vat 그림판

-unit test

양승민 200911400
정세진 200911418
한종철 200911429

1. test report

1. <testcasename="testSetlight" classname="SpecialEffectTest" time="0.0" />		
1-1. this.assertEquals(1,special.getlight())	1사분면 좌표 넣은 후 테스트	pass
1-2. this.assertEquals(2,special.getlight())	2사분면 좌표 넣은 후 테스트	
1-3. this.assertEquals(4,special.getlight())	3사분면 좌표 넣은 후 테스트	
1-4. this.assertEquals(3,special.getlight())	4사분면 좌표 넣은 후 테스트	

	input	expect output	output
1-1	50, 50(좌표), 500, 600(축)	1	1
1-2	600,50(좌표) 500,600(넓이)	2	2
1-3	700,800(좌표) 500,600(넓이)	3	3
1-4	50,800(좌표) 500,600(넓이)	4	4

2. <testcasename="testExcutedecal" classname="SpecialEffectTest" time="0.0" />		
this.assertNotNull(list.get(1))	데칼 기능 실행 후 리턴되는 리스트 테스트	pass

3. <testcasename="testExcuteshadow" classname="SpecialEffectTest" time="0.0" />		
this.assertEquals(list.get(0).getsx(),1.0)	그림자 기능 실행 전 리스트 set 테스트	pass
this.assertNotNull(list.get(0))	그림자 기능 실행 후 생성되어 리턴되는 리스트 테스트	
this.assertNotNull(list.get(1))	그림자 기능 실행 후 생성되어 리턴되는 리스트 테스트2	

4. <testcasename="testMakeShdow" classname="SpecialEffectTest" time="0.811" />		
assertEquals(sc.state,"shadow");	그림자 버튼 클릭후 바뀌는 상태 테스트	pass

	input	expect output	output
4.	"shadow"	"shadow"	"shadow"

5. <testcasename="testExcuteDottedLine" classname="SpecialEffectTest" time="0.0" />		
this.assertNotNull(special.excuteDottedLine())	점선 기능 실행 후 생성되어 리턴되는 리스트 테스트	pass

6. <testcasename="testGetxy" classname="SpecialEffectTest" time="0.0" />		
assertEquals(sc.X[0],1.0);	screen class sc에 값 넣는 메소드 테스트	pass

7. <testcasename="testResetxy" classname="SpecialEffectTest" time="0.0" />		
assertEquals(sc.X[0], 0.0);	screen class의 값 초기화 하는 메소드 테스트	pass

	input	expect output	output
7.	none	0	0

8. <testcasename="testAddlist" classname="SpecialEffectTest" time="0.0" />		
8-1. assertEquals(list.get(0).getsx(),5.0);	addlist 리스트 추가 결과 test 1	pass
8-2. assertEquals(list.get(0).getey(),6.0);	addlist 리스트 추가 결과 test 2	
8-3. assertEquals(list.get(1).getsx(),6.0);	addlist 리스트 추가 결과 test 3	

	input	expect output	output
8-1	addlist(5, 5, 6, 6, "pencil")	5	5
8-2	addlist(5, 5, 6, 6, "pencil")	6	6
8-3	addlist(6, 6, 7, 7, "brush")	6	6

9. <testcasename="testSetcolor" classname="SpecialEffectTest" time="0.0" />		
assertEquals(Color.black ,drawtool.getcolor());	색 검은색 지정후 세팅 결과 테스트	pass
assertEquals(Color.white ,drawtool.getcolor());	색 흰색 지정후 세팅 결과 테스트	

1. test code

```
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import java.awt.Color;
import java.util.ArrayList;
import org.junit.Test;
import junit.framework.TestCase;

public class SpecialEffectTest extends TestCase {
	SpecialEffect special = new SpecialEffect();
	DrawTool drawtool = new DrawTool(null);
	Screen sc = new Screen();
	public SpecialEffectTest(String name) {
		super(name);
	}
	protected void setUp() throws Exception {
		super.setUp();
	}
	protected void tearDown() throws Exception {
		super.tearDown();
	}
```

1.

```
	@Test
	public void testSetlight() {
		special.setlight(50, 50, 500, 600);
		this.assertEquals(1,special.getlight());
		special.setlight(600, 50, 500, 600);
		this.assertEquals(2,special.getlight());

		special.setlight(700, 800, 500, 600);
		this.assertEquals(4,special.getlight());
		special.setlight(50, 800, 500, 600);
		this.assertEquals(3,special.getlight());
	}
```

2.

```
	@Test
	public void testExcutedecal() {
```

```
        ArrayList<DrawShape> list = new ArrayList<DrawShape>();
        list.add(new DrawLine(1,1,1,1,Color.black,1));
        special.setlist(list);
        //list = special.excutedecal(500);
        this.assertNotNull(list.get(1));
    }
```

3.

```
    @Test
    public void testExcuteshadow() {
        ArrayList<DrawShape> list = new ArrayList<DrawShape>();
        list.add(new DrawLine(1,1,1,1,Color.black,1));
        special.setlist(list);
        list = special.excuteshadow();
        this.assertEquals(list.get(0).getsx(), 1.0);
        this.assertNotNull(list.get(0));
        this.assertNotNull(list.get(1));
    }
```

4.

```
    @Test
    public void testMakeShdow() {
        sc.makeshadow();
        assertEquals(sc.state,"shadow");
    }
```

5.

```
    @Test
    public void testExcuteDottedLine() {
        this.assertNotNull(special.excuteDottedLine());
    }
```

6.

```
    @Test
    public void testGetxy() {
        sc.getxy(1, 1);
        assertEquals(sc.X[0],1.0);
    }
```

7.

```
    @Test
    public void testResetxy() {
        for(int i = 0;i<10;i++) {
                sc.getxy(i+1, i+1);
```

```
            }
            sc.resetxy();
            assertEquals(sc.X[0], 0.0);
        }
```

8.

```
        @Test
        public void testAddlist() {
            drawtool.addlist(5, 5, 6, 6, "pencil");
            ArrayList<DrawShape> list = drawtool.getlist();
            assertEquals(list.get(0).getsx(),5.0);
            assertEquals(list.get(0).getey(),6.0);
            drawtool.addlist(6, 6, 7, 7, "brush");
            assertEquals(list.get(1).getsx(),6.0);
        }
```

9.

```
        @Test
        public void testSetcolor() {
            drawtool.setcolor(Color.black);
            assertEquals(Color.black ,drawtool.getcolor());
            drawtool.setcolor(Color.white);
            assertEquals(Color.white ,drawtool.getcolor());
        }
}
```