의 서브클래스이며 공을 들고있는 인형을 표현

    public BallDecorator(Doll decoratorDoll){ // 공을 들고있는 인형을 꾸미기 위한 객체 생성, decoratorDoll을 파라미터로 받아 상위클래스에 전달
        super(decoratorDoll);
        try{
            image = ImageIO.read(new File(pathname:"image/ball.png")); // 공 이미지 파일을 읽어서 image 필드에 저장
        } catch (IOException e){
            e.printStackTrace(); // 이미지 파일을 읽는 도중 오류가 발생할경우, 오류 메세지를 출력
        }
    }

    @Override
    public String describe() { // 인형에 대한 설명을 추가로 제공하는 메소드
        return decorateDoll.describe() + "with a ball";
    }


    @Override
    public void paintComponent(Graphics g) { // 인형을 그리는 메소드
        Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환
        if ( image != null) g2.drawImage(image,-50,y:550,this); // 좌표(-50,550)에 이미지를 그리도록 설정
    }
}

```

ToyDecorator 클래스)

```java
import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class ToyDecorator extends DollDecorator { // DollDecorator의 서브클래스이며 인형에 장난감을 표현

    public ToyDecorator(Doll decoratorDoll){  // 인형에 장난감을 추가하는 역할
        super(decoratorDoll);
        try{
            image = ImageIO.read(new File(pathname:"image/toy.png")); // toy이미지를 읽어와 image 변수에 저장
        } catch (IOException e){
            e.printStackTrace(); // 이미지 파일을 읽는 도중 오류가 발생하면 오류 메세지를 출력
        }
    }

    @Override
    public String describe() { // 인형에 대한 설명을 제공하는 메소드
        return decorateDoll.describe() + "with a toy";
    }


    @Override
    public void paintComponent(Graphics g) { // 인형을 그리는 메소드
        Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환
        if ( image != null) g2.drawImage(image,x:350,y:250,this); // image가 null이 아니라면 좌표(350,250)에 그림
    }
}
```

세 개의 클래스 모두 각 클래스에 맞게 생성자를 구현해 HatDecorator는 모자 사진을 가져오고, BallDecorator는 공 사진을 가져오고, ToyDecorator는 장난감 사진을 가져와 모두 image 변수에 저장해주었고 이미지 파일을 읽는 도중에 오류가 발생하면 오류 메시지를 출력할 수 있도록 해주었다. 그리고 세 개의 클래스 모두 인형에 대해 설명을 제공하는 메소드인 describe 메소드를 선언해 주었고, 인형을 그리는 메소드인 paintComponent 메소드를 통해 2D 그래픽으로 변환 해준후 image가 null이 아니라면 HatDecorator는 좌표(-50,-150)에 BallDecorator는 좌표(-50,550)에 ToyDecorator는 좌표(350,250)에 그리도록 선언해주었다.

Your Code)

```java
// Your Code

import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class DogDoll extends Doll { // Doll의 서브클래스이며 개 인형을 표현

    public DogDoll() { // DogDoll 객체 생성
        try{
            image = ImageIO.read(new File(pathname:"image/dog.png")); // 개 이미지를 경로에서 읽어와 image변수에 저장
        } catch (IOException e ){
            e.printStackTrace(); // 이미지 파일을 읽는 도중 오류가 발생하면 오류 메세지를 출력
        }
    }


    @Override
    public void paintComponent(Graphics g) { // 인형을 그리는 메소드
        Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환
        if ( image != null) g2.drawImage(image,x:1000,y:0,this); // image가 null이 아니라면 좌표(1000,0)에 개를 그림
    }


    @Override
    public String describe() { // 인형에 대한 설명을 제공하는 메소드
        return "Dog Doll";
    }


}
```

src / BalloonDecorator.java / ...

```java
// Your Code

import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class BalloonDecorator extends DollDecorator { // DollDecorator클래스의 서브 클래스이며 풍선을 표현함

    public BalloonDecorator(Doll decoratorDoll){ // 풍선 이미지를 가진 인형을 꾸며주기 위한 객체 생성
        super(decoratorDoll); // 상위 클래스의 생성자 호출
        try{
            image = ImageIO.read(new File(pathname:"image/balloon.png")); // 풍선 이미지를 읽어 image변수에 저장
        } catch (IOException e){
            e.printStackTrace(); // 이미지 파일을 읽는 도중 오류가 발생할 경우, 오류 메세지 출력
        }
    }

    @Override
    public String describe() { // 인형에 대한 설명을 추가하는 메소드
        return decorateDoll.describe() + "with a balloon";
    }


