JAVA를 이용한 다양한 게임 실행 프로그램의 구현

20117894 조혜원

대구가톨릭대학교 컴퓨터공학전공

1. 서 론

JAVA 프로그래밍을 한 학기 동안 배우고 배운 내용을 바탕으로 게임 프로젝트를 구현하며 JAVA 언어의 더욱 심화적인 내용을 공부할 수 있었다. 본 보고서를 통해 게임 프로그램을 구현하기 위해 공부한 관련 연구와 해당 프로그램을 구현하기 위해 프로그램 설계한 과정과 설계한 내용을 바탕으로 프로그램을 구현한 결과를 설명하고 구현한 내용을 바탕으로 동작한 결과에 대해 설명할 계획이다.

2. 관련 연구

1. java class

클래스는 객체를 정의하는 틀 또는 설계도를 의미하며 이러한 틀을 바탕으로 여러 개의 객체를 생성하여 사용한다. 클래스에는 객체의 상태를 나타내는 필드가 존재하며, 객체의 행동을 나타내는 메소드로 구성된다. 여기서 필드는 클래스에 포함된 변수를 의미하며, 메소드는 자바에서 클래스 내에 존재하는 함수를 의미한다.

- 클래스 생성

클래스를 생성하기 위해서는 클래스 멤버인 생성자와 멤버 변수, 함수가 필요한다. 여기서 생성자는 클래스 초기화와 관련된 기능을 처리하고, 멤버 변수 중 필드라고 하는 멤버 변수는 전경변수의 역할을 한다.

2. java actionListener

actionListener 인터페이스를 구현하는 방법은 크게 “익명 클래스(anonymous Class)”와 “내부 클래스(inner Class)로 나뉜다. 여기서 익명 클래스와 내부 클래스에 대해서 자세히 살펴보고자 한다.

우선, 익명 클래스는 클래스를 생성하지 않고 하나의 클래스 내에서 인터페이스를 구현한 다음

해당 인터페이스에 맞는 메소드를 오버라이딩 해주는 방식이다. 이와 달리 내부 클래스는 클래스 내에 또 하나의 클래스를 만들고 그 내부가 되는 클래스에 인터페이스를 구현하는 방법을 의미한다.

3. 프로그램 종료

자바에서 프로그램을 종료하기 위한 방법은 여러 가지가 존재한다. 우선, 가장 간단한 방법은 System.exit()을 활용하여 현재 실행중인 프로그램을 종료한다. 이러한 명령어를 사용하지 않고 프로그램을 종료하기 위해서는 return 함수를 사용하여 프로그램 기능을 종료해야한다. 만약 exit() 함수를 사용해서 프로그램을 종료하게 되는 경우 프레임의 개수와 상관없이 모든 프로그램을 종료하게 된다.

이와 달리 프로그램 전체를 종료하는 것이 아닌 서브 프레임만을 종료하는 기능이 필요하기도 한다. 이 때는 dispose()라는 명령어를 사용하며 이에 대해 자세히 설명하고자 한다. dispose() 함수는 여러 개의 프레임이 실행되고 있어도 실행되고 있는 프로그램 전체를 종료하는 것이 아닌 현재의 프레임만 종료하는 것을 의미한다.

3. 프로그램 설계

이 프로그램은 메인 화면에서 두 개의 게임 중 하나를 선택하여 원하는 게임을 할 수 있도록 구현했다. 구현된 게임은 지뢰 찾기와 야구게임이다. 해당 게임 진행 방법에 대해 먼저 설명한다.

- 메인 화면

메인 화면임을 알리는 “GAME WORLD”라는 문구와 게임을 선택할 수 있는 버튼 두 개 구현

1. 첫 번째 버튼은 지뢰 찾기 버튼으로 해당 버튼을 눌렀을 경우 지뢰 찾기 게임이 실행된다.
2. 두 번째 버튼은 야구게임 버튼으로 해당 버튼을 눌렀을 경우 야구게임이 실행된다.

- 지뢰 찾기

지뢰 찾기는 사용자가 클릭한 블록에 나오는 숫자를 기준으로 해당 블록을 둘러싸고 있는 9개의 블록에 사용자가 클릭한 블록의 타일에 적힌 숫자만큼의 지뢰가 있는 것을 의미한다. 즉, 사용자가 여러개의 블록들을 클릭하면서 모든 지뢰를 밟지 않고 찾아내는 게임이다.

1. 지뢰 찾기 게임을 실행하면 스레드를 사용하여 계속해서 시간이 흐르도록 구현
2. 스마일 버튼을 클릭했을 때 게임 재시작
3. 지뢰를 찾는 도중 지뢰를 밟았을 때 밟았다는 메시지 출력
4. 지뢰를 밟지 않고 모든 지뢰를 다 찾았을 경우 지뢰를 모두 다 찾았다는 메시지 출력

- 야구게임

야구게임은 사용자가 입력한 숫자와 랜덤한 숫자를 비교하여 입력한 숫자와 랜덤 숫자의 위치와 숫자 모두 동일할 경우 strike를 출력하고, 숫자는 해당되지만 위치가 다를 경우에는 ball, 모두 해당 사항이 없을 겨우 out을 출력하도록 구현했다.

1. 야구게임 실행 시 랜덤한 3자리 숫자가 정해짐
2. 사용자가 랜덤한 숫자를 추측하여 입력
3. 입력한 숫자와 랜덤한 숫자를 비교한 결과를 출력
4. 랜덤한 숫자와 입력한 숫자와 위치가 모두 같을 경우 three strike로 게임이 끝남

4. 프로그램 구현

1) 메인화면 구성

```
public game() {
    setTitle("Game World");  // 프레임 타이틀 정의
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLayout(null);

    //시작 화면 game title 설정
    gameTitle = new JLabel("GAME WORLD");
    gameTitle.setBounds(43, 90, 350, 50);
    gameTitle.setFont(new Font("Times", Font.BOLD, 33));
    //gameTitle.setForeground(new Color(164, 174, 250));
    gameTitle.setForeground(new Color(31, 64, 114));
    gameTitle.setVisible(true);
    this.add(gameTitle);

    //시작 화면 지뢰 찾기 시작 버튼 설정
    startBtn = new RoundedButton("지뢰찾기");
    //startBtn.setBackground(new Color(255, 255, 255));
    startBtn.setFont(new Font("Times", Font.BOLD, 15));
    startBtn.setSize(100, 40);
    startBtn.setLocation(110, 190);
    this.add(startBtn);

    //시작 화면 단어 게임 시작 버튼 설정
    startBtn2 = new RoundedButton("야구 게임");
    //startBtn2.setBackground(new Color(255, 255, 255));
    startBtn2.setFont(new Font("Times", Font.BOLD, 15));
    startBtn2.setSize(100, 40);
    startBtn2.setLocation(110, 250);
    this.add(startBtn2);

    //시작 화면 제작자 label 설정
    JLabel producer = new JLabel("제작 : 20117894 조혜원");
    producer.setBounds(98, 345, 200, 20);
    producer.setFont(new Font("Times", Font.BOLD, 12));
    producer.setForeground(new Color(31, 64, 114));
    this.add(producer);
```

