2015112083 유성근 객체지향언어 프로젝트 - 오목, 지뢰 찾기 구현

전체 코드는 크게 6개의 큰 클래스로 나누어져 있다.

1. GameRun => 제일 처음의 게임 선택 창 객체를 실행시키는 메인 함수가 있고, 파일 입출력에 대한 static 메소드들이 선언되어 있다..
2. GameStartMenu => 오목, 오목 점수 창, 지뢰 찾기, 지뢰 찾기 점수 창을 선택할 수 있는 게임 선택창
3. Mine => 지뢰 찾기가 구현되어 있는 게임 창
4. MineScore => 지뢰 찾기 점수를 확인 할 수 있는 창
5. Omok => 오목 게임이 구현되어 있는 게임 창
6. OmokScore => 오목 점수를 확인할 수 있는 창

이 이외에 내부에는 각각 여러 필요한 클래스들이 선언, 사용되고 있다.

1. GameRun

1) 게임 선택 창 실행

```
import java.io.*;
import java.util.StringTokenizer; // 문자열 토큰 사용
publicclass GameRun {
	publicstaticvoid main(String[] args) {
		GameStartMenu game =new GameStartMenu();
	}
```

게임 선택 창인 GameStartMenu 객체를 만들어 실행하는 main함수가 존재하고, 그 아래에는 파일 입출력에 대한 static 메소드들이 구현되어 있는 모습이다.

2) 파일 입출력 => 오목, 지뢰 찾기 게임을 실행 후 게임을 승리하게 되면 각 정보가 저장된 문자열을 파일에 저장하여 보관하는데, 그 순서는 다음과 같다. 처음에 기존의 파일을 불러와 저장된 순위를 읽고(read)->새로 이긴 정보를 원하는 위치에 넣고(put)-> 다시 바뀐 문자열을 다시 파일에 쓰고(write) 아래에는 차례로 지뢰 찾기, 오목 순으로 읽고, 넣고, 쓰는 코드를 설명하고 있다.

```
////////////////////////////////////////////////////////////////////////////// 지뢰찾기 파일처리

	staticvoid readMine() throws IOException{ // 지뢰 순위 읽기
		Gamerm.mineScore =newString[100]; // 적당히 큰 문자열 배열로 생성

		File file =new File("MineScore.txt"); // 파일 객체 생성
		BufferedReader br =new BufferedReader(new FileReader(file));
		String str =null;
		int n =0;
		while((str = br.readLine()) !=null) { // 지뢰 순위에 저장된 문자열을 한 문장씩 읽어옴
			Gamerm.mineScore[n] = str;
			n++;
		}
		Gamerm.index = n; // 인덱스를 저장
		br.close();
	}
```

지뢰 찾기 구현 부분 내부에 보면 static으로 선언된 변수와 메소드들을 묶어놓은 Gamerm 클래스가 존재하고, 그곳에는 지뢰 찾기 순위표를 파일로부터 불러와 저장할 mineScore 문자열 배열이 존재한다. 그 문자열 배열을 이곳에서 적당한 크기 100으로 생성해 준다. MineScore.txt 파일에서 버퍼리더를 통해 저장된 문자열을 한 문장씩 읽어 와서 mineScore 문자열 배열에 저장하고 총 읽어온 문자열 문장수 (인덱스)를 역시 Gamerm 클래스 내의 index 변수에 저장한다. 마지막으로 버퍼리더를 닫아주고 끝난다.

```
static void putMine(String ns) { // 지뢰 순위 문자열에 새로운 문자열 넣기
		if(Gamerm.mineScore[0] ==null) {Gamerm.mineScore[0] = ns; Gamerm.index++; return;} // nofile이
라면 새로운 순위 넣고 종료
		int num = Gamerm.index;
		int number[] =new int [num]; // 순위만 모아놓은 배열
		for(int i =0; i< num; i++) { // score 배열에 순위와 이름 저장하는 부분
			StringTokenizer tokens =new StringTokenizer(Gamerm.mineScore[i]); // 문자열을 한 줄씩
```

```
불러들여서
                    String tmp = tokens.nextToken("sec"); // 순위(시간)
                    number[i] = Integer.parseInt(tmp); // 순위를 숫자로 변경
                }
                StringTokenizer tokens =new StringTokenizer(ns); // 추가할 문자열의 순위 추출
                String tmp = tokens.nextToken("sec");
                int want = Integer.parseInt(tmp); // 넣고싶은 순위
                int i =0;
                while(i<num) {
                    if(number[0]>=want) {i =0; break;} // 새로운 순위가 제일 처음이면 i(0)으로 설정
                    elseif(number[num-1]<=want) {Gamerm.mineScore[num++] = ns; Gamerm.index++; return;}
// 만약 마지막에 넣어야 하면 마지막에 순위 추가 후 리턴
                    elseif(number[i]<want && number[i+1]>=want) {i = i+1; break;} // 새로운 순위가 가운데에
있으면 넣어야할 위치를 i로 설정
                } // 넣어야 할 위치 i를 찾았음
                num++; // 지뢰 순위 인덱스 하나 증가
                Gamerm.index++;
                for(int j = num; j>i;j--) {
                    Gamerm.mineScore[j] = Gamerm.mineScore[j-1]; // 순위 하나씩 뒤로 미룸
                }
                Gamerm.mineScore[i] = ns; // i위치에 ns 넣음

                number =null; num =0; // 변수 초기화
        }
```

다음은 새로운 문자열(방금 전 게임에서 이긴 사람의 정보)를 mineScore 문자열 배열에 추가하는 함수이다. 처음에 파일을 불러왔는데 순위표에 아무 정보가 없었다면 그 자리에 방금 이긴 사람의 정보를 넣고, 인덱스를 하나 늘려주고 리턴하는 부분이 처음 줄이다. 그 뒤부터는 이미 전에 순위가 존재할 경우의 코드이다. 순위만 모아 놓을 int형 배열 number을 생성해주고, 그 크기는 전에 불러온 순위표의 인덱스 크기이다. 그리고 “##sec @@size, 이름”으로 저장된 문자열에서 “sec”를 기준으로 토큰으로 잘라서 앞의 숫자(끝낸 시간)부분만 string을 숫자로 바꿔 number 배열에 넣는다. 그리고 want라는 int형 변수에 방금 이긴 사람의 시간을 저장한다. 이제 순위를 비교해서 want가 들어갈 자리를 찾는다. 코드 중간에 int i가 바로 want를 넣을 자리이다. I를 찾은 뒤에 I 이후의 원소들을 뒤로 미루고, I에 새로운 문자열을 넣는 부분은 주석으로 설명되어 있다. 그리고 마지막에 안에서 사용한 변수들을 초기화 해준다.

```
                staticvoid WriteMine() throws IOException { // 지뢰 순위 쓰기
                File file =new File("MineScore.txt"); // 파일 객체 생성
                BufferedWriter bf =new BufferedWriter(new FileWriter(file));
                int i =0;
                while(i<Gamerm.index) {
                bf.write(Gamerm.mineScore[i]+"\n"); // 오목 순위에 저장된 문자열을 파일에 써넣는다.
                i++;
                }
                bf.flush(); // 버퍼를 비워줌
                bf.close();
                Gamerm.mineScore =null; Gamerm.index =0; // 사용한 문자열 배열, 인덱스 초기화
        }
```

마지막으로 다시 파일에 써주는 부분이다. 역시 버퍼라이터를 통해 MineScore.txt 파일에 쓰는 것인데, 앞서 put하는 함수에서 업데이트 된 순위 문자열 배열 mineScore과, index가 존재하므로 write 함수를 통해 한 줄씩 파일에 쓰는 간단한 함수이다. 마지막에 버퍼를 비워주고 버퍼를 닫고, 사용한 문자열 배열과 인덱스를 초기화한후 끝난다.

```
////////////////////////////////////////////////////////////////////////// 오목 파일처리
        staticvoid readOmok() throws IOException{ // 오목 순위 읽기
                Gamer.omokScore =new String[100]; // 적당히 큰 문자열 배열로 생성
                File file =new File("OmokScore.txt"); // 파일 객체 생성
                BufferedReader br =new BufferedReader(new FileReader(file));
                String str =null;
```

```
                int n =0;
                while((str = br.readLine()) !=null) { // 오목 순위에 저장된 문자열을 한 문장씩 읽어옴
                        Gamer.omokScore[n] = str;
                        n++;
                }
                Gamer.index = n; // 인덱스를 저장
                br.close();
        }
```