    @Override
    public void paintComponent(Graphics g) { // 인형을 그리는 메소드
        Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환하여 사용
        if ( image != null) g2.drawImage(image,x:1400,y:550,this); // 좌표(1400,550)에 풍선 이미지를 그림
    }
}
```

```java
// Your Code

import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class DogballDecorator extends DollDecorator { // DollDecorator 의 서브클래스이며 개의 공을 표현

    public DogballDecorator(Doll decoratorDoll){ // 개의 공을 가진 인형을 꾸며주기 위한 객체 생성
        super(decoratorDoll); // 상위 클래스의 생성자 호출
        try{
            image = ImageIO.read(new File(pathname:"image/ball.png")); // ball 이미지를 읽어 image 변수에 저장
        } catch (IOException e){
            e.printStackTrace(); // 이미지 파일을 읽는 도중 오류가 발생하면 오류 메세지를 출력
        }
    }

    @Override
    public String describe() { // 인형에 대한 설명을 제공하는 메소드
        return decorateDoll.describe() + "with a ball";
    }


    @Override
    public void paintComponent(Graphics g) { // 인형을 그리는 메소드
        Graphics2D g2 = (Graphics2D) g; // 2D 그래픽으로 변환
        if ( image != null) g2.drawImage(image, x:1000,y:550,this); // 이미지가 null이 아니라면 좌표(1000,550)에 공을 그림
    }
}
```

```java
// Your Code
import javax.imageio.ImageIO;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.io.File;
import java.io.IOException;

public class DogHatDecorator extends DollDecorator { // DollDecorator의 서브클래스이며 개 인형에 모자를 씌우는걸 표현

public DogHatDecorator(Doll decoratorDoll){ // 인형에 모자를 추가하는 기능을 제공하는 객체 생성
    super(decoratorDoll); // 상위 클래스의 생성자 호출
    try {
        image = ImageIO.read(new File(pathname:"image/hat.png")); // 모자 이미지를 읽어와 image 변수에 저장
    } catch (IOException e){
        e.printStackTrace(); // 이미지를 읽는 도중 오류가 발생하면 오류 메세지를 출력
    }

}

@Override
public String describe() { // 인형에 대한 설명을 제공하는 메소드
    return decorateDoll.describe() + "with a hat";
}


@Override
public void paintComponent(Graphics g) {  // 인형을 그리는 메소드
    Graphics2D g2 = (Graphics2D) g; // 2D 그래픽을 변환
    if ( image != null) g2.drawImage(image,x:950,-130,this); // image가 null이 아니라면 좌표(950,-130)에 모자를 그림
}
}
```

우선적으로 다른 Doll 로 강아지를 택해서 DogDoll 클래스를 만들었고 Doll 클래스를 상속받게 한 후 DogDoll 객체를 생성해 개 이미지를 경로에서 읽어와 image 변수에 저장해주었다. 그리고서 위에서 구현한 고양이와 한 창에 같이 나오게 하기 위해 좌표를 (950,-130)으로 설정해주었다. 그리고 다른 Decorator로 Balloon을 추가시켰는데 구현방법 자체는 위에 구현한 데코레이터와 동일하게 구현했고 좌표만 (1400,550) 으로 설정해주었다.

그리고 기존에 있던 BallDecorator과 HatDecorator를 가져와 DogBallDecorator과 DogHatDecorator를 DogDoll에 추가시켜주었다. DogBallDecorator는 좌표(1000,550)에, DogHatDecorator는 좌표 (950,-130)에 추가 시켜주었다.

Main Frame 클래스)

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class MainFrame extends JFrame implements ActionListener {
    JButton[] buttons = new JButton[7]; // 버튼 배열 (Your Code)
    Doll doll; // 현재 보여질 인형 (CatDoll 또는 데코레이션된 인형)
    Doll doll2; // DogDoll (Your Code)
    JPanel displayPanel;

    public MainFrame() {
        doll = new CatDoll(); // 기본 CatDoll 설정
        doll2 = new DogDoll(); // DogDoll 설정(Your Code)

        // 디스플레이 패널 설정
        displayPanel = new JPanel(new BorderLayout());
        displayPanel.add(doll); // 처음에 CatDoll을 추가
        this.add(displayPanel, BorderLayout.CENTER);

        // 버튼 패널 설정
        JPanel buttonPanel = new JPanel(new GridLayout(rows:1, cols:7)); // 7개의 버튼(Your Code)
        buttons[0] = new JButton(text:"Hat");
        buttons[1] = new JButton(text:"Ball");
        buttons[2] = new JButton(text:"Toy");
        buttons[3] = new JButton(text:"Dog"); //(Your Code)
        buttons[4] = new JButton(text:"Balloon"); //(Your Code)
        buttons[5] = new JButton(text:"Dog Hat"); //(Your Code)
        buttons[6] = new JButton(text:"Dog ball"); //(Your Code)


        // 각 버튼에 ActionListener 추가
        for (int i = 0; i < buttons.length; i++) {
            buttons[i].addActionListener(this);
            buttonPanel.add(buttons[i]);
        }

        this.add(buttonPanel, BorderLayout.SOUTH); // 버튼 패널을 프레임 아래에 추가
```