- 프레임 타이틀 정의
- label의 위치와 button의 위치 설정‘

```
    startBtn.addActionListener(new ActionListener() { // 1번문제 메뉴에 있는 실행 아이템을 클릭 시 Start클래스 윈도우 실행
        public void actionPerformed(ActionEvent e) {
            new minesweeper();
        }
    });

    startBtn2.addActionListener(new ActionListener() { // 2번문제 메뉴에 있는 메모장실행 아이템을 클릭 시 Note클래스 윈도우 실행
        public void actionPerformed(ActionEvent e) {
            new baseballGame(new randomNum());
        }
    });
```

- button 1번 (지뢰찾기)를 눌렀을 때 액션 이벤트 처리(minesweeper)
- button 2반 (야구게임)을 눌렀을 때 액션 이벤트 처리(baseballGame)

```
public class RoundedButton extends JButton {
    public RoundedButton(String text) { super(text);}

        @Override
        protected void paintComponent(Graphics g) {
            Color c=new Color(230,180,78); //배경색 결정
            Color o=new Color(0,71,113); //글자색 결정
            int width = getWidth();
            int height = getHeight();
            Graphics2D graphics = (Graphics2D) g;
            graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
            if (getModel().isArmed()) { graphics.setColor(c.darker()); }
            else if (getModel().isRollover()) { graphics.setColor(c.brighter()); }
            else { graphics.setColor(c); }
            graphics.fillRoundRect(0, 0, width, height, 10, 10);
            FontMetrics fontMetrics = graphics.getFontMetrics();
            Rectangle stringBounds = fontMetrics.getStringBounds(this.getText(), graphics).getBounds();
            int textX = (width - stringBounds.width) / 2;
            int textY = (height - stringBounds.height) / 2 + fontMetrics.getAscent();
            graphics.setColor(o);
            graphics.setFont(getFont());
            graphics.drawString(getText(), textX, textY);
            graphics.dispose();
            super.paintComponent(g);
        }
}
```

- 메인 화면의 버튼 모양을 둥근 사각형으로 바꾸기 위한 코드로 해당 코드는 인터넷을 참고함

2) minesweeper

```
class Timer implements Runnable{ //쓰레드를 구현하기 위한 Timer
    JLabel timerLabel;
    private int n=0;
    /*
     *  @param  timerLabel 숫자를 보여주기위한 레이블
     *  @param n 시간을 체크하기 위한 변수
     */
    public Timer(JLabel timerLabel) { //생성자
        this.timerLabel=timerLabel;
    }
    @Override
    public void run() {
        int i=1; //플레이한 시간을 체크하기 위한 변수
        while(true) {
            timerLabel.setText(Integer.toString(n)); //레이블의 텍스트를 n으로 변경
            n++;//n값 증가
            try {
                Thread.sleep(1000);//1000만큼 대기
            }
            catch(InterruptedException e) {//예외발생시 종료
                return;
            }
```

- thread를 활용하여 게임 실행 시 타이머가 동작하도록 구현

```
    class resetAction implements ActionListener{ //스마일마크 클릭시 게임 재시작
        public void actionPerformed(ActionEvent e) {
            getContentPane().removeAll(); //컨탠트팬 삭제
            c.removeAll(); //c에 할당된 모든 것을 삭제
            c=getContentPane(); //c에 컨탠트팬 할당
            resetPanel(); //이미지버튼을 기본이지미로
            c.setLayout(new BorderLayout()); //레이아웃은BorderLayout
            c.add(p,BorderLayout.NORTH); //북쪽에 패널 추가
            Image img=iconArr[3].getImage();
            Image newImg;

            c.add(new Stage(9,9,10),BorderLayout.CENTER);////행, 열, 지뢰수, 초급단계

            c.setVisible(true);//프레임 출력
            iButton.setIcon(iconArr[0]); //스마일마크를 기본이미지로

        }
    }
```

- 스마일 버튼을 클릭했을 시 게임이 다시 재시작됨
- 지뢰의 사이즈는 9X9로 지정하고 10개의 지뢰가 존재하도록 구현

```
int count=0; //지뢰의 수 체크  변수
if( ((i-1)>=min) && ((i-1)<iRow)){ //a 위치의 지뢰체크
    int x=i-1; //행
    if( ( (s-1) >=min ) && ( (s-1)<iCol ) )//s-1이 0이상이면서 열보다 작다면
        if(checkArr[x][s-1]==true) //실제 그곳에 지뢰가 있다면 count증가
            count++;
    if( ( s >=min ) && ( s<iCol ) ) //s가 0이상이면서 열보다 작다면
        if(checkArr[x][s]==true)//실제 그곳에 지뢰가 있다면 count증가
            count++;
    if( ( (s+1) >=min ) && ( (s+1)<iCol ) )//s+1이 0이상이면서 열보다 작다면
        if(checkArr[x][s+1]==true)//실제 그곳에 지뢰가 있다면 count증가
            count++;
}
if( (i>=min) && (i<iRow)) { //b 위치의 지뢰체크
    int x=i;
    if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //s-1이 0이상이면서 열보다 작다면
        if(checkArr[x][s-1]==true)//실제 그곳에 지뢰가 있다면 count증가
            count++;
    if( ( (s+1) >=min ) && ( (s+1)<iCol ) ) //s+1이 0이상이면서 열보다 작다면
        if(checkArr[x][s+1]==true)//실제 그곳에 지뢰가 있다면 count증가
            count++;
}
if( ((i+1)>=min) && ((i+1)<iRow)) { //c 위치의 지뢰체크
    int x=i+1;
    if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //s-1이 0이상이면서 열보다 작다면
        if(checkArr[x][s-1]==true)//실제 그곳에 지뢰가 있다면 count증가
            count++;
    if( ( s >=min ) && ( s<iCol ) ) //s가 0이상이면서 열보다 작다면
        if(checkArr[x][s]==true) //실제 그곳에 지뢰가 있다면 count증가
            count++;
    if( ( (s+1) >=min ) && ( (s+1)<iCol ) ) //s+1이 0이상이면서 열보다 작다면
        if(checkArr[x][s+1]==true)//실제 그곳에 지뢰가 있다면 count증가
            count++;
}
this.numberArr[i][s]=count; //근처 지뢰의 수를 체크하는 배열에 count값 할당
```

- 사용자가 클릭하여 나온 숫자를 바탕으로 지뢰의 존재 여부를 파악

3) baseballGame

```
//배열에 3자리 랜덤 숫자 선언
class randomNum {
    int randomArr[];
    public randomNum() {
        randomArr = new int[3];

        randomArr[0] = (int)(Math.random()*9+1); //백의 자리의 숫자(랜덤값 1~9)
        randomArr[1] = (int)(Math.random()*9+1); //십의 자리의 숫자(랜덤값 1~9)
        randomArr[2] = (int)(Math.random()*9+1); //일의 자리의 숫자(랜덤값 1~9)

        //3자리의 숫자에 동일한 값이 들어가지 않도록 설정
        while(randomArr[0]==randomArr[1]) {
            randomArr[1] = (int)(Math.random()*9+1);
        }

        while(randomArr[0]==randomArr[2] || randomArr[1]==randomArr[2]) {
            randomArr[2] = (int)(Math.random()*9+1);
        }
    }

    public int[] getRandomArr() {
        return randomArr;
    }
}
```

