

# 제어 구조 2. 반복

Control Structure : Iteration

도경구

한양대학교 ERICA 소프트웨어학부

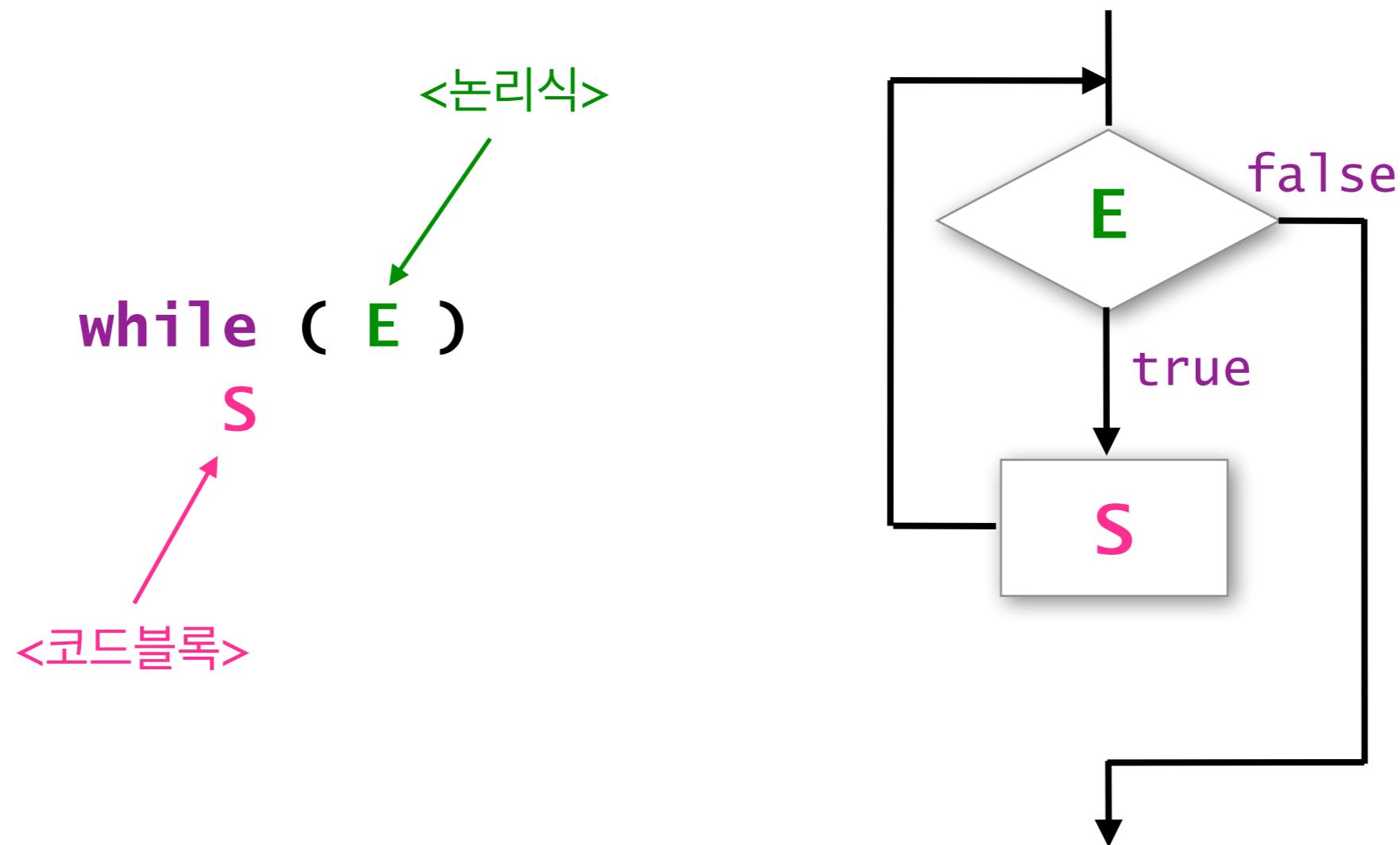


# 반복 구조

동일 코드의 반복 실행을 표현하는 구조

- 반복 횟수 고정 (Definitive Iteration)
- 반복 횟수 사전 예측 불가 (Indefinite Iteration)
- 무한 반복 (Unbounded Iteration) - 끝없이 반복 (diverge)

# while 루프



# 반복 횟수 고정

## 패턴

```
int i = INITIAL_VALUE;  
while (TEST_ON_i) {  
    // 필요한 계산 수행  
    // i 값 증가  
}
```

```
int i = INITIAL_VALUE;  
while (TEST_ON_i) {  
    // i 값 감소  
    // 필요한 계산 수행  
}
```

# 반복 횟수 고정

## 사례 - 시험 점수 평균 구하기

```

1 import javax.swing.*;
2
3 public class ClassManagement {
4
5     /* computeAverage - 제출한 시험 점수 평균 계산
6      * @param how_many - 시험 점수의 개수 (양수)
7      * @return - 평균 점수 */
8     public double computerAverage(int how_many) {
9         double total_points = 0.0;
10        int count = 0;
11        while (count < how_many) {
12            // loop invariant : total_points == exam_1 + ... + exam_count
13            String input = JOptionPane.showInputDialog("다음 시험 점수?");
14            int score = Integer.parseInt(input);
15            total_points = total_points + score;
16            count = count + 1;
17            System.out.println("개수 = " + count + ", 누적 점수 = " + total_points);
18        }
19        return total_points / how_many;
20    }
21
22    public static void main(String[] args) {
23        ClassManagement loop = new ClassManagement();
24        System.out.println("평균 점수 = " + loop.computerAverage(5));
25    }
26 }
```