다음은 오목 파일처리이다. 역시 읽기부터 진행되는데, 앞선 지뢰 찾기 읽기 코드와 다른 것이 없다. 차이는 Gamer이라는 클래스가 오목 게임 안에 존재하고 그 안에 존재하는 오목게임의 static 변수인 omokScore 문자열 배열과 index를 사용했다는 점이다.

```
        staticvoid putOmok(String ns) { // 순위표에 새로운 문자열을 더하는 함수
                if(Gamer.omokScore[0] ==null) {Gamer.omokScore[0] = ns; Gamer.index++; return;} // nofile이라면 새로운 순위 넣고 종료
                int num = Gamer.index;
                int number[] =newint [num]; // 순위만 모아놓은 배열
                for(int i =0; i< num; i++) { // score 배열에 순위와 이름 저장하는 부분
                        StringTokenizer tokens =new StringTokenizer(Gamer.omokScore[i]); // 문자열을 한 줄씩 불러들여서
                        String tmp = tokens.nextToken(" "); // 순위
                        number[i] = Integer.parseInt(tmp); // 순위를 숫자로 변경
                }
                StringTokenizer tokens =new StringTokenizer(ns); // 추가할 문자열의 순위 추출
                String tmp = tokens.nextToken(" ");
                int want = Integer.parseInt(tmp); // 넣고싶은 순위
                int i =0;
                while(i<num) {
                        if(number[0]>=want) {i =0; break;} // 새로운 순위가 제일 처음이면 i(0)으로 설정
                        elseif(number[num-1]<=want) {Gamer.omokScore[num++] = ns; Gamer.index++; return;} // 만약 마지막에 넣어야 하면 마지막에 순위 추가 후 리턴
                        elseif(number[i]<want && number[i+1]>=want) {i = i+1; break;} // 새로운 순위가 가운데에 있으면 넣어야할 위치를 i로 설정
                } // 넣어야 할 위치 i를 찾았음
                num++; // 오목 순위 인덱스 하나 증가
                Gamer.index++;
                for(int j = num; j>i;j--) {
                        Gamer.omokScore[j] = Gamer.omokScore[j-1]; // 순위 하나씩 뒤로 미룸
                }
                Gamer.omokScore[i] = ns; // i위치에 ns 넣음

                number =null; num =0; // 변수 초기화
        }
```

방금 이긴 사람 정보 넣는 부분도 다른 것은 없고, 차이는 이렇다. 오목은 순위표에 써넣는 정보가 이긴 수와 이름 뿐이므로 “이긴 수 이름” 이렇게 띄어쓰기를 기준으로 숫자와 이름을 분리할 수 있다. 이 코드에서는 이긴 수를 순위로 삼아서 지뢰 찾기 때와 같은 처리를 통해 방금 이긴 사람의 정보를 문자열 배열 omokScore에 추가한다.

```
        staticvoid WriteOmok() throws IOException { // 오목 순위 쓰기
                File file =new File("OmokScore.txt"); // 파일 객체 생성
                BufferedWriter bf =new BufferedWriter(new FileWriter(file));
                int i =0;
                while(i<Gamer.index) {
                bf.write(Gamer.omokScore[i]+"\n"); // 오목 순위에 저장된 문자열을 파일에 써넣는다.
                i++;
                }
                bf.flush(); // 버퍼를 비워줌
```

```
                bf.close();
                Gamer.omokScore =null; Gamer.index =0; // 사용 변수 초기화
        }
}
```

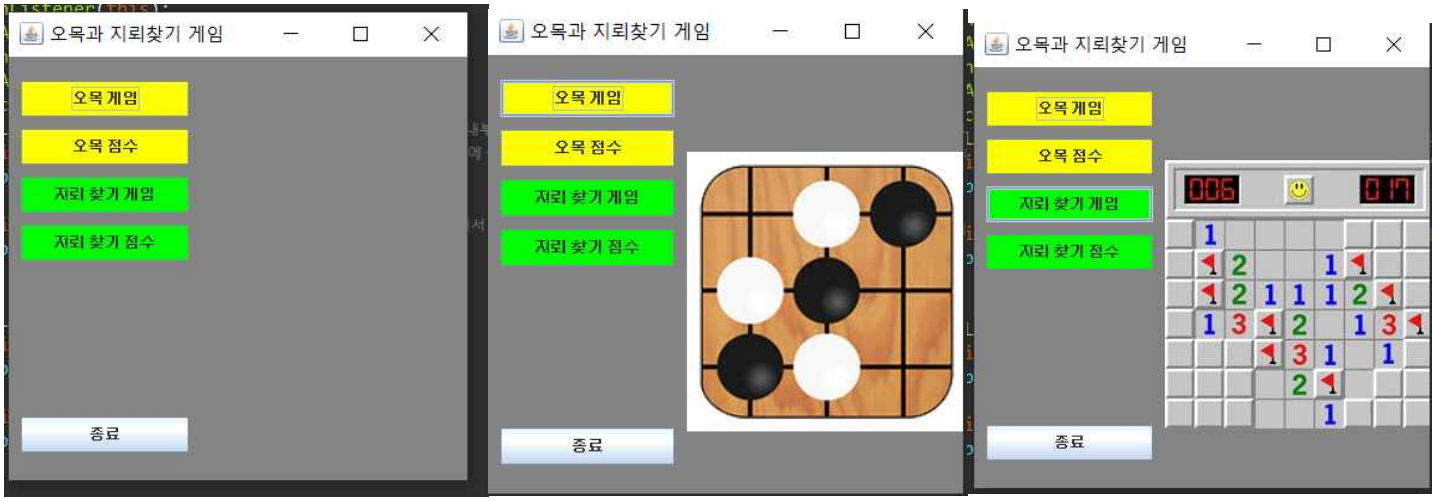
GameRun 클래스의 마지막, 오목 순위 쓰기 메소드이다. 앞서 지뢰 찾기와 역시 다른 점이 없다. 차이점은 사용한 static 변수가 omok 게임 안의 변수라는 것뿐이다.

2. GameStartMenu

```
import javax.swing.*; // 첫 번째 윈도우
import java.awt.*;
import java.awt.event.*;
publicclass GameStartMenu extends JFrame implementsActionListener { // 게임 선택 창

        public JPanel p; // 패널
        JButton omok;
        JButton mine;
        JButton omokScore;
        JButton mineScore;
        JButton exitGame;
        ImageIcon omokImg, mineImg; // 오목 게임, 지뢰찾기 그림을 위한 이미지
        JLabel omoklbl, minelbl; // 오목 게임, 지뢰찾기 그림을 위한 라벨

        GameStartMenu(){
                // 컴포넌트 생성
                p =new JPanel();
                omok =new JButton("오목 게임");
                mine =new JButton("지뢰 찾기 게임");
                omokScore =new JButton("오목 점수");
                mineScore =new JButton("지뢰 찾기 점수");
                exitGame =new JButton("종료");
                omoklbl =new JLabel();
                omokImg =new ImageIcon("omok.jpg");
                minelbl =new JLabel();
                mineImg =new ImageIcon("mine.jpg");
                // 프레임 설정
                this.setTitle("오목과 지뢰찾기 게임");
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                this.setSize(400, 400);
                this.add(p);
                p.setLayout(null); // 절대 위치로 설정
                p.setBackground(Color.gray);
                // 버튼의 위치, 크기, 색상 설정
                omok.setBounds(10, 20, 140, 30);
                omok.setBackground(Color.yellow);
                omokScore.setBounds(10, 60, 140, 30);
                omokScore.setBackground(Color.yellow);
                mine.setBounds(10, 100, 140, 30);
                mine.setBackground(Color.green);
                mineScore.setBounds(10, 140, 140, 30);
                mineScore.setBackground(Color.green);
                exitGame.setBounds(10, 300, 140, 30);
                omoklbl.setBounds(160,50,280,280);
                omoklbl.setIcon(omokImg);
                minelbl.setBounds(160,50,280,280);
```

```
                minelbl.setIcon(mineImg);
                omoklbl.setVisible(false); // 최초에 게임 이미지 안 보임
                minelbl.setVisible(false);
                // 패널에 컴포넌트 부착
                p.add(omok);
                p.add(mine);
                p.add(omokScore);
                p.add(mineScore);
                p.add(exitGame);
                p.add(omoklbl);
                p.add(minelbl);
                // 버튼 이벤트
                omok.addActionListener(this);
                omokScore.addActionListener(this);
                mine.addActionListener(this);
                mineScore.addActionListener(this);
                exitGame.addActionListener(this);
                omok.addMouseListener(newMouseAdapter() { // 마우스 이벤트를 무명 클래스로 작성(마우스 어댑터로
필요한 메소드만 사용)
                        publicvoid mouseEntered(MouseEvent e) { // 마우스가 버튼에 들어가면 게임 이미지 등장
                                omoklbl.setVisible(true);
                        }
                        publicvoid mouseExited(MouseEvent e) { // 마우스가 버튼에서 나가면 게임 이미지 사라짐
                                omoklbl.setVisible(false);
                        }
                });
                mine.addMouseListener(newMouseAdapter() {
                        publicvoid mouseEntered(MouseEvent e) {
                                minelbl.setVisible(true);
                        }
                        publicvoid mouseExited(MouseEvent e) {
                                minelbl.setVisible(false);
                        }
                });

                this.setVisible(true);
        }
@Override
        publicvoid actionPerformed(ActionEvent e) {
                if(e.getSource() == exitGame) { // 종료 버튼 누를 시 게임 종료
                        System.exit(0);
                }
                elseif(e.getSource() == omok) { // 오목 버튼 누르면 오목 게임 시작
                        Omok g1 =new Omok(); // 오목 게임 실행
                        this.dispose(); // 원래의 창은 종료
                }
                elseif(e.getSource() == mine) { // 오목 버튼 누르면 오목 게임 시작
                        Mine g2 =new Mine(); // 오목 게임 실행
                        this.dispose(); // 원래의 창은 종료
                }
                elseif(e.getSource() == omokScore) { // 오목 순위를 보고자 하면
                        OmokScore s1 =new OmokScore();
                }
                elseif(e.getSource() == mineScore) { // 오목 순위를 보고자 하면
```

```
                    MineScore s2 =new MineScore();
                }
        }
}
```