- 게임이 시작되었을 때 게임을 진행하기 위한 랜덤 값을 지정
- 각 배열에 한 자리씩 중복되는 숫자가 없도록 설정

```
//사용자가 입력한 숫자와 랜덤 숫자 비교 결과를 나타내는 클래스
class resultCheck {
    int strike, ball;
    boolean out;
    public resultCheck(int[] randomArr, int[] inputNum) {
        for(int i=0; i<3; i++) {
            for(int j=0; j<3; j++) {
                if(randomArr[i] == inputNum[j]) { //랜덤 숫자와 사용자의 입력 숫자가 같을 경우 아래의 조건를 실행
                    if(i == j) strike++; //랜덤 숫자와 사용자가 입력한 숫자가 같고 위치도 같을 경우 스트라이크 값 증가
                    else ball++; //랜덤 숫자와 사용자가 입력한 숫자는 같지만 위치가 다를 경우 볼 값 증가
                }
            }
        }
        //사용자가 입력한 숫자가 랜덤 숫자에 모두 포함되어 있지 않을 경우 아웃값을 true, 그렇지 않을 경우 false 반환
        if(strike == 0 && ball == 0) out = true;
        else out = false;
    }

    public int resultStrike() {
        return strike;
    }

    public int resultBall() {
        return ball;
    }

    public boolean resultOut() {
        return out;
    }
}
```

- 사용자 입력한 값과 게임 시작 시 지정되었던 랜덤 숫자를 비교하여 사용자가 입력한 값이 strike인지 ball인지 out인지 판단
- if문으로 조건을 걸어 판단

```
public void paintComponent (Graphics g) {
    super.paintComponent(g);

    g.setColor(Color.WHITE);
    g.setFont(new Font("", Font.BOLD, 15));
    g.drawString("빈칸에 3자리 숫자를 입력하세요", 80, 90);

    if(strike == 3) {
        g.setFont(new Font("", Font.BOLD, 40));
        g.drawString("!THREE STRIKE!", 30, 180); //랜덤 숫자와 입력한 숫자가 모두 같을 경우 출력되는 문구
        g.setFont(new Font("", Font.BOLD, 20));
        g.drawString("숫자를 모두 맞췄습니다!", 70, 220);
        g.drawString("게임을 재시작해주세요", 75, 260);
    }
    else {
        g.setFont(new Font("", Font.BOLD, 20));

        //입력한 숫자와 랜덤 숫자의 위치와 숫자가 같을 경우 출력되는 문구
        if(strike == 0) g.drawString("-", 30, 160);
        else if(strike == 1) g.drawString("ONE STRIKE", 30, 160);
        else if(strike == 2) g.drawString("TWO STRIKE", 30, 160);

        //입력한 숫자와 랜덤 숫자의 숫자는 같지만 위치가 다를 경우 출력되는 문구
        if(ball == 0) g.drawString("-", 30, 230);
        else if(ball == 1) g.drawString("ONE BALL", 30, 230);
        else if(ball == 2) g.drawString("TWO BALL", 30, 230);

        //입력한 숫자와 랜덤 숫자가 모두 일치하는게 없는 경우 출력되는 문구
        if(out) g.drawString("OUT", 30, 300);
        else g.drawString("-", 30, 300);
    }
```

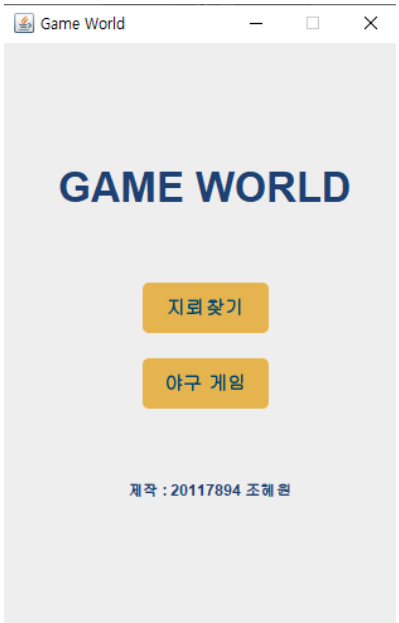
- 위의 조건에 따라 게임 플레이 화면에 랜덤 숫자와 사용자가 입력한 숫자를 비교하여 결과를 출력

```java
@Override
public void actionPerformed(ActionEvent e) {
    int[] userNum = new int[3];
    String[] num = text.getText().split("");

    for(int i=0; i<num.length; i++)
        userNum[i] = Integer.parseInt(num[i]);

    resultCheck rc = new resultCheck(rn.getRandomArr(), userNum);
    System.out.println(rc.resultBall() + "" + rc.resultStrike());
    rp.setResult(rc.resultStrike(), rc.resultBall(), rc.resultOut());
    rp.repaint();
}
```

- 사용자가 숫자를 입력하고 “결과 보기” 버튼을 클릭했을 시 발생하는 이벤트

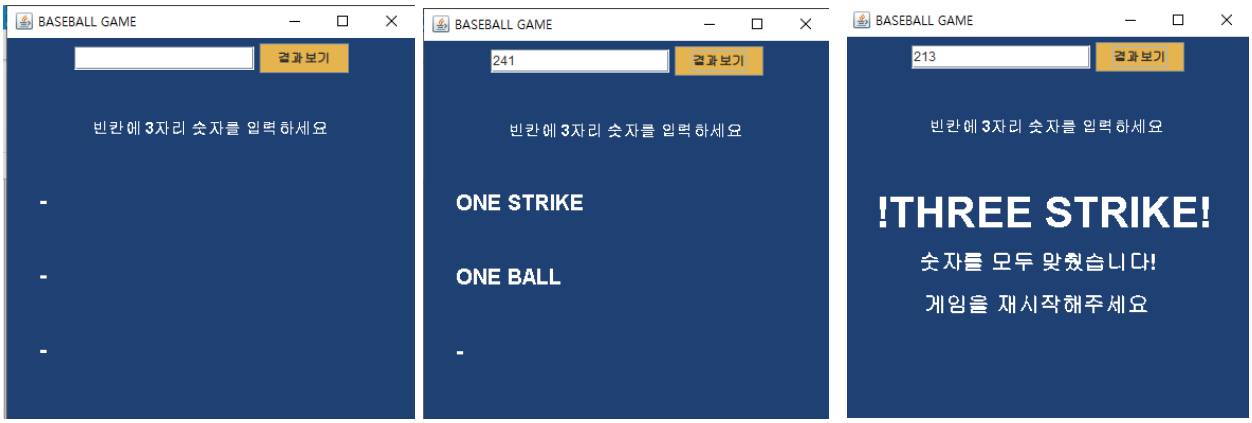
5. 프로그램의 동작과 결과

1) 메인 화면



2) 지뢰찾기 화면 -> 지뢰찾기 버튼을 클릭했을 때 나타나는 창

3) 야구게임 화면 -> 야구게임 버튼을 클릭했을 때 나타나는 창



해당 메인 화면에서 각각의 게임 실행 버튼을 눌렀을 경우 모두 제대로 동작하며 해당 게임을 진행할 때 오류없이 제대로 진행이 되는 것을 확인할 수 있었다.