## 반복 횟수 고정

### 사례 학습 - 양궁 과녁 그리기



# 반복 횟수 사전 예측 불가

## 입력 값에 종속

```
boolean processing = true;
while (processing) {
    // 입력을 받음
    if /* 입력 종료 신호를 받았음 */ )
        processing = false;
    else {
        // 입력을 처리
    }
}
```

# 반복 횟수 사전 예측 불가

## 사례 - 시험 점수 평균 구하기

```

1 import javax.swing.*;
2
3 public class ClassManagement {
4
5     /* computeAverage - 제출한 시험 점수 평균 계산
6      * @return - 평균 점수
7      * @throw RuntimeException - Cancel 버튼을 누른 즉시 */
8     public double computerAverage() {
9         double total_points = 0.0;
10        int count = 0;
11        boolean processing = true;
12        while (processing) {
13            // loop invariant : total_points == exam_1 + ... + exam_count
14            String message = "다음 시험 점수? (입력 완료시 Cancel 버튼 누름)";
15            String input = JOptionPane.showInputDialog(message);
16            if (input == null) // Cancel 버튼을 눌렀음
17                processing = false;
18            else {
19                int score = Integer.parseInt(input);
20                total_points = total_points + score;
21                count = count + 1;
22            }
23            System.out.println("개수 = " + count + ", 누적 점수 = " + total_points);
24        }
25        if (count == 0)
26            throw new RuntimeException("computeAverage error: 계산할 점수가 없습니다.");
27        return total_points / count;
28    }
29
30    public static void main(String[] args) {
31        ClassManagement loop = new ClassManagement();
32        System.out.println("평균 점수 = " + loop.computerAverage());
33    }
34 }
```

# 반복 횟수 사전 예측 불가

## 검색 결과에 종속

```
int index = 0;
boolean found = false;
while (!found && index < s.length()) {
    if (s.charAt(index) == c)
        found = true;
    else
        index = index + 1;
}
```

# 반복 횟수 사전 예측 불가

# 문자열 검색

```

1 public class TextSearch {
2
3     /* findChar - 문자열에서 가장 먼저 나타나는 문자를 검색
4      * @param c - 검색할 문자
5      * @param s - 검색 대상 문자열
6      * @return - s에서 가장 먼저 나타나는 문자 c의 인덱스; 없으면 -1 */
7     public int findChar(char c, String s) {
8         boolean found = false;
9         int index = 0;
10        while (!found && index < s.length()) {
11            // loop invariant:
12            // (1) found == false : s[0], ..., s[index-1]은 모두 c가 아님
13            // (2) found == true : s.charAt(index) == c
14            if (s.charAt(index) == c)
15                found = true;
16            else
17                index = index + 1;
18        }
19        if (!found)
20            index = -1;
21        return index;
22    }
23
24    public static void main(String[] args) {
25        TextSearch text_search = new TextSearch();
26        System.out.println(text_search.findChar('a', "Hanyang"));
27        System.out.println(text_search.findChar('e', "Hanyang"));
28    }
29 }
```

# for 루프

```
int i = INITIAL_VALUE;  
while (TEST_ON_i) {  
    // 필요한 계산 수행  
    // i 값 증가  
}
```

```
for (int i = INITIAL_VALUE; TEST_ON_i; i = i + c) {  
    // 필요한 계산 수행  
}
```

# for 루프

```

public int findChar(char c, String s) {
    boolean found = false;
    int index = 0;
    while (!found && index < s.length()) {
        // loop invariant:
        // (1) found == false : s[0], ..., s[index-1]은 모두 c가 아님
        // (2) found == true : s.charAt(index) == c
        if (s.charAt(index) == c)
            found = true;
        else
            index = index + 1;
    }
    if (!found)
        index = -1;
    return index;
}

```

```

public int findChar(char c, String s) {
    int index;
    for (index = 0; index < s.length() && s.charAt(index) != c; index++);
    // loop invariant: s[0], ..., s[index-1]은 모두 c가 아님
    if (index == s.length())
        index = -1;
    return index;
}

```

# 중첩 루프

## Nested Loop

1x1=1 1x2=2 1x3=3 1x4=4 1x5=5 1x6=6 1x7=7 1x8=8 1x9=9  
 2x1=2 2x2=4 2x3=6 2x4=8 2x5=10 2x6=12 2x7=14 2x8=16 2x9=18  
 3x1=3 3x2=6 3x3=9 3x4=12 3x5=15 3x6=18 3x7=21 3x8=24 3x9=27  
 4x1=4 4x2=8 4x3=12 4x4=16 4x5=20 4x6=24 4x7=28 4x8=32 4x9=36  
 5x1=5 5x2=10 5x3=15 5x4=20 5x5=25 5x6=30 5x7=35 5x8=40 5x9=45  
 6x1=6 6x2=12 6x3=18 6x4=24 6x5=30 6x6=36 6x7=42 6x8=48 6x9=54  
 7x1=7 7x2=14 7x3=21 7x4=28 7x5=35 7x6=42 7x7=49 7x8=56 7x9=63  
 8x1=8 8x2=16 8x3=24 8x4=32 8x5=40 8x6=48 8x7=56 8x8=64 8x9=72  
 9x1=9 9x2=18 9x3=27 9x4=36 9x5=45 9x6=54 9x7=63 9x8=72 9x9=81

```

for (int i = 1; i < 10; i++) {
    // loop invariant: 구구단의 i-1 단까지 출력했음
    for (int j = 1; j < 10; j++) {
        // loop invariant: 구구단의 i-1 단까지 출력하고, i단의 j항까지 출력했음
        System.out.print(i + "x" + j + "=" + (i * j) + " ");
    }
    System.out.println();
}
  