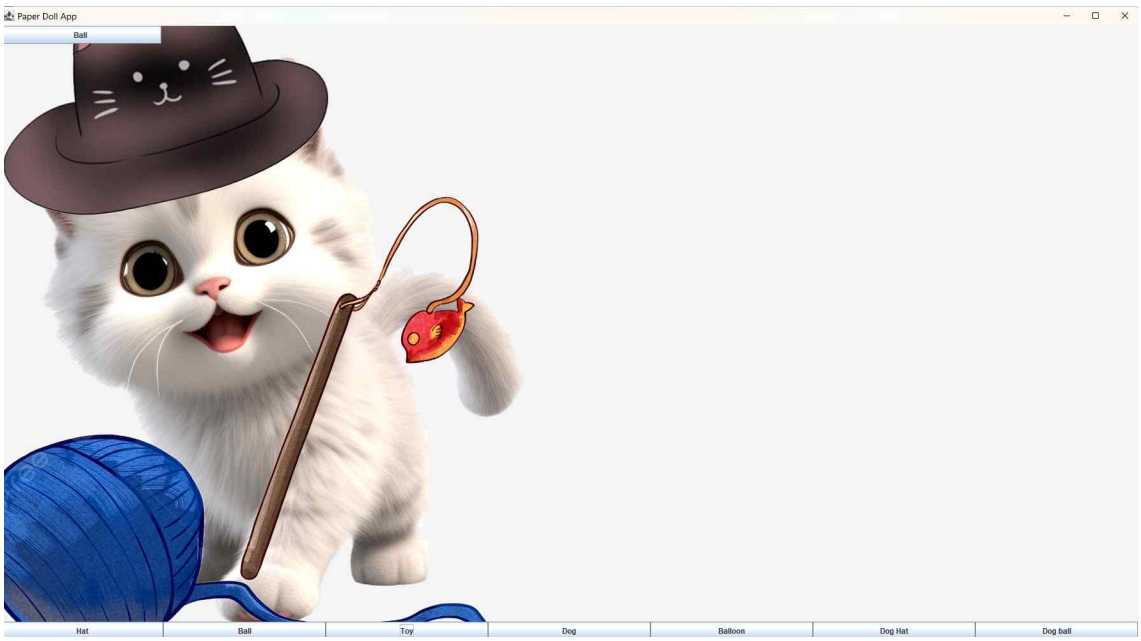
```java
        // 기본 프레임 설정
        this.setTitle(title:"Paper Doll App");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(width:1800, height:1000);
        this.setVisible(b:true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // 패널에서 기존 인형 제거
        displayPanel.remove(doll);
        displayPanel.remove(doll2); // (Your Code)

        // 어떤 버튼이 눌렸는지 확인하고 해당 인형 추가 (Your Code)
        if (e.getSource() == buttons[0]) {
            doll = new HatDecorator(doll); // CatDoll에 Hat 추가
            displayPanel.add(doll); // 데코레이션된 CatDoll 추가
        } else if (e.getSource() == buttons[1]) {
            doll = new BallDecorator(doll); // CatDoll에 Ball 추가
            displayPanel.add(doll); // 데코레이션된 CatDoll 추가
        } else if (e.getSource() == buttons[2]) {
            doll = new ToyDecorator(doll); // CatDoll에 Toy 추가
            displayPanel.add(doll); // 데코레이션된 CatDoll 추가
        } else if (e.getSource() == buttons[3]) {
            displayPanel.add(doll2); // DogDoll 추가
        }
          else if (e.getSource() == buttons[4]){
            doll = new BalloonDecorator(doll); // DogDoll에 Balloon 추가
            displayPanel.add(doll); // 데코레이션된 DogDoll 추가
        }
          else if (e.getSource() == buttons[5]){
            doll = new DogHatDecorator(doll);
            displayPanel.add(doll); // 데코레이션된 DogDoll 추가
        }
          else if (e.getSource() == buttons[6]){
            doll = new DogballDecorator(doll);
            displayPanel.add(doll); // 데코레이션된 DogDoll 추가
          }


        // 패널 갱신
        displayPanel.revalidate(); // 레이아웃 갱신
        displayPanel.repaint(); // 화면 다시 그리기
    }
}
```