6. 결과 분석 및 토의

1) 프로그램 메인 화면

- 프로그램이 시작되면 프로그램의 메인 화면에는 메인 화면임을 나타내는 label과 게임을 실행시킬 수 있는 버튼 두 개가 나타난다.
- 지뢰 찾기 버튼을 누르면 지뢰 찾기 게임을 할 수 있는 새로운 화면이 나타난다.
- 야구게임 버튼을 누르면 야구게임을 할 수 있는 새로운 화면이 나타난다.
- 프로그램 메인 화면이나 각 프로그램의 실행화면을 닫기 버튼을 눌러 종료하면 강제로 게임이 종료된다.

2) 지뢰 찾기 화면

- thread를 통해 게임이 실행되었을 때 계속해서 숫자가 올라가도록 설정
- 스마일 버튼을 눌렀을 경우 지뢰 찾기 게임을 리셋하고 다시 게임을 재시작
- 지뢰찾기를 진행하는 과정에서 지뢰를 밟았을 경우 지뢰를 밟았다는 문구 출력
- 지뢰찾기를 진행하는 과정에서 지뢰를 밟지 않고 모든 지뢰를 찾았을 경우에는 모두 다 찾았다는 문구를 출력한다.

3) 야구 게임 화면

- 정의한 class에서 게임을 실행했을 때 바로 랜덤 숫자를 할당하도록 함
- 사용자가 랜덤으로 할당받은 값을 맞추기 위해 계속해서 숫자 비교
- 숫자 비교를 하며 사용자가 입력한 해당 숫자와 랜덤 숫자의 맞춘 빈도수 출력
- 해당 결과가 three strike가 출력될 시 모든 숫자를 맞춘 것으로 간주

7. 결론

이번 JAVA 텀 프로젝트를 진행하며 이론으로 공부한 것을 바탕으로 제대로 된 실습을 진행할 수 있었다. 비록 처음 JAVA 프로그래밍 언어로 구현할 때 처음 해보는 프로젝트였기 때문에 유익한 시간이었던 것 같다. 이론으로만 공부를 진행했을 때는 정확히 어떻게 사용해야 되는지 정확히 알 수가 없었는데 이를 실제로 프로그램을 구현할 때 사용해보니까 JAVA에서 매우 중요하다는 것을 알게 되었다. 해당 프로젝트를 작성하면서 class뿐만 아니라 thread에 대해서도 어려움을 겪었는데 해당 프로그램을 구현하기 위해 다양한 관련 연구를 찾아보며 thread에 대해 공부하고 프로그램을 구현하니까 조금 더 수월하게 코드를 작성할 수 있었던 것 같다.

8. 참고문헌

[1] 황기태, 김효수, “명품 JAVA 프로그래밍”, 생능출판사, 2018

[2] “java 버튼 둥근 사각형”, https://the-illusionist.me/42

[3] “숫자 야구 게임”, https://it-dolphin.tistory.com/entry/JavaGUI%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-%EC%88%AB%EC%9E%90%EC%95%BC%EA%B5%AC%EA%B2%8C%EC%9E%84

[4] “java class”, https://wikidocs.net/214

[5] “java class“, http://tcpschool.com/java/java_class_intro

[6] ”method“, https://wikidocs.net/225

[7] ”클래스 생성“, https://nowonbun.tistory.com/298

[8] ”actionListener“, https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=hotkimchi13&logNo=221280610116

[9] ”java 프로그램 종료“, https://www.delftstack.com/ko/howto/java/java-end-program/

[10] ”java 여러 프레임 종료“, https://modesty101.tistory.com/173

9. 별첨 (소스코드와 주석)

<main 화면 코드>

```
import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import java.awt.event.*;
import java.io.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class game extends JFrame { // 초기화면 클래스 정의
	private JButton startBtn;
	private JButton startBtn2;
	private JLabel gameTitle;

	public game() {
		setTitle("Game World");  // 프레임 타이틀 정의
		setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		setLayout(null);

		//시작 화면 game title 설정
		gameTitle = new JLabel("GAME WORLD");
		gameTitle.setBounds(43, 90, 350, 50);
		gameTitle.setFont(new Font("Times", Font.BOLD, 33));
		//gameTitle.setForeground(new Color(164, 174, 250));
		gameTitle.setForeground(new Color(31, 64, 114));
		gameTitle.setVisible(true);
		this.add(gameTitle);

		//시작 화면 지뢰 찾기 시작 버튼 설정
		startBtn = new RoundedButton("지뢰찾기");
		//startBtn.setBackground(new Color(255, 255, 255));
		startBtn.setFont(new Font("Times", Font.BOLD, 15));
		startBtn.setSize(100, 40);
		startBtn.setLocation(110, 190);
		this.add(startBtn);

		//시작 화면 단어 게임 시작 버튼 설정
		startBtn2 = new RoundedButton("야구 게임");
		//startBtn2.setBackground(new Color(255, 255, 255));
		startBtn2.setFont(new Font("Times", Font.BOLD, 15));
		startBtn2.setSize(100, 40);
		startBtn2.setLocation(110, 250);
		this.add(startBtn2);

		//시작 화면 제작자 label 설정
		JLabel producer = new JLabel("제작 : 20117894 조혜원");
		producer.setBounds(98, 345, 200, 20);
		producer.setFont(new Font("Times", Font.BOLD, 12));
		producer.setForeground(new Color(31, 64, 114));
		this.add(producer);

		startBtn.addActionListener(new ActionListener() { // 1번문제 메뉴에 있는 실행 아이템을 클릭 시 Start클래스 윈도우 실행
			public void actionPerformed(ActionEvent e) {
				new minesweeper();
			}
		});

		startBtn2.addActionListener(new ActionListener() { // 2번문제 메뉴에 있는 메모장실행 아이템을 클릭 시 Note클래스 윈도우 실행
			public void actionPerformed(ActionEvent e) {
				new baseballGame(new randomNum());
			}
		});

		setSize(330, 500);
		setVisible(true);
		setResizable(false); // 창의 크기를 고정
		setLocationRelativeTo(null);
	}

	//버튼 둥근 사각형 디자인 - 출처 : https://the-illusionist.me/42
```

```
	public class RoundedButton extends JButton {
		public RoundedButton(String text) { super(text);}

			@Override
			protected void paintComponent(Graphics g) {
				Color c=new Color(230,180,78); //배경색 결정
				Color o=new Color(0,71,113); //글자색 결정
				int width = getWidth();
				int height = getHeight();
				Graphics2D graphics = (Graphics2D) g;
				graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
				if (getModel().isArmed()) { graphics.setColor(c.darker()); }
				else if (getModel().isRollover()) { graphics.setColor(c.brighter()); }
				else { graphics.setColor(c); }
				graphics.fillRoundRect(0, 0, width, height, 10, 10);
				FontMetrics fontMetrics = graphics.getFontMetrics();
				Rectangle stringBounds = fontMetrics.getStringBounds(this.getText(),
graphics).getBounds();
				int textX = (width - stringBounds.width) / 2;
				int textY = (height - stringBounds.height) / 2 + fontMetrics.getAscent();
				graphics.setColor(o);
				graphics.setFont(getFont());
				graphics.drawString(getText(), textX, textY);
				graphics.dispose();
				super.paintComponent(g);
			}
	}

	public static void main(String [] args) {
		new game(); // No1 객체 실행
	}
}
```