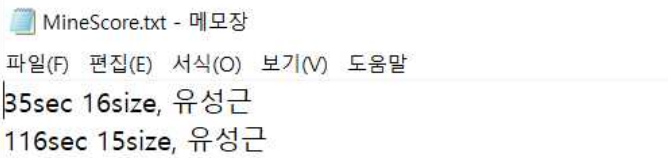
기본 선택 창 설명은 스크린샷을 보면 알 수 있는데, 패널, 버튼 등을 달고 위치 설정, 색상 설정 등은 크게 특이한 것이 없다. 스크린 샷을 찍으면서 마우스 포인터가 같이 찍히지 않아 정확히 표현이 안 되었는데, 오목 게임 또는 지뢰찾기 게임에 마우스 포인터를 가져다 대면 해당 게임 이미지가 옆에 뜨도록 마우스 리스너 (필요한 메소드만 사용하기 위해 정확히는 마우스 어댑터)를 사용해 무명 클래스로 해당 버튼에 마우스를 올리면 게임 이미지가 보이고, 떼면 사라지도록 설정했다. 그리고 actionlistener 인터페이스를 implement해서 사용하므로 역시 무명클래스를 이용해 각 버튼을 눌렀을 때 해당 게임/순위표 객체를 실행시키고 현재의 창은 dispose() 함수를 통해 종료시키도록 설정했다. 종료 버튼을 누를 경우에는 System.exit(0); 코드를 통해 아예 전체 종료 하도록 설정했다.

3. MineScore

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.IOException;
publicclass MineScore extends JFrame {

        JPanel p;
        JTextField word;
        JTextArea sc;
        Font f, f2;
        JButton exit;
        MineScore(){
                this.setTitle("지뢰찾기 순위표");
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                this.setSize(400,700);

                p =new JPanel();
                p.setLayout(null);
                word =new JTextField("순위표");
                f =new Font("San Serif", Font.BOLD, 30);
                f2 =new Font("San Serif", Font.PLAIN, 20);
                sc =new JTextArea();
                exit=new JButton("닫기");

                word.setFont(f);
                word.setEditable(false);
                word.setHorizontalAlignment(JTextField.CENTER); // 텍스트 가운데정렬
```

```
                word.setBounds(130, 10, 120, 40);
                sc.setBounds(10,60,360,500);
                exit.setBounds(130,610,120,40);
                exit.addActionListener(newActionListener() {
                        publicvoid actionPerformed(ActionEvent e) {
                                dispose(); // 창 닫음
                        }
                });
                try {
                        GameRun.readMine(); // 지뢰찾기 순위 정보를 불러오고
                } catch (IOException e) {
                        // TODO Auto-generated catch block, IOException 잡기 위한 try-catch문
                        e.printStackTrace();
                }

                String line =""; // line에 순위를 순서대로 모두 저장
                for(int i =0; i < Gamerm.index; i++) {
                        line += (i+1)+"순위 : "+ Gamerm.mineScore[i] +"\n";
                }
                Gamerm.mineScore =null; Gamerm.index =0; // 변수 초기화
                sc.setText(line); // 텍스트공간에 line 출력
                sc.setFont(f2);
                sc.setBackground(null);

                p.add(word);
                p.add(sc);
                p.add(exit);
                this.add(p);
                this.setVisible(true);
        }
}
```

게임 구현부는 코드가 길고 복잡하므로 상대적으로 코드가 짧고 간단한 순위표 클래스부터 설명하자면, 기본적으로 게임 선택창에서 '지뢰 찾기 점수' 버튼을 누르면 다음과 같은 창이 뜬다.



(MineScore.txt 내용)

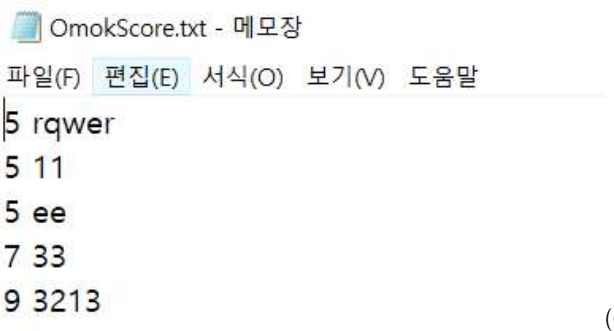
순위표라고 적힌 텍스트 필드 컴포넌트 설정과 닫기를 누르면 무명클래스를 통한 ActionListener로 현재 순위 창을 닫는 것은 특별한 것이 없다. 가운데에 텍스트 공간(JTextArea)가 존재하는데, 이곳에 텍스트 파일로부터 문자열을 불러와 표시해준다. 코드를 살펴보자면, GameRun 클래스 안의 static 메소드 readMine()를 통해 순위를 불러온다. 이 때 IOException을 잡기 위해 try-catch문을 사용한다. 이제 불러온 순위 문자열 배열을 line이라는 문자열에 띄어쓰기를 통해 한 문장씩 더해 넣는다. 사용한 변수는 초기화하고 line 문자열을 텍스트 공간에 넣어주면 순위가 표시되게 된다.

4.OmokScore

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.IOException;
publicclass OmokScore extends JFrame {

	JPanel p;
	JTextField word;
	JTextArea sc;
	Font f, f2;
	JButton exit;
	OmokScore(){
		this.setTitle("오목 순위표");
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setSize(400,700);

		p =new JPanel();
		p.setLayout(null);
		word =new JTextField("순위표");
		f =new Font("San Serif", Font.BOLD, 30);
		f2 =new Font("San Serif", Font.PLAIN, 20);
		sc =new JTextArea();
		exit=new JButton("닫기");

		word.setFont(f);
		word.setEditable(false);
		word.setHorizontalAlignment(JTextField.CENTER); // 텍스트 가운데정렬

		word.setBounds(130, 10, 120, 40);
		sc.setBounds(10,60,360,500);
		exit.setBounds(130,610,120,40);
		exit.addActionListener(newActionListener() {
			publicvoid actionPerformed(ActionEvent e) {
				dispose(); // 창 닫음
			}
		});

		try {
			GameRun.readOmok(); // 오목 순위 정보를 불러오고
		} catch (IOException e) {
			// TODO Auto-generated catch block, IOException 잡기 위한 try-catch문
			e.printStackTrace();
		}
		String line =""; // line에 순위를 순서대로 모두 저장
		for(int i =0; i < Gamer.index; i++) {
			line += (i+1)+"순위 (이긴 수, 이름) : "+ Gamer.omokScore[i] +"\n";
		}
		Gamer.omokScore =null; Gamer.index =0; // 변수 초기화
		sc.setText(line); // 텍스트공간에 line 출력
		sc.setFont(f2);
		sc.setBackground(null);
```

```
                p.add(word);
                p.add(sc);
                p.add(exit);
                this.add(p);
                this.setVisible(true);
        }
}
```

오목 순위표는 지뢰 찾기 순위표와 다른 것이 하나도 없으므로 설명을 위의 지뢰 찾기 순위표 설명으로 대체한다. 오목 순위표를 눌렀을 때의 화면은 다음과 같다.