MainFrame클래스에서는 우선적으로 JPanel를 상속받고 ActionListener를 구현해주었고 그 후 버튼배열을 선언해주었다. 그리고 현재 보여질 인형인 doll과 버튼을 눌렀을 때 강아지가 나올 수 있도록 doll2를 먼저 선언해주었고 메인프레임 메소드 안에서 catdoll과 dogdoll를 설정해주었고 처음에 catdoll이 나올 수 있도록 기본프레임을 설정해주었다. 그 후 7개의 버튼배열에 가로로 7개가 나오도록 버튼패널을 설정해주었고 1번부터 7번까지 Hat, Ball, Toy, Dog, Balloon, Doghat, Dogball로 설정해주었다. 그리고 this.add(buttonPanel, BorderLayout.SOUTH)를 통해 버튼패널을 프레임 맨 아래에 오도록 설정해주었고 한 프레임에 고양이와 강아지가 동시에 나오게 하기 위해 사이즈를 1800 x 1000 으로 설정해주었다. 그리고 if문을 통해 buttons[0] 번이 눌렸을 때 Hat을 추가, buttons[1] 번을 눌렀을 때 Ball 추가, buttons[2] 번을 눌렀을 때 Toy 추가, buttons[3]번을 눌렀을 때 DogDoll 추가, buttons[4]번을 눌렀을 때 Balloon추가, buttons[5]번을 눌렀을 때 강아지쪽에 Doghat추가, buttons[6] 번을 눌렀을 때 DogBall이 추가되도록 해주었다.

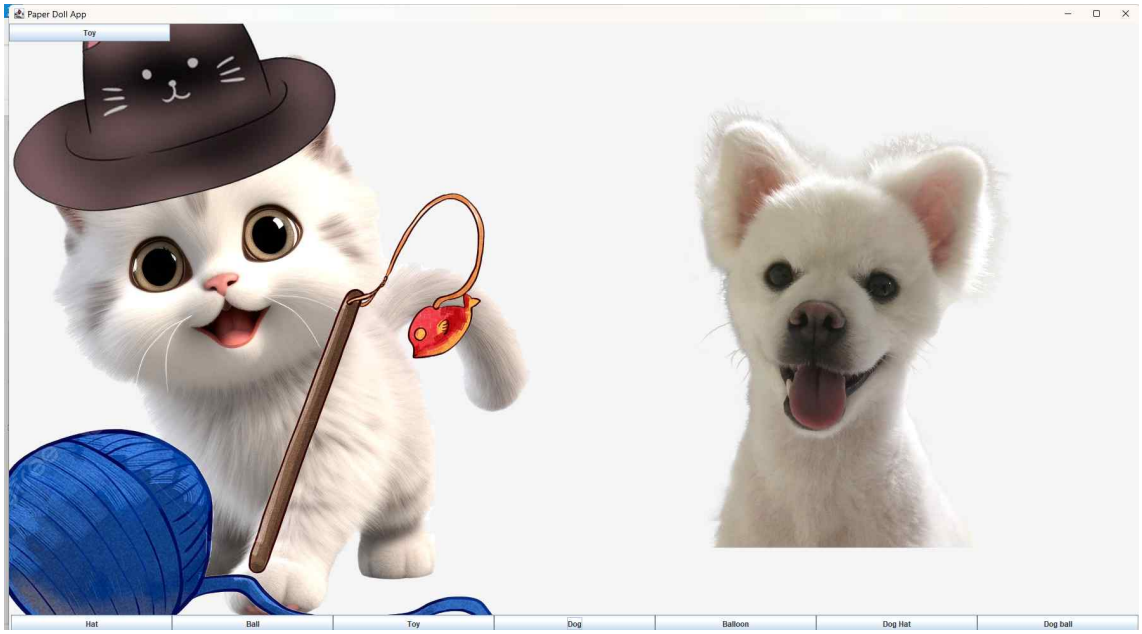
실행결과)

기본프레임)

Cat에 대한 Decorator만 눌렀을 때)



Dog를 눌렀을 때)



Dog Decorator들을 눌렀을 때)