<지뢰 찾기 화면>

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.Random;

public class minesweeper extends JFrame {
	public Container c;
	myPanel p; //패널
	private JButton iButton=new JButton(); //이미지버튼
	private JLabel stageShow=new JLabel("<html>초급<br/>지뢰수 : 10"); //패널에 부착할 라벨
	ImageIcon iconArr[]= {new ImageIcon("images/스마일마크.jpg"),new ImageIcon("images/승리마크.png"),new
ImageIcon("images/사망마크.jpg"),new ImageIcon("images/타일.png")};
	private JLabel timerLabel;
	private Thread th;
	private Timer runnable;

	public minesweeper() { //생성자
		c=getContentPane();//mine 클래스의 컨탠트팬 저장
		p=new myPanel(); //p에 새로운 패널 할당
		c.setLayout(new BorderLayout()); //레이아웃을은borderLayout
		c.add(p,BorderLayout.NORTH); //레이아웃의 북쪽에 패널할당
		c.add(new Stage(9,9,10),BorderLayout.CENTER);

		setSize(300,300);//크기는 300,300
		setVisible(true); //프레임 출력
		setTitle("MINESWEEPER");
	}

	private void resetPanel() {
		iButton.setIcon(iconArr[0]); //화면 상단의 이미지를 기본 스마일마크로 설정
	}
	class myPanel extends JPanel{//화면 상단에 부착될 패널
		public myPanel() {
			setLayout(new FlowLayout(FlowLayout.CENTER,30,0)); //레이아웃은 FlowLayout 중심정렬
및 수평간격 30 수직간격 0으로 설정
			//타이머설정
			timerLabel=new JLabel(); //Label 새로 동적할당
			timerLabel.setSize(200, 200); //레이블 크기는 200,200
			runnable=new Timer(timerLabel); //쓰레드로 동작시킬 메소드 할당
			th=new Thread(runnable); //쓰레드 할당
			th.start(); //쓰레드 시작
			//스마일마크 설정
```

```
                iButton.setBackground(Color.white); //이미지버튼 배경색은 흰색
                iButton.setIcon(iconArr[0]);//기본 이미지
                iButton.setBorder(null); //이미지의 여백 삭제
                iButton.setBorderPainted(false); //
                iButton.addActionListener(new resetAction()); //이미지버튼에 리스너 부착

                add(timerLabel); //쓰레드로 돌아가는 레이블 부착
                add(iButton); //이미지버튼
                add(stageShow); //스테이지
        }
        class resetAction implements ActionListener{ //스마일마크 클릭시 게임 재시작
                public void actionPerformed(ActionEvent e) {
                        getContentPane().removeAll(); //컨탠트팬 삭제
                        c.removeAll(); //c에 할당된 모든 것을 삭제
                        c=getContentPane(); //c에 컨탠트팬 할당
                        resetPanel(); //이미지버튼을 기본이지미로
                        c.setLayout(new BorderLayout()); //레이아웃은BorderLayout
                        c.add(p,BorderLayout.NORTH); //북쪽에 패널 추가
                        Image img=iconArr[3].getImage();
                        Image newImg;

                        c.add(new Stage(9,9,10),BorderLayout.CENTER);////행, 열, 지뢰수, 초급단계


                        c.setVisible(true);//프레임 출력
                        iButton.setIcon(iconArr[0]); //스마일마크를 기본이미지로

                }
        }
        private void resetPanel() {
                iButton.setIcon(iconArr[0]); //화면 상단의 이미지를 기본 스마일마크로 설정
        }
}


class Stage extends JPanel {
        private boolean left=false,right=false,same=false;
        final int min=0;
        int iRow,iCol;
        JButton button[][];
        int numberArr[][];
        int mine;
        boolean checkArr[][];

        private void numberInit(int i,int s) { //근처의 지뢰의 수를 파악해 버튼이 클릭 되었을때 삽입
                int count=0; //지뢰의 수 체크  변수
                if( ((i-1)>=min) && ((i-1)<iRow)){ //a 위치의 지뢰체크
                        int x=i-1; //행
                        if( ( (s-1) >=min ) && ( (s-1)<iCol ) )//s-1이 0이상이면서 열보다 작다면
                                if(checkArr[x][s-1]==true) //실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                        if( ( s >=min ) && ( s<iCol ) ) //s가 0이상이면서 열보다 작다면
                                if(checkArr[x][s]==true)//실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                        if( ( (s+1) >=min ) && ( (s+1)<iCol ) )//s+1이 0이상이면서 열보다 작다면
                                if(checkArr[x][s+1]==true)//실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                }
                if( (i>=min) && (i<iRow)) { //b 위치의 지뢰체크
                        int x=i;
                        if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //s-1이 0이상이면서 열보다 작다면
                                if(checkArr[x][s-1]==true)//실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                        if( ( (s+1) >=min ) && ( (s+1)<iCol ) ) //s+1이 0이상이면서 열보다 작다면
                                if(checkArr[x][s+1]==true)//실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                }
                if( ((i+1)>=min) && ((i+1)<iRow)) { //c 위치의 지뢰체크
                        int x=i+1;
                        if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //s-1이 0이상이면서 열보다 작다면
                                if(checkArr[x][s-1]==true)//실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                        if( ( s >=min ) && ( s<iCol ) ) //s가 0이상이면서 열보다 작다면
                                if(checkArr[x][s]==true) //실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                        if( ( (s+1) >=min ) && ( (s+1)<iCol ) ) //s+1이 0이상이면서 열보다 작다면
                                if(checkArr[x][s+1]==true)//실제 그곳에 지뢰가 있다면 count증가
                                        count++;
                }
                this.numberArr[i][s]=count; //근처 지뢰의 수를 체크하는 배열에 count값 할당

        }
```

```
		public Stage(int iRow,int iCol,int imine) { //Stage클래스의 생성자
			//생성자로 부터 전달받은 값을 할당
			this.iRow=iRow;this.iCol=iCol; //전달 받은 행과 열 할당
			this.button=new JButton[this.iRow][this.iCol]; //button배열에 행과 열에 맞게 할당
			this.numberArr=new int[this.iRow][this.iCol]; //numberArr배열에 행과 열에 맞게 할당
			this.mine=imine; //전달받은 지뢰의 수 할당
			this.checkArr=new boolean[this.iRow][this.iCol]; //지뢰를 저장하는 배열
			setLayout(new GridLayout(this.iRow,this.iCol)); //레이아웃은 grid 행과 열에 맞게 생성
			Random rand=new Random(System.currentTimeMillis()); //랜덤을 쓰기위해
			for(int i=0;i<this.iRow;i++) //지뢰 배열초기화
				for(int s=0;s<this.iCol;s++)
					checkArr[i][s]=false; //기본 false로 초기화