```

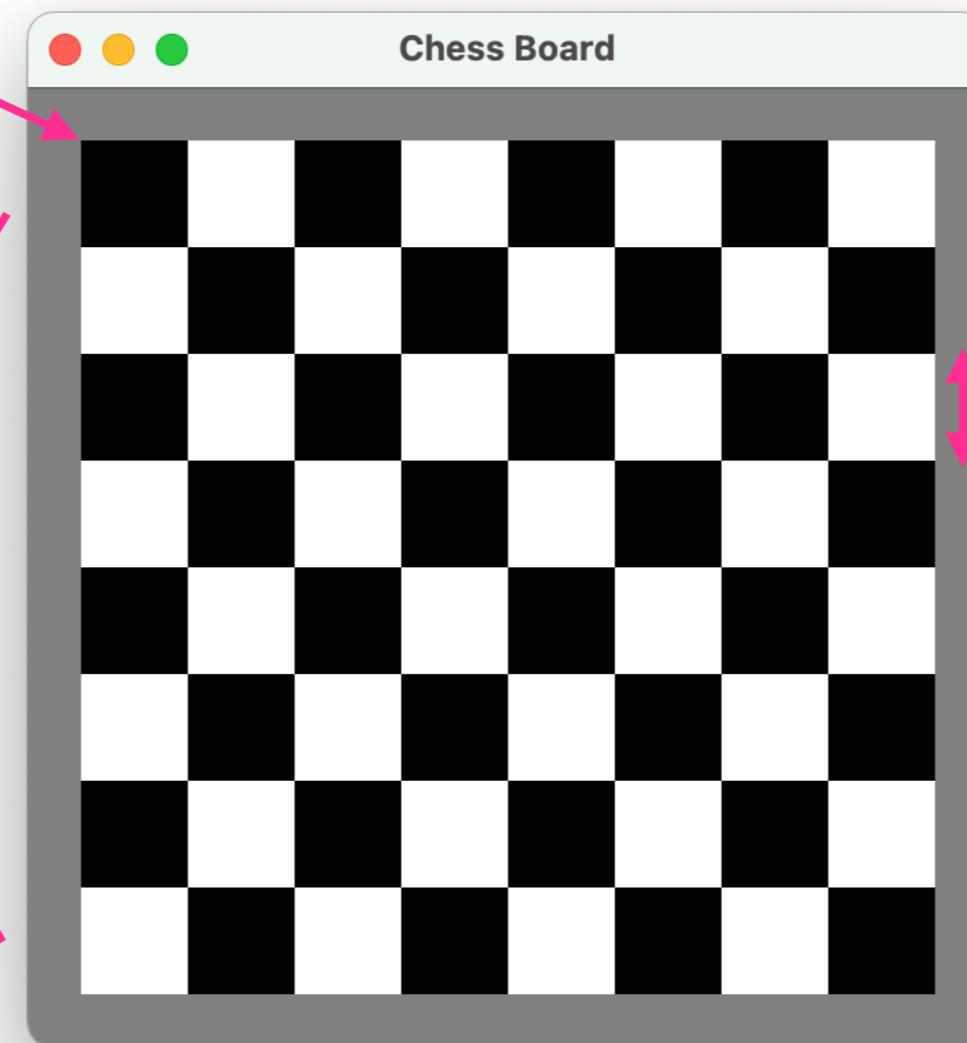
## 중첩 루프

## 체스 보드 그리기

(**start\_x**, **start\_y**)