(OmokScore.txt 내용)

5. Mine

게임을 구현하면서 가장 문제가 되었던 것이 특정 버튼을 기점으로 팔방을 확인할 때에 (0,0)의 인덱스에서 좌, 상, 좌상 등에 접근할 때마다 ArrayIndexOutOfBounds 오류가 뜨는 것이었다. 그래서 아예 처음에 사이즈를 입력받을 때 입력받은 사이즈보다 2가 큰 사이즈를 게임 사이즈로 설정해서 가장자리의 버튼을 사용하지 않도록 설정했다.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random; // 랜덤을 사용하기 위해
import java.io.IOException; // IOException을 위해
import javax.swing.Timer; // 타이머 기능을 위해
class Gamerm{ // 지뢰 게임 static 변수들

        staticString mineScore[]; // 지뢰찾기 점수 기록
        staticint index; // 지뢰찾기 점수 인덱스
        staticint size; // 지뢰찾기 판의 사이즈
        staticint time; // 깬 시간
        staticboolean mine[][]; // 지뢰의 위치를 저장할 배열 있으면 true, 없으면 false
        staticint mineNum; // 지뢰의 개수를 저장할 변수
        staticint reNum; // 게임 재시작 시 지뢰 개수를 저장할 변수
        staticboolean finish=false; // 게임 끝 여부, 최초 false로 초기화
        staticboolean gameover =false; // 게임오버 여부, 최초 false로 초기화
        staticString name; // 이름을 입력받을 변수
        staticint isMine(int i, int j) { // 팔방을 검사해서 지뢰의 개수를 리턴하는 함수
                int n =0; // 지뢰의 개수를 셀 변수
                if(i>=1&& j>=1&& i<size-1&& j<size-1) { // 배열의 index out of bounds를 방지 가장 자리 제외
                        if(mine[i-1][j-1]) n++;
                        if(mine[i-1][j]) n++;
```

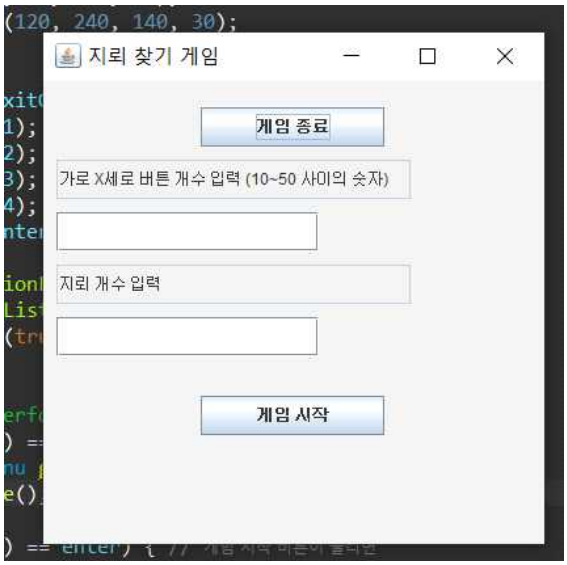
```
                        if(mine[i-1][j+1]) n++;
                        if(mine[i][j-1]) n++;
                        if(mine[i][j+1]) n++;
                        if(mine[i+1][j-1]) n++;
                        if(mine[i+1][j]) n++;
                        if(mine[i+1][j+1]) n++;
                }
                return n;
        }
        staticvoid clear() { // static 변수 초기화 함수
                for(int i =0; i<size;i++) {
                        for(int j =0; j<size; j++) {
                                mine[i][j] =false;
                        }
                }
                time =0;
                size =0;
                mineNum =0;
                reNum =0;
                finish =false;
                gameover =false;
                name =null;
        }
}
```

이 부분은 지뢰 찾기 구현부에서 가장 상단의 모습인데, 타이머와 랜덤 등을 위한 유틸 등을 import해주었고, 지뢰 찾기 게임에 필요한 static 변수, 메소드 등을 갖는 클래스 Gamerm의 선언이 존재한다. 변수들의 용도는 모두 주석으로 설명되어 있고, 특정 인덱스에서 팔방을 방문해 지뢰의 개수를 새서 그 개수를 리턴하는 isMine()함수와 static 변수들을 초기화하는 clear() 함수가 존재한다.

```
publicclass Mine extends JFrame implementsActionListener {

        JPanel inputMine; // 사이즈, 지뢰 개수 입력받는 패널
        JButton exitGame;
        JButton enter; // 입력한 값으로 게임 생성
        JTextField f1,f2, f3, f4, f5, f6;

        Mine(){
                // 컴포넌트 생성
                inputMine =new JPanel();
                exitGame =new JButton ("게임 종료");
                f1 =new JTextField("가로 X세로 버튼 개수 입력 (10~50 사이의 숫자)");
                f2 =new JTextField("지뢰 개수 입력");
                f3 =new JTextField(); // 버튼 수 입력받음
                f4 =new JTextField(); // 지뢰 개수 입력
                enter =new JButton("게임 시작");

                // 프레임 설정
                this.setTitle("지뢰 찾기 게임");
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                this.setSize(400, 400);
                this.add(inputMine);
                inputMine.setLayout(null);

                // 컴포넌트 위치 설정
                f1.setEditable(false);
                f2.setEditable(false);
```

```
                exitGame.setBounds(120, 20, 140, 30);
                f1.setBounds(10,60, 270, 30);
                f2.setBounds(10,140, 270, 30);
                f3.setBounds(10,100, 200, 30);
                f4.setBounds(10,180, 200, 30);
                enter.setBounds(120, 240, 140, 30);

                //패널에 컴포넌트 부착
                inputMine.add(exitGame);
                inputMine.add(f1);
                inputMine.add(f2);
                inputMine.add(f3);
                inputMine.add(f4);
                inputMine.add(enter);
                exitGame.addActionListener(this); // 게임 종료
                enter.addActionListener(this); // 게임 실행
                this.setVisible(true);
        }
        @Override
        publicvoid actionPerformed(ActionEvent e) {
                if(e.getSource() == exitGame) { // 종료 버튼이 눌리면
                        GameStartMenu game =new GameStartMenu();
                        this.dispose(); // 입력창 종료
                }
                if(e.getSource() == enter) { // 게임 시작 버튼이 눌리면
                        int size =0; int mineNum =0;
                        size = Integer.parseInt(f3.getText())+2; // 입력받은 사이즈 저장, 가장자리 포함 2개 더함
                        mineNum = Integer.parseInt(f4.getText()); // 입력받은 지뢰 개수 저장
                        MineFrame gf =new MineFrame(size, mineNum); // 매개 변수를 갖는 지뢰 찾기 게임 생성
                        this.dispose(); // 입력창 종료
                }
        }
}
```

게임 선택창에서 '지뢰찾기'를 선택하면 처음 뜨는 창이 바로 이 창이다.



게임 종료는 클릭하면 현재 창을 종료하고 다시 게임 선택창을 띄우는 것으로 계속 작성해왔던 코드이고, 설명이 적힌 텍스트필드들은 해당 문자열로 초기화하고, setEditable을 false로 설정해서 수정 불가능하게 만들었다. 입력할 수 있게 되어있는 텍스트 필드 내에 각각 숫자를 입력하면 '게임 시작'버튼을 누르는 순간 입력받은 문자열을 숫자로 바꿔 해당 사이즈와 지뢰 개수를 매개변수로 갖는 실제 게임 클래스 MineFrame 객체가 생성된다. 그 후 입력창은 닫는다. 특이 사항으로는 위에서도 설명했듯이 배열 범위 벗어나는 오류를 방지하기 위해 size는 2를 더한 값으로 설정한다.

```
class MineFrame extends JFrame implementsActionListener{

	JPanel mineP; // 지뢰찾기 판
	JPanel info; // 정보가 표시될 부분
	JButton bt[][];
	JLabel mineNum; // 지뢰의 개수가 표시될 부분
	JButton face; // 스마일 버튼
	JButton exit; // 종료 버튼
	int fakeNum; // 표시용 지뢰 개수
	int realSize; // 실제 사이즈


	//////////////////////////// 이미지 아이콘
	ImageIcon sm;// 스마일 아이콘
	ImageIcon sunsm; // 기본 선글라스 낀 스마일
	ImageIcon minenormal; // 기본적인 이미지
	ImageIcon flag; // 깃발, 우클릭 한 번 시
	ImageIcon question; // 물음표, 우클릭 두 번 시
	ImageIcon unfoundmine; // 못 찾은 지뢰
	ImageIcon clickedmine; // 지뢰를 누름, 게임오버
	ImageIcon nomine; // 지뢰 없는 부분
	ImageIcon gameover; // 게임오버시 표시될 우는 스마일
	ImageIcon wrong; // 잘못 찾은 부분
	ImageIcon m1, m2, m3, m4, m5, m6, m7, m8; // 지뢰  숫자

	JLabel time; // 시간이 표시될 라벨
	int sec; // 시간
	Timer timer;
	boolean firstClick =false; // 처음 클릭되는 것을 파악
```

여기서부터 실제 지뢰 찾기 게임 프레임인 MineFrame 클래스이며, 컴포넌트, 이미지, 변수 등이 선언된 부분이다.

```
	void numPop(int indexa, int indexb) { // 인덱스를 매개변수로 받아서 해당 숫자로 이미지를 바꿔주는 함수
		if(Gamerm.mine[indexa][indexb]) return;  // 해당 인덱스가 지뢰이지 않은 경우에 한 해서(지뢰 이미지가 바뀌면 안 되므로)
		int num = Gamerm.isMine(indexa, indexb);
		switch(num) { // 주변 지뢰 개수에 따라서 숫자로 표시해줌
		case1:
			bt[indexa][indexb].setIcon(m1); break;
		case2:
			bt[indexa][indexb].setIcon(m2); break;
		case3:
			bt[indexa][indexb].setIcon(m3); break;
		case4:
			bt[indexa][indexb].setIcon(m4); break;
		case5:
			bt[indexa][indexb].setIcon(m5); break;
		case6:
			bt[indexa][indexb].setIcon(m6); break;
		case7:
			bt[indexa][indexb].setIcon(m7); break;
		case8:
			bt[indexa][indexb].setIcon(m8); break;
		}
```

```
        }
```

MineFrame 클래스 안에는 몇 가지 메소드들이 존재하는데, 이 메소드는 주위에 지뢰가 있는 버튼을 누르면 그 인덱스의 버튼을 isMine() static 함수로 알아낸 숫자의 이미지로 바꿔주는 메소드이다.