			for(int i=0;i<mine;i++) { //지뢰배치
				int x=rand.nextInt(iRow); //행에 맞게 난수 할당
				int y=rand.nextInt(iCol); //열에 맞게 난수 할당
				if(checkArr[x][y]==false) { //난수에 해당하는 인덱스가 가리키는 값이 false라면
true로 변경(지뢰 배치)
					checkArr[x][y]=true;
				}
				else
					i--; //이미 해당하는 위치에 지뢰가 배치되었기에 다시 난수 할당
			}
//			JOptionPane.showMessageDialog(null, "게임을 시작합니다.\n지뢰의 갯수는 "+ mine + "개
입니다.", "지뢰의 수 ", JOptionPane.INFORMATION_MESSAGE); //지뢰 갯수 출력 메시지
			Image img=iconArr[3].getImage();
			Image newImg;
			newImg=img.getScaledInstance(32, 27, java.awt.Image.SCALE_SMOOTH);

			iconArr[3]=new ImageIcon(newImg);
			//버튼생성
			for(int i=0;i<this.iRow;i++) { //행의 수만큼 반복
				for(int s=0;s<this.iCol;s++) { //열의 수만큼 반복
					//button[i][s]=new JButton(); //버튼할당

					button[i][s]=new JButton(iconArr[3]);
					//button[i][s].setBorder(null);
					//button[i][s].setBorderPainted(false);
					button[i][s].setMargin(new Insets(0,0,0,0)); //버튼의 여백을 없게
					button[i][s].setFont(new Font("함초롱바탕",Font.BOLD,20)); //버튼의
폰트설정

					button[i][s].setForeground(Color.red); //지뢰의 색설정
					if(checkArr[i][s]!=true) //해당버튼이 지뢰가 아니라면
						this.numberInit(i, s); //주위의 지뢰의 수 체크 후 저장
					else
						this.numberArr[i][s]=-1; //지뢰이기에 numberArr에 -1저장

					button[i][s].addMouseListener(new Mouse()); //버튼에 리스너 부착
					button[i][s].setBackground(Color.blue); //버튼의 배경색은 회색
					add(button[i][s]); //패널에 버튼 부착
				}
			}
		}
		class Mouse extends MouseAdapter{ //버튼을 눌렀을 시에 동작하는 마우스리스너

			private int blankCheck(int row,int col) {//해당하는 버튼이 공백이라면 주위의 버튼들도
체크하는 메소드
				if( ( (row>=min) && (row<iRow) ) && ( (col>=min) && (col<iCol) ) ){ //전달
받은 행과 열이 범위를 초과하지 않는지
					if(numberArr[row][col]==0 && (button[row][col].isEnabled()==true)
&& !button[row][col].getText().equals("?")) { //행과 열에 해당하는 곳이 빈칸이면서 사용가능하다면
						button[row][col].setIcon(null); //변경점
						button[row][col].setText(""); //해당하는 곳의 텍스트를 ""로
						button[row][col].setBackground(Color.white);//배경색을 흰색
						button[row][col].setEnabled(false);//사용가능하지 않게(빈칸은
클릭되지 않게)

						blankCheck(row-1,col-1); //a위치의 맨 좌측 체크
						blankCheck(row-1,col); //위치의 중간체크
						blankCheck(row-1,col+1); //a위치의 맨 우측 체크
						blankCheck(row,col-1); //b위치의 좌측 체크
						blankCheck(row,col+1); //b위치의 우측 체크
						blankCheck(row+1,col-1); //c위치의 맨 좌측 체크
						blankCheck(row+1,col); //c위치의 중간 체크
						blankCheck(row+1,col+1); //c위치의 맨 우측 체크
						return 1; //재귀를 사용과 동시에 종료하기위한 의미없는 리턴값
					}
					else if(numberArr[row][col]>0) { //가라키는 곳에 지뢰가 없고 빈칸이
아니라면
						int x=numberArr[row][col]; //가리키는 곳의 값을 저장

button[row][col].setText(Integer.toString(numberArr[row][col])); //버튼에 해당하는 숫자를 저장
```

```
                                        button[row][col].setIcon(null);
                                        //숫자에 맞게 버튼의 글자색을 변경하기 위해
                                        if(x==1)
                                                button[row][col].setForeground(Color.blue);
                                        else if(x==2)
                                                button[row][col].setForeground(Color.green);
                                        else if(x==3)
                                                button[row][col].setForeground(Color.red);
                                        else if(x==4)
                                                button[row][col].setForeground(Color.magenta);
                                        else if(x==5)
                                                button[row][col].setForeground(Color.darkGray);
                                        else
                                                button[row][col].setForeground(Color.red);

                                        button[row][col].setBackground(Color.white); //배경색을
흰색으로

                                        return 1; //재귀를 종료하기 위한 의미없는 리턴 값
                                }
                                else
                                        return 1; //재귀를 종료하기 위한 의미없는 리턴 값
                        }
                        else
                        return 1; //재귀를 종료하기 위한 의미없는 리턴 값
                }
                public void mousePressed(MouseEvent e) { //마우스를 눌렀을때 동작하는 마우스 리스너
                        if(e.getButton()==MouseEvent.BUTTON1)//왼쪽 버튼이 눌려졌다면 left를 true
                                left=true;
                        if(e.getButton()==MouseEvent.BUTTON3)//오른쪽 버튼이 눌려졌다면 right를 true
                                right=true;

                        if(left==true && right==true) //마우스 왼쪽 오른쪽 둘다 눌려졌다면 same을 true로
                                same=true;

                }
                public void mouseReleased(MouseEvent e) { //마우스를 뗐을 때 동작하는 리스너
                        JButton bu=(JButton)e.getSource();//bu에 해당하는 버튼을 캐스팅해 저장
                        int row=0,col=0;
                        for(int i=0;i<iRow;i++) { //마우스가 떼진 버튼의 인덱스를 찾기위한 반복문
                                for(int s=0;s<iCol;s++) {
                                        if(button[i][s]==bu) {
                                                row=i;
                                                col=s;
                                                break;
                                        }
                                }
                        }

                        if(same) { //마우스 양쪽을 같이 뗐을 경우
                                int num=numberArr[row][col]; //해당하는 버튼의 텍스트를 저장
                                int count=0; //근처 지뢰의 수를 체크하기위한 변수
                                if(row>=min&&row<iCol &&col>=min&&col<iCol) { //해당하는 행과
열이 범위를 벗어나지는 않았는지

                                        int i=row,s=col; //i는 행 s는 열을 가리킴
                                        boolean bFlag=true; //패배를 체크하기위한 bool형 변수 bflag

                                        if( ((i-1)>=min) && ((i-1)<iRow) && bFlag){ //a 위치의 깃발
체크
                                                int x=i-1; //x에 i-1저장, a위치의 행을 가리키기
위함

                                                if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //a위치의
맨좌측 깃발체크

if(button[x][s-1].getText().equals("★")==true) //깃발이 꼽혀있니?

if(numberArr[x][s-1]==-1)//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                                        count++;
                                                                else //깃발이 있는 곳에 지뢰가
없으면 bflag를 false로
                                                                        bFlag=false;

                                                if( ( s >=min ) && ( s<iCol ) ) //a위치의  중간
깃발 체크

if(button[x][s].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                                if(numberArr[x][s]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                                        count++;
                                                                else //깃발이 있는
곳에 지뢰가 없으면 bflag를 false로
```