**number\_of\_rows**

**square\_size**



```
public static void main(String[] args) {  
    new ChessBoardWriter(8, 40);  
}
```

## 중첩 루프

```

1① import javax.swing.*;
2 import java.awt.*;
3
4 /* ChessBoardWriter - 패널에 과녁을 그림 */
5 public class ChessBoardWriter extends JPanel {
6     private int number_of_rows;
7     private int square_size;
8     private int panel_width;
9     private int offset = 20;
10
11②     /* Constructor ChessBoardWriter - 패널을 만들고 프레임을 써움
12      * @param rows - 각 열별 칸의 갯수
13      * @param size - 한 칸의 길이 */
14③     public ChessBoardWriter(int rows, int size) {
15         number_of_rows = rows;
16         square_size = size;
17         panel_width = number_of_rows * square_size + 2 * offset;
18
19         JFrame f = new JFrame();
20         f.getContentPane().add(this);
21         f.setTitle("Chess Board");
22         f.setSize(panel_width, panel_width + 28);
23         f.setVisible(true);
24         f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
25     }
26

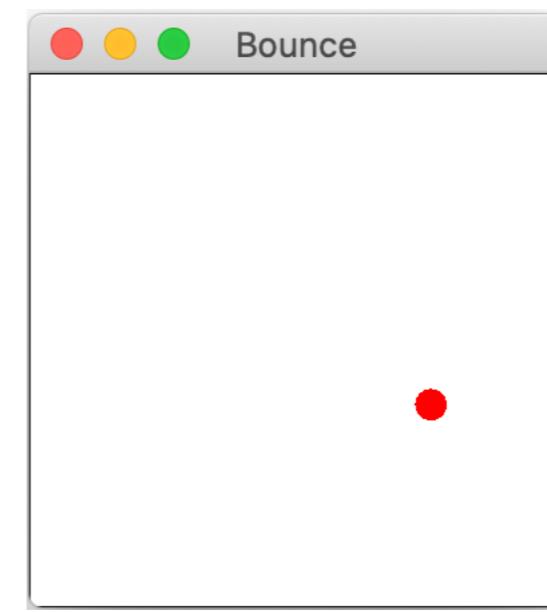
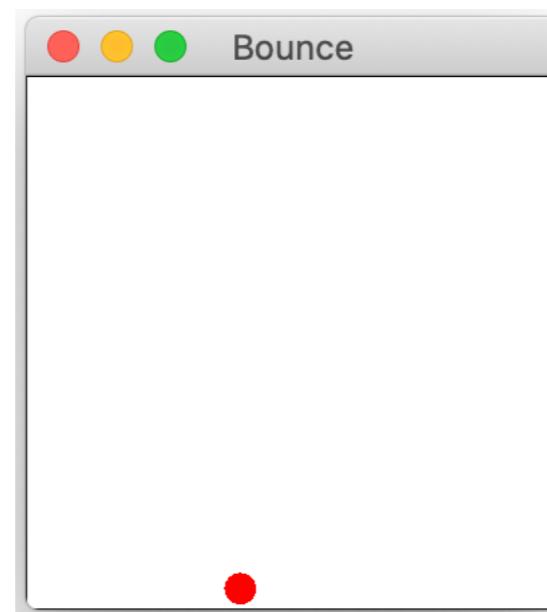
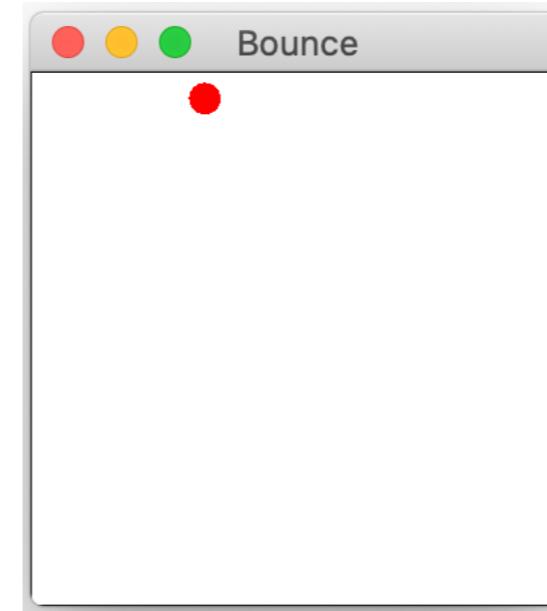
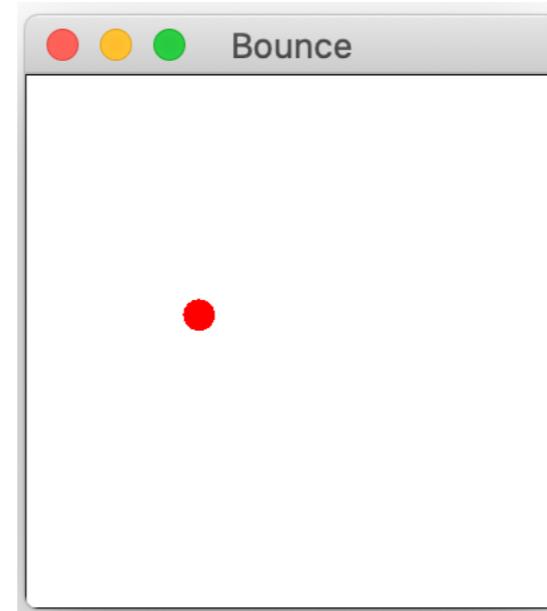
```

```