```
        void popAround(int i, int j) {
                if(i>=1&& j>=1&& i<realSize && j<realSize) { // 가장자리 제외
                        bt[i][j].setIcon(nomine); // 자기 위치를 지뢰 없는 이미지로 바꿈
                        if(Gamerm.isMine(i-1, j-1) >0) numPop(i-1, j-1); // 팔방을 지뢰없는 이미지 혹은 숫자 패널로 바꿈
                        elseif(!Gamerm.mine[i-1][j-1]) bt[i-1][j-1].setIcon(nomine);
                        if(Gamerm.isMine(i-1, j) >0) numPop(i-1, j);
                        elseif(!Gamerm.mine[i-1][j]) bt[i-1][j].setIcon(nomine);
                        if(Gamerm.isMine(i-1, j+1) >0) numPop(i-1, j+1);
                        elseif(!Gamerm.mine[i-1][j+1]) bt[i-1][j+1].setIcon(nomine);
                        if(Gamerm.isMine(i, j-1) >0) numPop(i, j-1);
                        elseif(!Gamerm.mine[i][j-1]) bt[i][j-1].setIcon(nomine);
                        if(Gamerm.isMine(i, j+1) >0) numPop(i, j+1);
                        elseif(!Gamerm.mine[i][j+1]) bt[i][j+1].setIcon(nomine);
                        if(Gamerm.isMine(i+1, j-1) >0) numPop(i+1, j-1);
                        elseif(!Gamerm.mine[i+1][j-1]) bt[i+1][j-1].setIcon(nomine);
                        if(Gamerm.isMine(i+1, j) >0) numPop(i+1, j);
                        elseif(!Gamerm.mine[i+1][j]) bt[i+1][j].setIcon(nomine);
                        if(Gamerm.isMine(i+1, j+1) >0) numPop(i+1, j+1);
                        elseif(!Gamerm.mine[i+1][j+1]) bt[i+1][j+1].setIcon(nomine);
                }
        }
```

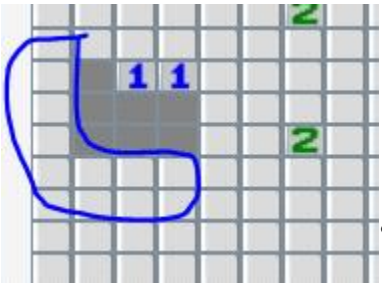
이 메소드는 지뢰도 아니고, 주변에 지뢰도 없는 버튼을 눌렀을 때 자신은 지뢰 없는 이미지로 바꾸고 8방을 검색해서 주변에 지뢰가 있으면 숫자 이미지로, 주변에 지뢰도 없으면 지뢰 없는 이미지로 바꾸는 메소드이다. (Gamerm.mine[][]은 해당 인덱스에 지뢰가 있으면 true, 없으면 false이다.)

```
        void pop1(int i, int j) { // 자신 포함 8방을 터뜨림
                if(i>=1&& j>=1&& i<realSize && j<realSize) { // 가장자리 제외
                        if(Gamerm.isMine(i, j) >0) {numPop(i,j); return;} // 숫자 패널을 발견하면 숫자를 표시하고 함수를 나감
                        popAround(i,j);
                }
        }
```

숫자 판을 발견하면 숫자판만 표시하고 함수를 나가도록 설정했고, 그렇지 않은 경우에는 바로 위의 popAround 메소드를 실행하도록 설정했다.

```
int popStop() {//////////////////////////////////////없을 때까지 미처리 개수 리턴
                int n =0;
                for(int i =1; i < realSize;i++) {
                        for(int j =1; j < realSize;j++) {
                                if(bt[i][j].getIcon() == minenormal && bt[i-1][j-1].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i-1][j].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i-1][j+1].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i][j+1].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i+1][j+1].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i+1][j].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i+1][j-1].getIcon() == nomine) n++;
                                if(bt[i][j].getIcon() == minenormal && bt[i][j-1].getIcon() == nomine) n++;
                        }
                }
                return n;
        }
```

뒤에서 pop1을 한 번 시행하면 팔방과 자기 자신만 이미지가 바뀌고 함수가 끝난다. 때문에 바뀌다 만 부분(자기 자신은 아직 보통 지뢰찾기 이미지 인데(지뢰가 있는지 숫자판인지, 아무것도 없는지 모름), 팔방 중에 지뢰가 없다고 알게 된 부분이 있으면) 그런 버튼의 개수를 세서 리턴하는 메소드이다.

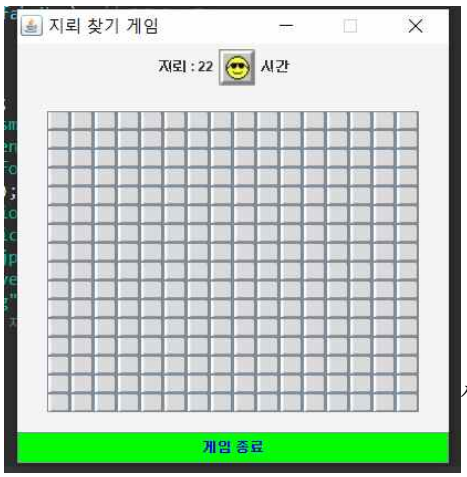


옆의 파란 부분과 같은 곳

```
MineFrame(int a, int b){ // 사이즈와 지뢰 개수를 매개변수로 받는 생성자

                sec =0; // 0부터 시간을 셈
                timer =new Timer(1000, newActionListener () { // 타이머 설정
                        publicvoid actionPerformed(ActionEvent e) {
                                time.setText("시간 : "+ Integer.toString(sec));
                                sec +=1; // 1초마다 1씩 늘어남
                        }
                }); // 타이머 설정

                realSize = a-1; // 실제 사이즈

                Gamerm.size = a; // 지뢰찾기 판 사이즈저장
                Gamerm.mineNum = b; // 지뢰 개수 저장
                Gamerm.reNum = b;
                fakeNum = b;
                Gamerm.mine =newboolean[a][a]; // 사이즈 크기의 지뢰가 매설될 가능성이 있는 배열 생성

                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 종료 버튼
                this.setTitle("지뢰 찾기 게임");
                this.setLayout(new BorderLayout()); // BorderLayout으로 프레임 설정
                this.setSize(a*22,a*23); // 매개변수로 받은 a 값에 따라 프레임 크기를 조절
                this.setResizable(false); // 프레임 사이즈 변경 불가

                mineP =new JPanel(); // 지뢰 페널
                mineP.setLayout(new GridLayout(a,a)); // a*a 개의 버튼을 갖는 패널
                bt=new JButton[a][a]; // 버튼의 ㅁ행 a열의 이차원 배열 생성
                info =new JPanel();
                exit=new JButton("게임 종료");
                mineNum =new JLabel("지뢰 : "+ fakeNum); // 지뢰 개수 표시
                time =new JLabel("시간       ");
                face =new JButton();

                sm =new ImageIcon("smile.jpg"); // 스마일 이미지
                sunsm =new ImageIcon("sunglasssm.jpg"); // 기본 선글라스 낀 스마일 이미지
                minenormal =new ImageIcon("minenormal.jpg"); // 기본 지뢰찾기 판
                unfoundmine =new ImageIcon("unfoundmine.jpg"); // 찾은 지뢰 이미지
                flag =new ImageIcon("flag.jpg"); // 깃발표시
                question =new ImageIcon("question.jpg"); // 물음표 표시
                clickedmine =new ImageIcon("clickedmine.jpg"); // 지뢰 누름 게임오버
                nomine =new ImageIcon("nomine.jpg"); // 지뢰 없는 부분
                gameover =new ImageIcon("gameover.jpg"); // 게임오버
                wrong =new ImageIcon("wrong.jpg"); // 잘못 찾은 지뢰
```

```
m1 =new ImageIcon("1.jpg"); // 지뢰 숫자
m2 =new ImageIcon("2.jpg");
m3 =new ImageIcon("3.jpg");
m4 =new ImageIcon("4.jpg");
m5 =new ImageIcon("5.jpg");
m6 =new ImageIcon("6.jpg");
m7 =new ImageIcon("7.jpg");
m8 =new ImageIcon("8.jpg");
```

이제 생성자 부분인데, 여기서 매개변수 a가 사이즈(입력한 사이즈보다 2가 크다.), b가 지뢰의 개수이다. 타이머 부분은 sec 변수가 1초마다 1씩 증가해서 게임 프레임 좌측 상단에 sec 변수를 time이라는 라벨을 통해 문자열을 숫자로 변경해 표시해준다. 그리고 나머지는 컴포넌트들을 생성하고 설정하는 부분이다. 특이 사항으로는 변수 realSize는 가장자리를 사용하지 않기 때문에 실제 사용되는 사이즈가 사이즈 a보다 1작은 수임을 나타내고, reNum은 스마일 아이콘으로 게임을 재시작 할 때를 위해 처음 지뢰 개수를 저장하는 변수이다. 또 fakeNum은 실제 지뢰 개수가 아닌 깃발 표시를 했을 때 줄어드는 지뢰 개수이다. 때문에 -개수까지 내려갈 수 있다.

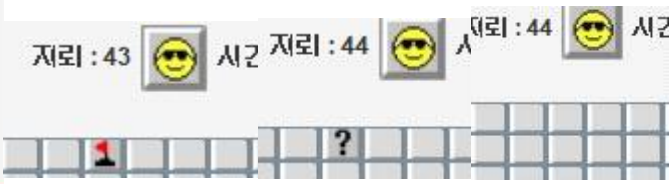


생성자의 매개변수로 인해 생겨나는 프레임