```
bFlag=false;
                                                    if( ( (s+1) >=min ) && ( (s+1)<iCol ) ) //a위치의
중간 깃발 체크
if(button[x][s+1].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                        if(numberArr[x][s+1]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                            count++;
                                                            else //깃발이 있는
곳에 지뢰가 없으면 bflag를 false로
bFlag=false;
                                                }
                                                if( (i>=min) && (i<iRow) && bFlag) { //b 위치의 깃발 체크
                                                        int x=i;
                                                        if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //b위치의
맨좌측 깃발체크
if(button[x][s-1].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                        if(numberArr[x][s-1]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                            count++;
                                                            else //깃발이 있는
곳에 지뢰가 없으면 bflag를 false로
bFlag=false;
                                                    if( ( (s+1) >=min ) && ( (s+1)<iCol ) )
//b위치의 맨우측 깃발체크
if(button[x][s+1].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                        if(numberArr[x][s+1]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                            count++;
                                                            else //깃발이 있는
곳에 지뢰가 없으면 bflag를 false로
bFlag=false;
                                                }
                                                if( ((i+1)>=min) && ((i+1)<iRow) && bFlag) { //c 위치의
깃발 체크
                                                        int x=i+1;
                                                        if( ( (s-1) >=min ) && ( (s-1)<iCol ) ) //c위치의
맨좌측 깃발체크
if(button[x][s-1].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                        if(numberArr[x][s-1]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                            count++;
                                                            else //깃발이 있는
곳에 지뢰가 없으면 bflag를 false로
bFlag=false;
                                                    if( ( s >=min ) && ( s<iCol ) ) //c위치의 중간
깃발체크
if(button[x][s].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                        if(numberArr[x][s]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                            count++;
                                                            else //깃발이 있는
곳에 지뢰가 없으면 bflag를 false로
bFlag=false;
                                                    if( ( (s+1) >=min ) && ( (s+1)<iCol ) ) //c위치의
맨우측 깃발체크
if(button[x][s+1].getText().equals("★")==true) //깃발이 꼽혀있니?
                                                        if(numberArr[x][s+1]==-1)
//깃발이 있는곳에 지뢰가 있다면 count를 증가
                                                        count++;
                                                        else //깃발이 있는 곳에 지뢰가
없으면 bflag를 false로
                                                            bFlag=false;
                                                }
```

```
                                        if(bFlag==false) { //가리킨 버튼의 숫자를 초과하는 수의
깃발을 꼽았거나 깃발을 지뢰가 없는 곳에 자신의 수이상 꼽은 경우
                                            JOptionPane.showMessageDialog(null, "지뢰를
밟았습니다", "패배", JOptionPane.ERROR_MESSAGE); //패배 메시지 출력
                                            iButton.setIcon(iconArr[2]); //화면 상단의 이미지를
실패 이미지로

                                            for(i=0;i<iRow;i++)  //해당하는 행만큼
                                                for(s=0;s<iCol;s++)  //해당하는 열만큼
                                                    button[i][s].setEnabled(false);
//모든 버튼들을 사용 불가로
                                        }
                                    }
                                    if(num==count) { //깃발을 제대로 꼽고 마우스 양쪽을 클릭한 경우
                                        //공백을 체크하는 함수 실행
                                        blankCheck(row-1,col-1); //a위치의 맨 우측
                                        blankCheck(row-1,col); //a위치의 중앙
                                        blankCheck(row-1,col+1); //a위치의 맨 좌측
                                        blankCheck(row,col-1); //b위치의 좌측
                                        blankCheck(row,col+1); //b위치의 우측
                                        blankCheck(row+1,col-1); //c위치의 맨 좌측
                                        blankCheck(row+1,col); //c위치의 중앙
                                        blankCheck(row+1,col+1); //c위치의 맨 우측
                                    }
                                }
                                else if(left) { //마우스 왼쪽만 클릭한 경우

                                    if(button[row][col].getText().equals("★")!=true
&&button[row][col].isEnabled()==true &&button[row][col].getText().equals("")) { //깃발이 꼽혀있는 곳은 클릭되지 않게

                                        if(numberArr[row][col]==0) { //비어있는 곳을 클릭한 경우
                                            this.blankCheck(row, col);//공백체크 메소드
                                        }
                                        else if(numberArr[row][col]==-1) { //지뢰를 밟은 경우
                                            JOptionPane.showMessageDialog(null, "지뢰를
밟았습니다", "패배", JOptionPane.ERROR_MESSAGE); //실패 메시지

                                            for(int i=0;i<iRow;i++)  //행의 수만큼 반복
                                                for(int s=0;s<iCol;s++)  //열의 수만큼
반복
                                                    button[i][s].setEnabled(false);
// 버튼들을 사용 불가로

                                            iButton.setIcon(iconArr[2]); //스마일마크를 실패
이미지로
                                        }
                                        else { //지뢰 근처의 버튼을 클릭한 경우(숫자출력)
                                            int x=numberArr[row][col]; //해당하는 곳의 숫자를
저장
                                            button[row][col].setIcon(null);

button[row][col].setText(Integer.toString(numberArr[row][col])); //해당하는버튼의 텍스트를 x로 변경

                                            //해당하는 곳의 숫자에 따라서 글자 색을 변경
                                            if(x==1)

button[row][col].setForeground(Color.blue);
                                            else if(x==2)

button[row][col].setForeground(Color.green);
                                            else if(x==3)

button[row][col].setForeground(Color.red);
                                            else if(x==4)

button[row][col].setForeground(Color.magenta);
                                            else if(x==5)

button[row][col].setForeground(Color.darkGray);
                                            else

button[row][col].setForeground(Color.red);
                                            button[row][col].setBackground(Color.white);
//배경색은 흰색
                                        }
                                    }
                                }
                                else if(right) { //마우스 오른쪽을 클릭한 경우
                                    if(button[row][col].getText().equals("★")==true) {//해당하는 곳에
깃발이 꼽혀 있다면
                                        button[row][col].setText("?");
                                    }
```

```
                                        else if(button[row][col].getText().equals("?")==true) {//해당하는 곳에
깃발이 꽂혀 있다면
                                                button[row][col].setText(""); //깃발 제거
                                                button[row][col].setIcon(iconArr[3]);
                                        }
                                        else if(button[row][col].getText().equals("")) { //빈칸이라면
                                                button[row][col].setIcon(null);
                                                button[row][col].setText("★"); //깃발 생성
                                        }

                                }
                                int counter=0; //승리조건을 카운트하는 변수
                                //해당 마우스 리스너가 실행될 때 마다 승리조건을 체크
                                for(int i=min;i<iRow;i++) { //행의 수만큼 반복
                                        for(int s=min;s<iCol;s++) { //열의 수만큼 반복

if(button[i][s].getBackground()==Color.blue||button[i][s].getIcon()==iconArr[3]) { //사용가능한 칸 체크
                                                        counter++;

                                                }
                                        }
                                        if(counter>mine)//사용가능한 칸이 지뢰 갯수 이상이면 탈출(아직 찾아야할
지뢰가 있다는 뜻)
                                                break;
                                }
                                if(counter==mine) { //누를 수 있는 버튼이 지뢰 밖에 안 남았다는 뜻
                                        JOptionPane.showMessageDialog(null, "지뢰를 모두 찾았습니다", "승리",
JOptionPane.INFORMATION_MESSAGE);//승리 메시지
                                        for(int i=min;i<iRow;i++) {
                                                for(int s=min;s<iCol;s++)
                                                        button[i][s].setEnabled(false); //버튼을 모두
사용불가로 변경
                                        }
                                        iButton.setIcon(iconArr[1]); //화면 상단 이미지를 승리 이미지로 변경
                                }