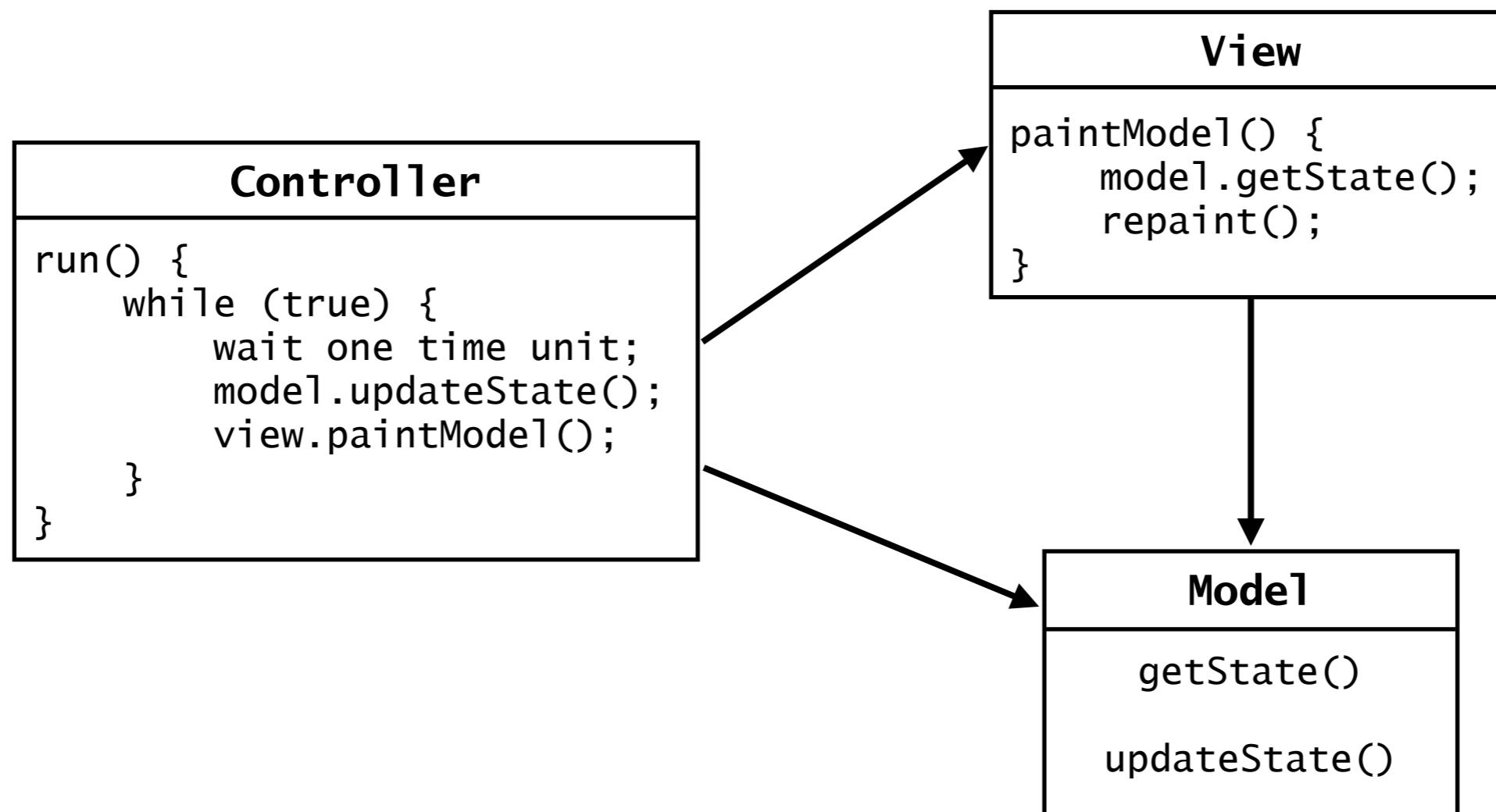
27①     /* paintComponent - 패널에 그림을 그림
28      * @param g - 그래픽스 펜 */
29② public void paintComponent(Graphics g) {
30     g.setColor(Color.gray);
31     g.fillRect(0, 0, panel_width, panel_width);
32     paintBoard(offset, offset, number_of_rows, square_size, g);
33 }
34
35③     /* paintBoard - 체스보드를 그림
36      * @param start_x - 체스보드의 좌상단 구석의 x 좌표
37      * @param start_y - 체스보드의 좌상단 구석의 y 좌표
38      * @param rows - 체스보드 열의 갯수
39      * @param size - 체스보드 칸의 너비
40      * @param g - 그래픽스 펜 */
41④ private void paintBoard(int start_x, int start_y,
42                           int rows, int size, Graphics g) {
43     for (int x = 0; x < rows; x = x + 1) {
44         // loop invariant: x열까지 그렸음 (x 증가 전)
45         int x_position = start_x + x * size;
46         for (int y = 0; y < rows; y = y + 1) {
47             // loop invariant: x열의 y칸까지 그렸음 (x 증가 후, y 증가 전)
48             int y_position = start_y + y * size;
49             if ((x + y) % 2 == 0) // 빨간색 칠할 차례
50                 g.setColor(Color.black);
51             else
52                 g.setColor(Color.white);
53             g.fillRect(x_position, y_position, size, size);
54         }
55     }
56 }