```
for(int i =0; i <a; i++) { // 행만큼 반복
        for(int j =0; j <a; j++) { // 열만큼 반복
                bt[i][j] =new JButton(); // 버튼 생성
                mineP.add(bt[i][j]); // 패널에 부착
                bt[i][j].setIcon(minenormal); // 지뢰찾기 판 이미지를 부착
                        bt[i][j].addActionListener(this); // 버튼에 이벤트 추가
                        Gamerm.mine[i][j] =false;// 지뢰를 모두 false로 초기화
                        bt[i][j].addMouseListener(newMouseAdapter() { /////// 마우스 우클릭 시
                           publicvoid mousePressed(MouseEvent e) {
                            if(e.getModifiers() == MouseEvent.BUTTON3_MASK) {  // 우클릭을 실행하면
                                ////////////// 예외처리
                                if(Gamerm.finish|| Gamerm.gameover) return; // 게임을 이기거나 지면 오른쪽 클릭
금지
                                int  indexa =0, indexb =0; // 눌린 버튼의 인덱스를 저장할 변수
                                for(int i =0 ; i <a ;i++) {
                                        for(int j =0; j< a; j++) {
                                                if(e.getSource() == bt[i][j]) { indexa = i; indexb = j;} // 눌린 버
튼의 인덱스 파악
                                        }
                                }
                                if(bt[indexa][indexb].getIcon() == minenormal) { // 기본 이미지에서 누를 때
                                        bt[indexa][indexb].setIcon(flag); // 이미지를 깃발로 바꾸고
                                        fakeNum--; // 지뢰 개수 하나 줄이고 표시
                                        mineNum.setText("지뢰 : "+ fakeNum);
                                        ///////// 만약 진짜 지뢰를 발견하면
                                        if(Gamerm.mine[indexa][indexb]) {
                                                Gamerm.mineNum--; // 진짜 지뢰 개수를 감소
```

```
                                        ////////////////////////////////////// 게임 승리
                                                if(Gamerm.mineNum ==0&& fakeNum ==0) { // 진짜 남은 지뢰
개수가 0이고 표시된 지뢰 수도 0이면 승리
                                                        timer.stop(); // 타이머 멈춤
                                                        face.setIcon(sm); // 얼굴 웃는 표시로 변경
                                                        exit.setText("지뢰를 모두 찾았습니다. 아무 버튼이나 클릭
하십시오."); // 게임 끝나면 버튼 클릭을 유도
                                                        exit.setBackground(Color.RED); // 색 변경으로 시선을
끔
                                                        exit.setForeground(Color.WHITE);
                                                        Gamerm.finish =true; // finish가 true가 됨. 버튼 이벤
트에 finish일 경우를 추가해놓음.
                                                        }
                                                }
                                        }
                                        elseif(bt[indexa][indexb].getIcon() == flag) { // 깃발일 때 다시 클릭하면
                                            bt[indexa][indexb].setIcon(question); // 이미지를 물음표로 바꾸고
                                             fakeNum++; // 지뢰 개수 하나 늘리고 표시
                                             mineNum.setText("지뢰 : "+ fakeNum);
                                                        ///////// 만약 진짜 지뢰를 취소하면
                                                if(Gamerm.mine[indexa][indexb]) {
                                                        Gamerm.mineNum++; // 진짜 지뢰 개수를 증가
                                                        }
                                                }
                                                elseif(bt[indexa][indexb].getIcon() == question) {
                                                        bt[indexa][indexb].setIcon(minenormal); // 기본 이미지
로 다시 변경
                                                }
                                            }
                                         }
                                });
                                if(i ==0|| i == realSize || j ==0|| j == realSize) { // 가장자리의 버튼들은
                                        bt[i][j].setEnabled(false); // 사용 불가 처리
                                        bt[i][j].setVisible(false); // 보이지 않게 처리
                                }
                        }
                }
```

버튼 생성, 부착, actionListener 부착, 오른쪽 클릭에 대한 코드들이다. 우선 입력받은 size+2의 크기대로 버튼을 생성한다. 우클릭을 실행했을 때, 총 세 가지 경우로 나눌 수 있는데, 1> 기본 이미지에서 누를 때 2> 깃발 상태에서 누를 때 3> 물음표 상태에서 누를 때이다. 기본 이미지에서 누르면 이미지를 깃발로 바꾸고 fakeNum(보여지는 지뢰 수)는 -1을 해준다. 그리고 진짜 지뢰를 발견했으면 진자 Gamerm.minNum을 하나 줄여준다. 게임 승리 조건은 mineNum도 0이고 fakeNum도 0이면 승리이다. 타이머를 멈추고 얼굴을 스마일로 바꾼 뒤 finish 변수를 true로 바꿔 승리했음을 알린다. 승리하면 아래와 같이 종료 버튼이 해당 문구로 바뀌게 된다.



이후 적힌 글대로 아무 버튼이나 누르게 되면 actionListener에서 finsh 조건이 달성되면서 게임 승리 이후의 작업이 진행된다. 여기까지가 기본 이미지에서 오른쪽 클릭할 때이고, 깃발 이미지가 된 후 다시 오른쪽 클릭하면 물음표로 바뀌고 fakeNum이 하나 늘어난다. 그리고 만약 진짜 지뢰를 다시 클릭한거면 mineNum도 하나 늘어난다. 물음표일 때는 그냥 이미지만 다시 기본 이미지로 바뀐다. 마지막에 가장자리에 존재하는 버튼들은 사용불가에 보이지 않게 처리한다.

```
                exit.addActionListener(this); // 종료 버튼 이벤트추가
                face.addActionListener(this); // 얼굴 버튼 이벤트 추가
                exit.setBackground(Color.green);
                exit.setForeground(Color.blue);
                face.setIcon(sunsm); // 선글라스 스마일 그림 버튼에 추가
                face.setBackground(null); // 스마일 버튼 그림만 보이도록 설정
                face.setBorder(null);

                info.add(mineNum);
                info.add(face);
                info.add(time);

                add(info, BorderLayout.NORTH); // 정보 부분을 프레임 위쪽에 부착
                add(mineP, BorderLayout.CENTER); // 패널을 프레임 중앙에 부착
                add(exit, BorderLayout.SOUTH); // 프레임 아래에 종료 버튼 추가
                setVisible(true); // 가시화

                // 지뢰 생성 부분
                Random random =newRandom(); // 랜덤 객체 생성

                        for(int i =0; i<b;i++) { // 지뢰 개수만큼 반복
                                int x = random.nextInt(a); // 랜덤으로 0부터 a-1 까지의 숫자(인덱스)를 갖는다.
                                int y = random.nextInt(a);
                                if(Gamerm.mine[x][y] || x ==0|| y ==0|| x == realSize || y == realSize) { // 이미
그 위치에 지뢰가 존재하거나 가장자리면 i를 하나 감소
                                        i--;
                                }
                                Gamerm.mine[x][y] =true; // 지뢰의 위치를 저장
                                if( x ==0|| y ==0|| x == realSize || y == realSize) Gamerm.mine[x][y] =false;
                        }
```

이 코드는 앞쪽은 각종 컴포넌트 설정 부분이고, 중요한 것은 뒤의 지뢰 생성 부분이다. 랜덤 객체를 생성해 지뢰 개수만큼 반복해서 어떠한 작업을 행하는데, x와 y인덱스를 랜덤으로 받는다. 만약 이미 그 자리에 지뢰가 존재하거나 가장자리면 다시 I를 하나 감소시켜서 지뢰 개수를 맞춘다. 해당 자리에는 지뢰를 위치시키고, 가장자리는 지뢰가 무조건 없도록 다시 설정한다.