                                left=false;right=false;same=false;//마우스를 다시 체크하기위해 false로 초기화
                        }

                }

        }

        public static void main(String[] args) {
                new minesweeper();
        }

}

class Timer implements Runnable{ //쓰레드를 구현하기 위한 Timer
        JLabel timerLabel;
        private int n=0;
        /*
         *  @param  timerLabel 숫자를 보여주기위한 레이블
         *  @param n 시간을 체크하기 위한 변수
         */
        public Timer(JLabel timerLabel) { //생성자
                this.timerLabel=timerLabel;
        }
        @Override
        public void run() {
                int i=1; //플레이한 시간을 체크하기 위한 변수
                while(true) {
                        timerLabel.setText(Integer.toString(n)); //레이블의 텍스트를 n으로 변경
                        n++;//n값 증가
                        try {
                                Thread.sleep(1000);//1000만큼 대기
                        }
                        catch(InterruptedException e) {//예외발생시 종료
                                return;
                        }

                }
        }
}
```

<야구게임 화면>

```
import java.awt.*;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;

import javax.swing.*;

//배열에 3자리 랜덤 숫자 선언
class randomNum {
	int randomArr[];
	public randomNum() {
		randomArr = new int[3];

		randomArr[0] = (int)(Math.random()*9+1); //백의 자리의 숫자(랜덤값 1~9)
		randomArr[1] = (int)(Math.random()*9+1); //십의 자리의 숫자(랜덤값 1~9)
		randomArr[2] = (int)(Math.random()*9+1); //일의 자리의 숫자(랜덤값 1~9)

		//3자리의 숫자에 동일한 값이 들어가지 않도록 설정
		while(randomArr[0]==randomArr[1]) {
			randomArr[1] = (int)(Math.random()*9+1);
		}

		while(randomArr[0]==randomArr[2] || randomArr[1]==randomArr[2]) {
			randomArr[2] = (int)(Math.random()*9+1);
		}
	}

	public int[] getRandomArr() {
		return randomArr;
	}
}


//사용자가 입력한 숫자와 랜덤 숫자 비교 결과를 나타내는 클래스
class resultCheck {
	int strike, ball;
	boolean out;
	public resultCheck(int[] randomArr, int[] inputNum) {
		for(int i=0; i<3; i++) {
			for(int j=0; j<3; j++) {
				if(randomArr[i] == inputNum[j]) { //랜덤 숫자와 사용자의 입력 숫자가 같을 경우 아래의 조건물 실행
					if(i == j) strike++; //랜덤 숫자와 사용자가 입력한 숫자가 같고 위치도 같을 경우 스트라이크 값 증가
					else ball++; //랜덤 숫자와 사용자가 입력한 숫자는 같지만 위치가 다를 경우 볼 값 증가
				}
			}
		}
		//사용자가 입력한 숫자가 랜덤 숫자에 모두 포함되어 있지 않을 경우 아웃값을 true, 그렇지 않을 경우 false 반환
		if(strike == 0 && ball == 0) out = true;
		else out = false;
	}

	public int resultStrike() {
		return strike;
	}

	public int resultBall() {
		return ball;
	}

	public boolean resultOut() {
		return out;
	}
}

class resultPanel extends JPanel{
	int strike = 0;
	int ball = 0;
	boolean out = false;

	void setResult(int strike, int ball, boolean out) {
		this.strike = strike;
		this.ball = ball;
		this.out = out;
	}

	@Override
	public void paintComponent (Graphics g) {
		super.paintComponent(g);

		g.setColor(Color.WHITE);
		g.setFont(new Font("", Font.BOLD, 15));
```

```
                    g.drawString("빈칸에 3자리 숫자를 입력하세요", 80, 90);

                    if(strike == 3) {
                              g.setFont(new Font("", Font.BOLD, 40));
                              g.drawString("!THREE STRIKE!", 30, 180); //랜덤 숫자와 입력한 숫자가 모두 같을 경우
출력되는 문구
                              g.setFont(new Font("", Font.BOLD, 20));
                              g.drawString("숫자를 모두 맞췄습니다!", 70, 220);
                              g.drawString("게임을 재시작해주세요", 75, 260);
                    }
                    else {
                              g.setFont(new Font("", Font.BOLD, 20));

                              //입력한 숫자와 랜덤 숫자의 위치와 숫자가 같을 경우 출력되는 문구
                              if(strike == 0) g.drawString("-", 30, 160);
                              else if(strike == 1) g.drawString("ONE STRIKE", 30, 160);
                              else if(strike == 2) g.drawString("TWO STRIKE", 30, 160);

                              //입력한 숫자와 랜덤 숫자의 숫자는 같지만 위치가 다를 경우 출력되는 문구
                              if(ball == 0) g.drawString("-", 30, 230);
                              else if(ball == 1) g.drawString("ONE BALL", 30, 230);
                              else if(ball == 2) g.drawString("TWO BALL", 30, 230);

                              //입력한 숫자와 랜덤 숫자가 모두 일치하는게 없는 경우 출력되는 문구
                              if(out) g.drawString("OUT", 30, 300);
                              else g.drawString("-", 30, 300);
                    }
          }
}

public class baseballGame extends JFrame implements ActionListener {
          resultPanel rp;
          JTextField text;
          JButton btn;
          randomNum rn;
          JLabel l;
          int strike;

          public baseballGame(randomNum rn) {
                    setTitle("BASEBALL GAME");
                    this.rn = rn;
                    System.out.println(rn.getRandomArr()[0] + "" + rn.getRandomArr()[1] + "" +
rn.getRandomArr()[2]);
                    Container c = getContentPane();

                    c.setLayout(new BorderLayout());

                    rp = new resultPanel();
                    text = new JTextField(15);
                    rp.add(text);

                    btn = new JButton("결과보기");
                    btn.addActionListener(this);
                    btn.setBackground(new Color(230, 180, 78));
                    rp.add(btn);

                    c.add(rp);

                    Color bgColor = new Color(31, 64, 114);
                    rp.setBackground(bgColor);
                    setSize(400, 400);
                    setVisible(true);
                    //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          }

          @Override
          public void actionPerformed(ActionEvent e) {
                    int[] userNum = new int[3];
                    String[] num = text.getText().split("");

                    for(int i=0; i<num.length; i++)
                              userNum[i] = Integer.parseInt(num[i]);

                    resultCheck rc = new resultCheck(rn.getRandomArr(), userNum);
                    System.out.println(rc.resultBall() + "" + rc.resultStrike());
                    rp.setResult(rc.resultStrike(), rc.resultBall(), rc.resultOut());
                    rp.repaint();
          }

          public static void main(String[] args) {
                    new baseballGame(new randomNum());
          }
```

```
}

/*
출처 :
https://it-dolphin.tistory.com/entry/JavaGUI%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-%EC%88%AB%EC%9E%90%EC%95%BC%EA%B5%AC%EA%B2%8C%EC%9E%84
변경사항 : 코드 참고하여 GUI 위주로 변경
*/
```