```

# 사례 학습 - 상자 속의 공 애니메이션



# Arhitecture Pattern of a Simple Animation



# Model

class	MovingBall	2차원 상자에서 움직이는 공
field	int x_pos, int y_pos	공의 중심 x 좌표, 공의 중심 y 좌표
	radius	공의 반지름
	int x_velocity, int y_velocity	속도 x축, 속도 y축
method	move( <b>int</b> time_units)	time_unit 만큼 공을 이동, 벽에 부딪치면 방향을 바꿈

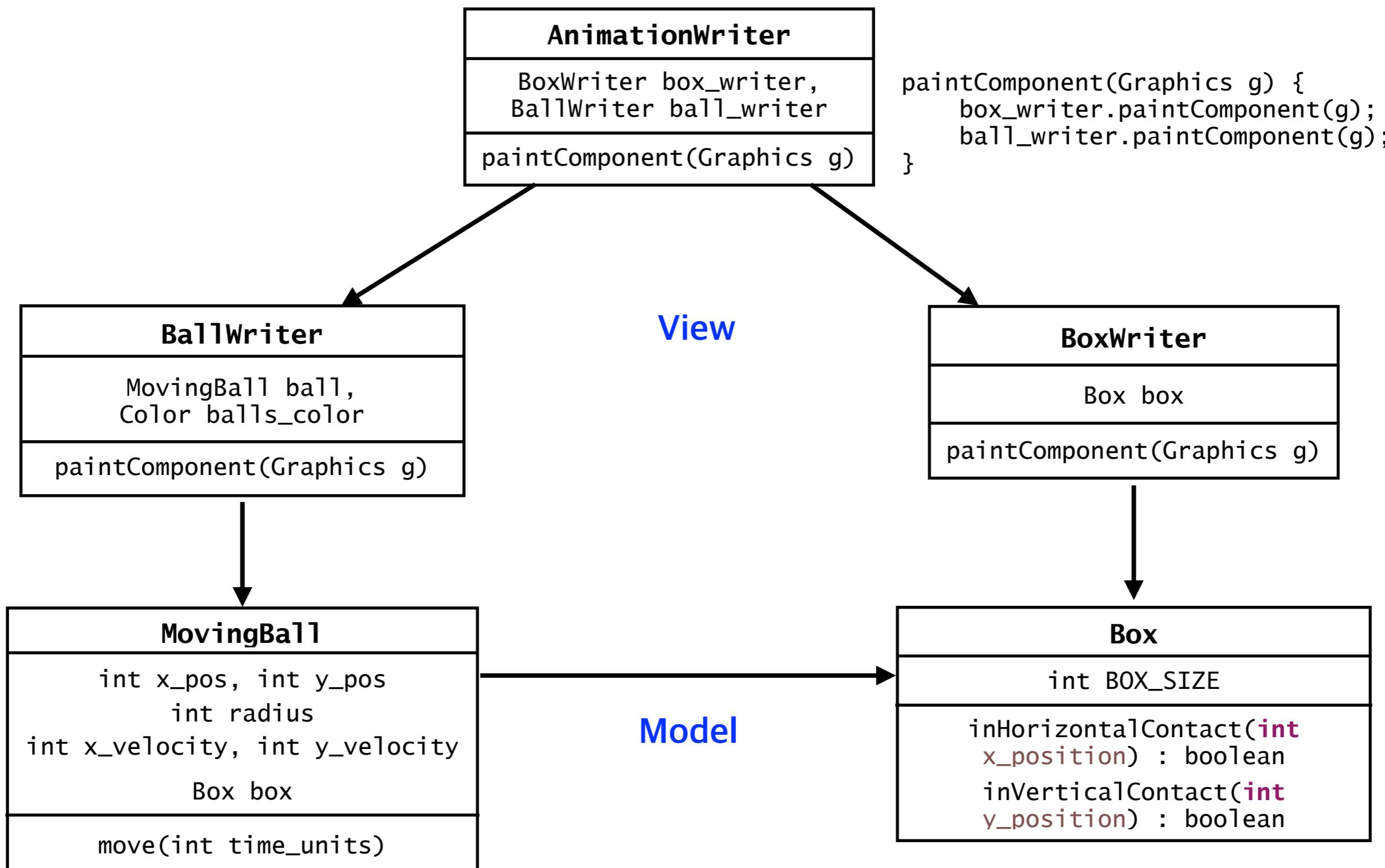
class	Box	공이 돌아다니는 상자
field	BOX_SIZE	상자의 크기
method	inHorizontalContact( <b>int</b> x_position) : boolean	공이 x축 방향으로 좌/우 벽에 도달 여부를 리턴
	inVerticalContact( <b>int</b> y_position) : boolean	공이 y축 방향으로 아래/위 벽에 도달 여부를 리턴

```
1  /** MovingBall - 2차원 상자에서 움직이는 공 */
2  public class MovingBall {
3      private int x_pos; // 공의 중심 x 좌표
4      private int y_pos; // 공의 중심 y 좌표
5      private int radius; // 공의 반지름
6
7      private int x_velocity = +5; // 속도 x축
8      private int y_velocity = +2; // 속도 y축
9
10     private Box container;
11
12     /**
13      * @param x_initial - 공의 중심 x 좌표
14      * @param y_initial - 공의 중심 y 좌표
15      * @param r - 공의 반지름
16      * @param box - 공이 살고 있는 상자 */
17     public MovingBall(int x_initial, int y_initial, int r, Box box) {
18         x_pos = x_initial;
19         y_pos = y_initial;
20         radius = r;
21         container = box;
22     }
23 }
```

```
24     /** xPosition - 공의 x축 위치 리턴 */
25     public int xPosition() {
26         return x_pos;
27     }
28
29     /** yPosition - 공의 y축 위치 리턴 */
30     public int yPosition() {
31         return y_pos;
32     }
33
34     /** radiusOf - 공의 반지름 리턴 */
35     public int radiusOf() {
36         return radius;
37     }
38
39     /** move - time_unit 만큼 공을 이동, 벽에 부딪치면 방향을 바꿈
40      * @param time_units - 프레임 사이의 시간 */
41     public void move(int time_units) {
42         x_pos = x_pos + x_velocity * time_units;
43         if (container.inHorizontalContact(x_pos))
44             x_velocity = - x_velocity;
45         y_pos = y_pos + y_velocity * time_units;
46         if (container.inVerticalContact(y_pos))
47             y_velocity = - y_velocity;
48     }
49 }
```

```
1  /** Box - 공이 돌아다니는 상자 */
2  public class Box {
3
4      private int BOX_SIZE; // 상자의 크기
5
6      /** Constructor Box - 상자 생성
7       * @param size - 상자의 크기 */
8      public Box(int size) {
9          BOX_SIZE = size;
10     }
11
12     /** inHorizontalContact - 공이 x축 방향으로 좌/우 벽에 도달 여부를 리턴
13      * @param x_position - 공의 x 좌표
14      * @return true, 공의 x 좌표가 좌우 벽의 x 좌표와 같거나 벗어났으면 true, 그렇지 않으면 false */
15     public boolean inHorizontalContact(int x_position) {
16         return (x_position <= 0 ) || (x_position >= BOX_SIZE);
17     }
18
19     /** inVerticalContact - 공이 y축 방향으로 아래/위 벽에 도달 여부를 리턴
20      * @param y_position - 공의 y 좌표
21      * @return true, 공의 y 좌표가 아래위 벽의 y 좌표와 같거나 벗어났으면 true, 그렇지 않으면 false */
22     public boolean inVerticalContact(int y_position) {
23         return (y_position <= 0 ) || (y_position >= BOX_SIZE);
24     }
25
26     /** sizeOf - 상자의 크기를 리턴 */
27     public int sizeOf() {
28         return BOX_SIZE;
29     }
30 }
```

```
1 public class TestModel {  
2  
3     public static void main(String[] args) {  
4         Box box = new Box(50);  
5         MovingBall ball = new MovingBall(25, 25, 10, box);  
6         while (true) {  
7             ball.move(1);  
8             System.out.println("x = " + ball.xPosition() + "; y = " + ball.yPosition());  
9         }  
10    }  
11 }
```



```

1 Ⓜ import java.awt.*;
2 import javax.swing.*;
3
4 /**
5  * AnimationWriter - 상자와 공의 애니메이션 디스플레이 */
6 public class AnimationWriter extends JPanel {
7     private BoxWriter box_writer; // 상자 그리는 객체
8     private BallWriter ball_writer; // 공 그리는 객체
9
10    /**
11     * @param b - 상자 그리는 객체
12     * @param l - 공 그리는 객체
13     * @param size - 프레임의 크기 */
14    public AnimationWriter(BoxWriter b, BallWriter l, int size) {
15        box_writer = b;
16        ball_writer = l;
17        JFrame f = new JFrame();
18        f.getContentPane().add(this);
19        f.setTitle("Bounce");
20        f.setSize(size, size+22);
21        f.setVisible(true);
22        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
23    }
24
25    /**
26     * paintComponent - 공과 상자 그리기
27     * @param g - 그래픽스 펜 */
28    public void paintComponent(Graphics g) {
29        box_writer.paintComponent(g);
30        ball_writer.paintComponent(g);
31    }
32 }