```
@Override
        publicvoid actionPerformed(ActionEvent e) {
                //////////////////////////// 예외처리
                if(Gamerm.gameover) return; // 게임오버 당하면 버튼 이벤트 실행 금지
                if(!firstClick) {timer.start(); firstClick =true;} // 처음 클릭한 순간부터 타이머 실행
                ////////////////////////////게임 종료 클릭시
                if(e.getSource() ==exit) {
                        dispose();
                        Gamerm.clear(); // 게임 정보 초기화
                        GameStartMenu a =new GameStartMenu();
                }
                //////////////////////////// 스마일 아이콘 클릭시
                if(e.getSource() == face) {
                        int n1 = Gamerm.size; int n2 = Gamerm.reNum; // 사이즈와 지뢰 개수 저장 후 매개변수로
넣어줌
                        Gamerm.clear(); // 게임 초기화 후
```

```
                dispose(); // 현재 창 종료 후
                MineFrame gf =new MineFrame(n1, n2); // 매개 변수를 갖는 지뢰 찾기 게임 생성
        }
        //////////////////버튼을 눌렀을 때 인덱스 찾는 부분
        int indexa =0; int indexb =0; // 인덱스 저장 변수
        for(int i =0; i <Gamerm.size;i++) {
                for(int j =0; j< Gamerm.size;j++) {
                        if(e.getSource() == bt[i][j]) {
                                indexa = i; indexb = j; // 인덱스 저장
                        }
                }
        }
        ////////////////////// 예외처리 2
        if(bt[indexa][indexb].getIcon() == nomine &&!Gamerm.finish) return; // 게임이 안 끝났는데 지뢰 없는
곳 확인된 곳을 클릭시에는 버튼 이벤트 실행 금지
        if(bt[indexa][indexb].getIcon() == flag) return; // 깃발인 부분은 클릭하면 지뢰가 눌리므로 이벤트 실행
금지
        ///////////////// 지뢰를 눌렀을 때
        if(Gamerm.mine[indexa][indexb]) {
                Gamerm.gameover =true; // 게임오버 표시
                face.setIcon(gameover); // 얼굴 죽은 스마일로 바꿈
                timer.stop(); // 타이머 멈춤
                for(int i =0; i < Gamerm.size;i++) {
                        for(int j =0; j<Gamerm.size;j++) {
                                if(bt[i][j].getIcon() == flag &&!Gamerm.mine[i][j]) { // 지뢰가 아닌데 깃발
친 부분
                                        bt[i][j].setIcon(wrong); // 잘못 찾음
                                }
                                if(Gamerm.mine[i][j] && bt[i][j].getIcon() != flag) { // 찾지 못한 지뢰(깃발
안 친 지뢰)
                                        bt[i][j].setIcon(unfoundmine); // 못 찾음
                                } bt[indexa][indexb].setIcon(clickedmine); // 클릭한 지뢰는 따로 표시
                        }
                }
                GameOver n =new GameOver(this); // , 현재 게임 객체를 매개변수로 넘겨서 게임 오버 창 띄
움
        }
```

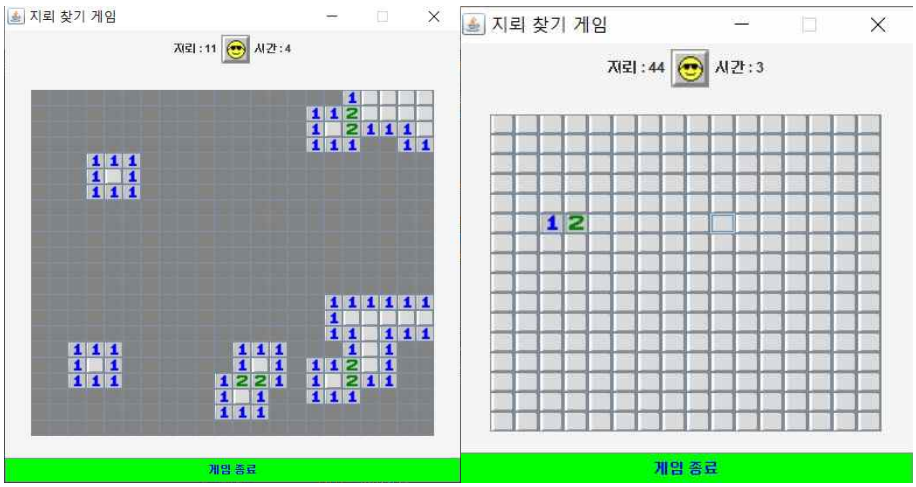
여기서부터는 버튼을 클릭했을 때의 버튼 이벤트이다. 게임 오버당하면 버튼을 누를 수 없고, 처음 클릭하는 순간 static 변수 firstClick이 true가 되면서 timer가 실행된다. 이후부터는 firstClick이 true이므로 게임이 끝나기 전까지 타이머가 재실행될 일은 없다. exit을 클릭하면 static 정보들을 초기화하고, 창을 끄고 다시 게임 선택화면으로 나가며 스마일 얼굴을 누르면 게임사이즈와 아까 reNum에 저장한 초기 지뢰 개수를 이용해 새로 다시 게임을 생성하고 원래 창은 닫는다. 이후 눌린 인덱스를 indexa, indexb에 저장하는 부분, 추가적인 예외처리가 있다. 지뢰를 눌렀을 경우 (게임 오버)에서 중요한 부분은 지뢰가 아닌데 깃발 친 부분은 wrong 이미지로 바꿔주고, 찾지 못한 지뢰는 못찾은 이미지로 바꾸고 클릭한 지뢰(게임 오버된 이유)는 빨간색 칸으로 바꿔준다. 그리고 GameOver이라는 새로운 창 객체를 실행시킨다.(현재 게임 객체를 매개 변수로 넘김)

누른버튼 빨간색, 잘못 찾은 버튼 엑스표, 못찾은 지뢰 표시

```
////////////////// 근처에 지뢰가 없는 곳을 눌렀을 때
                if(Gamerm.isMine(indexa, indexb) ==0&&!Gamerm.mine[indexa][indexb]) {
////////////////////////////////////// 만약 팔방이 다 숫자판이면
                        if(Gamerm.isMine(indexa-1, indexb-1) >0&& Gamerm.isMine(indexa-1, indexb) >0&&
Gamerm.isMine(indexa-1, indexb+1) >0
                                && Gamerm.isMine(indexa, indexb-1) >0&& Gamerm.isMine(indexa,
indexb+1) >0
                                && Gamerm.isMine(indexa+1, indexb-1) >0&& Gamerm.isMine(indexa+1,
indexb) >0
                                && Gamerm.isMine(indexa+1, indexb+1) >0) {
                            bt[indexa][indexb].setIcon(nomine); // 가운데는 빈칸으로 처리하고 팔방을 다 숫자
판으로 바꿔준 후
                            numPop(indexa-1, indexb-1); numPop(indexa-1, indexb); numPop(indexa,
indexb-1); numPop(indexa, indexb+1);
                            numPop(indexa+1, indexb-1); numPop(indexa+1, indexb); numPop(indexa+1,
indexb+1); numPop(indexa-1,indexb+1);
                        return; // 리턴
                        }
                        pop1(indexa,indexb); // 나머지의 경우에 pop 함수로 숫자 판이 나올 때까지 버튼 이미지 변경

                        int n = popStop(); // 처리 안 된 칸 개수받아서

                        for(int k =0; k < n+3; k ++) { // 적당한 횟수 (그 횟수 + 3)만큼 처리
                                for(int i =1; i < realSize;i++) {
                                        for(int j =1; j < realSize;j++) { ////////////////////////////////// 팝하
다가 만 곳 발견시 그 곳 부터 다시 팝
                                                if(bt[i][j].getIcon() == minenormal && bt[i-1][j-1].getIcon() ==
nomine) pop1(i-1, j-1);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i-1][j].getIcon() ==
nomine) pop1(i-1, j);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i-1][j+1].getIcon()
== nomine) pop1(i-1, j+1);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i][j+1].getIcon() ==
nomine) pop1(i, j+1);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i+1][j+1].getIcon()
== nomine) pop1(i+1, j+1);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i+1][j].getIcon() ==
nomine) pop1(i+1, j);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i+1][j-1].getIcon()
== nomine) pop1(i+1, j-1);
                                                elseif(bt[i][j].getIcon() == minenormal && bt[i][j-1].getIcon() ==
nomine) pop1(i, j-1);
```

```
                                        }
                                    }
                                }
                            }
                            ////////////////// 근처에 지뢰가 있는 곳을 눌렀을 때
                            if(Gamerm.isMine(indexa, indexb) >0) { // 근처에 지뢰가 존재하면
                                numPop(indexa, indexb);
                            }
                        ////////////////////// 게임 끝나면
                        if(Gamerm.finish) {
                            Gamerm.time = sec; // 게임을 깬 시간 저장
                            GameWin w =new GameWin(Gamerm.time, this); // 시간초와 현재 진행한 게임 객체를 매개변수로 넘
겨줌
                        }
                        }
}
```

이 부분은 코드는 난잡하지만 내용은 뚜렷하다. 근처에 지뢰가 없는 부분을 누를 때인데, 만약 주변이 모두 숫자 판이면 누른 버튼만 아무것도 없는 이미지로 바꾸고 팔방은 각 숫자판으로 바꾸고 버튼 이벤트를 종료한다. 아닐 경우에 일단 8방과 자신의 이미지를 바꾸는 pop1 메소드를 실행한다. 그리고 popStop()으로 미처리된 버튼의 개수를 받아낸다. 사실 이 숫자는 큰 의미를 갖지는 않고, 적당히 그 수+3 정도의 횟수로 3중 반복 되어있는 코드를 실행한다. 그 부분은 popStop 함수에서처럼 전체 버튼 판에서 미처리된 부분을 찾아서 그곳을 다시 pop1 해주는 부분이다. 이 행위를 적당한 숫자로 반복해야 숫자판이 나올 때까지 진행되는데, 그 적당한 횟수를 얻기 위해 popStop()을 통해 얻어낸 수+3번 정도 반복을 시킨다. 사실 이 부분이 가장 처리하기 힘든 부분이었고, 처음에는 재귀함수를 떠올렸으나 StackOverFlow가 계속 발생해서 완벽하진 않지만 이 방법을 사용했다. 마지막으로 근처에 지뢰가 있는 부분을 클릭하면 numPop()이 실행되고, 아까 finish가 true가 되면 버튼 클릭으로 처리하는 부분이 맨 아래줄에 구현되어 있다.