```

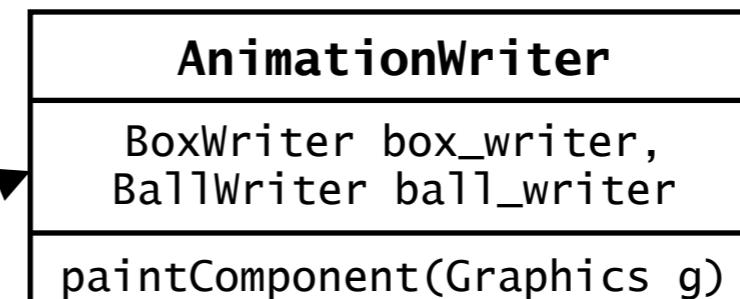
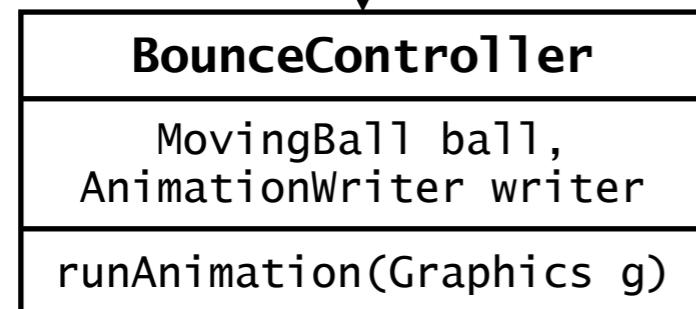
```
1 import java.awt.*;
2
3 /**
4  * BoxWriter - 상자를 그림 */
5 public class BoxWriter {
6
7     private Box box;    // 상자 객체
8
9     /**
10      * Constructor BoxWriter
11      * @param b - 상자 객체 */
12     public BoxWriter(Box b) {
13         box = b;
14
15     /**
16      * paint - 상자 그리기
17      * @param g - 그래픽스 펜 */
18     public void paintComponent(Graphics g) {
19         int size = box.sizeOf();
20         g.setColor(Color.white);
21         g.fillRect(0, 0, size, size);
22         g.setColor(Color.black);
23         g.drawRect(0, 0, size, size);
24     }
25 }
```

```
1 import java.awt.*;
2
3 /**
4  * BallWriter - 움직이는 공을 그림
5  */
6 public class BallWriter {
7
8
9     /**
10      * @param x - 공 객체
11      * @param c - 공의 색깔
12     */
13    public BallWriter(MovingBall x, Color c) {
14        ball = x;
15        balls_color = c;
16    }
17
18    /**
19     * @param g - 그래픽스 펜
20     */
21    public void paintComponent(Graphics g) {
22        g.setColor(balls_color);
23        int radius = ball.radiusOf();
24        g.fillOval(ball.xPosition() - radius, ball.yPosition() - radius,
25                    radius * 2, radius * 2);
26    }
27}
```

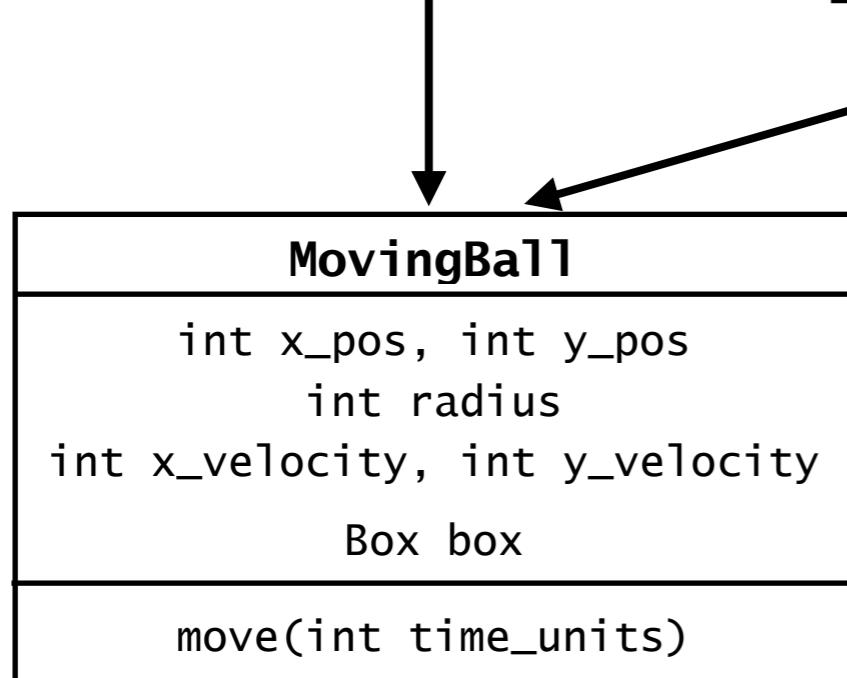
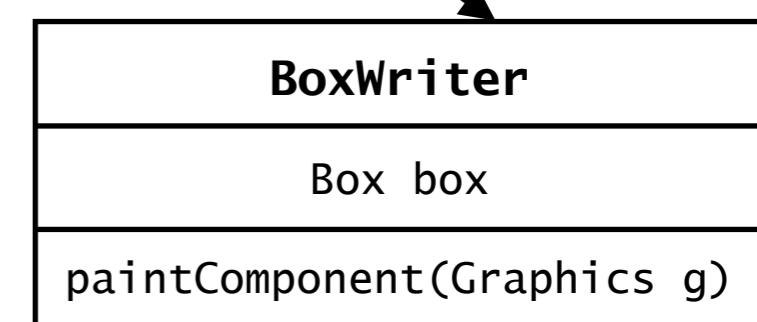
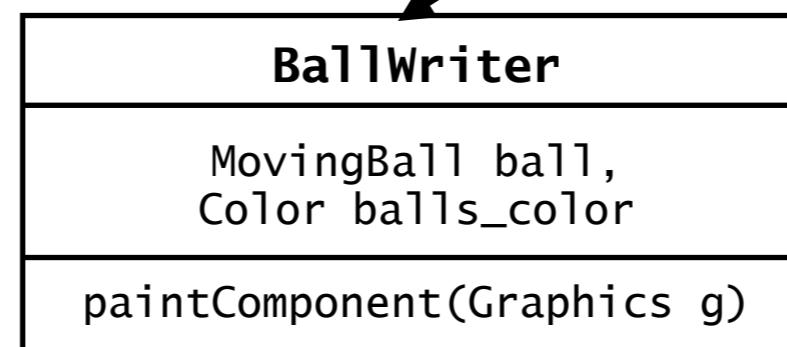
## Starter



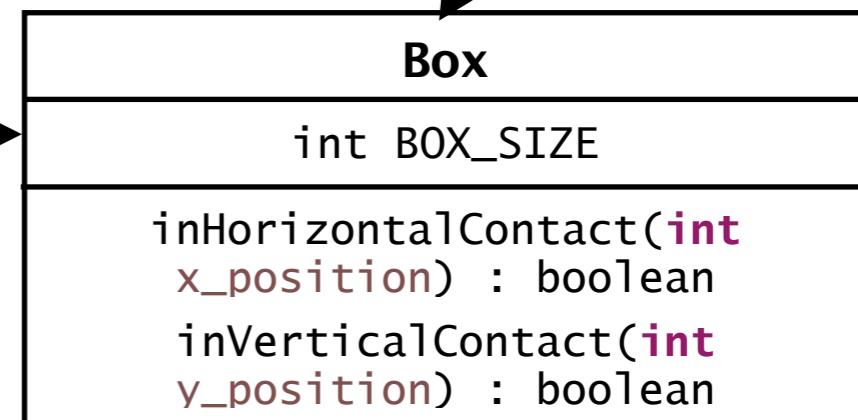
## Controller



View



Model



```

1  /** BounceController - 상자 안에서 움직이는 공 제어 */
2  public class BounceController {
3      private MovingBall ball; // 공 객체 (Model)
4      private AnimationWriter writer; // 애니메이션 객체 (Output-View)
5
6o   /** Constructor BounceController 컨트롤러 초기화
7     * @param b - 공 객체 (Model)
8     * @param w - 애니메이션 객체 (Output-View) */
9o   public BounceController(MovingBall b, AnimationWriter w) {
10    ball = b;
11    writer = w;
12}
13
14  /** runAnimation - 내부 시계를 활용하여 애니메이션 구동 */
15o  public void runAnimation() {
16      int time_unit = 1; // 애니메이션 스텝의 시간 단위
17      int painting_delay = 20; // 다시 그리기 사이의 지연 시간 간격
18      while (true) {
19          delay(painting_delay);
20          ball.move(time_unit);
21          System.out.println(ball.xPosition() + ", " + ball.yPosition());
22          writer.repaint();
23      }
24  }
25
26  /** delay - how_long millisecond 동안 실행 정지 */
27o  private void delay(int how_long) {
28      try { Thread.sleep(how_long); }
29      catch (InterruptedException e) { }
30  }
31 }

```

```
1 import java.awt.*;
2
3 /**
4  * BounceTheBall – 애니메이션 객체를 생성하고 공 운동 시작 */
5 public class BounceTheBall {
6     public static void main(String[] args) {
7         // 모델 객체 생성
8         int box_size = 200;
9         int balls_radius = 6;
10        Box box = new Box(box_size);
11        // 공을 상자의 적절한 위치에 둠
12        MovingBall ball = new MovingBall((int)(box_size / 3.0),
13                                         (int)(box_size / 5.0),
14                                         balls_radius, box);
15        BallWriter ball_writer = new BallWriter(ball, Color.red);
16        BoxWriter box_writer = new BoxWriter(box);
17        AnimationWriter writer = new AnimationWriter(box_writer, ball_writer, box_size);
18        // 컨트롤러 객체를 생성하고 애니메이션 시작
19        new BounceController(ball, writer).runAnimation();
20    }
21 }
```

# Lab #6

## #6-1. 파란 공을 하나 추가

- 두 공은 다른 장소에서 다른 방향으로 출발한다.
- 두 공이 움직이는 속도는 같다.
- 공이 충돌해도 그대로 통과한다.

## #6-2. 충돌시 진로 수정

- 두 공이 충돌하면, 둘 다 진행 방향을 역방향으로 바꾸도록 한다.

## #6-3. 장애물 설치

- 중앙에 다음과 같은 모양의 적당한 길이의 장애물을 설치한다.



- 공이 이 장애물 위면 또는 아래 면을 만나면 y축 진행 방향을 바꾸도록 한다.