```
class GameWin extends JFrame{ // 게임 승리
        JPanel p;
        JTextArea a, n1, name, info;
        Font f, f2;
        JButton  b2;
        GameWin(int sec, MineFrame mf){ // 시간과 게임 객체를 매개변수로 받음
            this.setTitle("오옷!");
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setSize(300, 200);

            p =new JPanel();
            a =new JTextArea("게임 승리!!");
            n1 =new JTextArea("이름 입력: ");
            name =new JTextArea(); // 이름을 입력받음
            info =new JTextArea("맵 사이즈:"+Gamerm.size+" 시간:"+"");
            f =new Font("San Serif", Font.BOLD, 40);
            f2 =new Font("San Serif", Font.PLAIN, 20);
```

```
                b2 =new JButton("게임 종료");

                p.setLayout(null);

                b2.addActionListener(newActionListener() { ////////////////////////// 종료 버튼 누르면, 최종 작업
                        publicvoid actionPerformed(ActionEvent e) {
                                Gamerm.name = name.getText(); // 입력받은 이름 저장

                                String s = Integer.toString(Gamerm.time)+"sec "+ Integer.toString(Gamerm.size-2)
+"size, "+ Gamerm.name; // 사이즈는 경계 제외(-2) , 'sec '로 시간과 구별

                                try {
                                        GameRun.readMine(); // 순위를 불러와서
                                } catch (IOException e1) {
                                        // TODO Auto-generated catch block, IOException을 해결하기 위한
try-catch
                                        e1.printStackTrace();
                                }
                                GameRun.putMine(s); // 새로운 순위를 넣고
                                try {
                                        GameRun.WriteMine(); // 다시 파일에 순위를 써넣는다.
                                        } catch (IOException e1) {
                                        // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }
                                Gamerm.clear(); // 게임 정보 초기화
                                dispose(); // 현재 창 닫고
                                mf.dispose(); // 매개변수로 넘겨받은 게임을 종료
                                GameStartMenu c =new GameStartMenu(); // 처음 화면으로 돌아감
                        }
                });

                a.setEditable(false);
                a.setFont(f);
                a.setBackground(null);
                a.setForeground(Color.BLUE);
                n1.setEditable(false);
                n1.setFont(f2);
                name.setFont(f2);

                a.setBounds(35,10,300,50);
                n1.setBounds(10,60,100,30);
                name.setBounds(120,60,130,30);
                b2.setBounds(85,100,100,40);

                p.add(a);
                p.add(n1);
                p.add(name);
                p.add(b2);
                this.add(p);
                this.setVisible(true);
        }
}
```

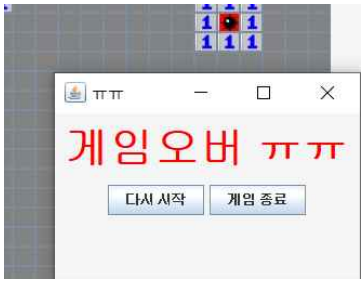
이 코드는 게임에 승리하면 나오는 프레임이다.

가장 중요한 부분이 이름을 입력 받은 뒤에, 문자열 s에 시간초, 사이즈, 이름을 문자열 하나로 만들어 저장하고 read->put->write static 메소드를 실행하는 부분이다. 여기서 게임 종료를 누르면 매개변수로 받은 이전 게임 창 객체도 종료하고 게임 승리 창도 종료하고 다시 게임 선택창으로 돌아간다.

```
class GameOver extends JFrame{ // 게임 오버시에
	JPanel p;
	JTextArea a;
	Font f;
	JButton b1, b2;
	GameOver(MineFrame mf){
		this.setTitle("ㅠㅠ");
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setSize(300, 200);

		p =new JPanel();
		a =new JTextArea("게임오버 ㅠㅠ");
		f =new Font("San Serif", Font.BOLD, 40);
		b1 =new JButton("다시 시작");
		b2 =new JButton("게임 종료");
		b1.addActionListener(newActionListener() { // 같은 사이즈와 지뢰 개수로 다시 시작
			publicvoid actionPerformed(ActionEvent e) {
				int n1 = Gamerm.size; int n2 = Gamerm.reNum; // 사이즈와 지뢰 개수 저장 후 매
개변수로 넣어줌

				Gamerm.clear(); // 게임 초기화 후
				mf.dispose(); // 게임 창 닫고
				dispose(); // 현재 창 종료 후
				MineFrame gf =new MineFrame(n1, n2); // 매개 변수를 갖는 지뢰 찾기 게임 생성
			}
		});
		b2.addActionListener(newActionListener() { // 게임 초기화면으로 돌아감
			publicvoid actionPerformed(ActionEvent e) {
				mf.dispose(); // 게임 창 닫고
				dispose(); // 창 닫고
				GameStartMenu c =new GameStartMenu(); // 처음 화면으로 돌아감
			}
		});

		a.setEditable(false);
		a.setFont(f);
		a.setBackground(null);
		a.setForeground(Color.RED);

		p.add(a);
		p.add(b1);
		p.add(b2);
		this.add(p);
		this.setVisible(true);
```

```
        }
}
```

이 프레임은 게임 오버시에 나오는 창이다. 이 창에는 버튼이 두 개 존재하는데, 다시 시작을 누르면 size와 reNum을 이용해 스마일을 눌렀을 때처럼 기존 정보로 새로운 게임 객체를 생성한다. 역시 여기서도 매개변수로 받은 이전 게임 객체를 dispose하고 현재 창도 dispose 해준다. 게임 종료를 누르면 게임 객체, 현재 창 모두 dispose 하고 게임 선택 메뉴를 띄운다.



6. Omok

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.IOException; // IOException을 위해
import java.util.Random; // 랜덤을 사용하기 위해
import javax.swing.Timer; // 타이머 기능을 위해
class Gamer{ // 게이머들의 정보를 저장하는 클래스
        staticString omokScore[]; // 오목 순위를 저장할 문자열 배열
        staticint index; // 오목 순위 문자열 배열의 인덱스
        staticString g1; // 게이머 1, 2 이름
        staticString g2;
        staticString winner;

        staticint size; // 오목판 사이즈
        staticboolean turn =false; // 흑 백 턴을 결정할 변수 (최초에 false, 백의 차례)
        staticint sooB =0; // 흑 백 수를 저장할 변수 처음에 0으로 초기화
        staticint sooW =0;
        staticboolean bpan[][]; // 검은돌 판의 배열
        staticboolean wpan[][]; // 흰돌 판의 배열
        staticint winSoo; // 이긴 수
        staticboolean finish =false; // 게임 끝 여부, 최초 false로 초기화

        staticvoid clear() { // 게임을 초기화(static 변수들 초기화)
                for(int i =0; i<size; i++) {
                        for(int j =0; j<size; j++) {
                                bpan[i][j] =false;
                                wpan[i][j] =false;
                        }
                }
                g1 =null; // 게이머 1, 2 이름
                g2 =null;
                winner =null;
                size =0; // 오목판 사이즈
                turn =false; // 흑 백 턴을 결정할 변수 (최초에 false, 백의 차례)
                sooB =0; // 흑 백 수를 저장할 변수 처음에 0으로 초기화
                sooW =0;
                winSoo =0; // 이긴 수
                finish =false; // 게임 끝 여부
        }
}
```

다음은 오목 게임 구현부이다. 지뢰 찾기와 크게 다른 것은 없지만 static 변수가 들어있는 Gamer 클래스 내에 bpan[][], wpan[][] 으로 각각 검은돌, 흰 돌이 눌리면 그 인덱스의 원소를 true로 바꿔준다.

```
publicclass Omok extends JFrame implementsActionListener {

	JPanel inputOmok; // 입력하는 패널
	JButton exitGame;
	JButton order; // 입력한 값으로 게임 생성
	JTextField f1,f2, f3, f4, f5, f6, info;
	Omok(){
		// 컴포넌트 설정
		inputOmok =new JPanel();
		exitGame =new JButton ("게임 종료");
		f1 =new JTextField("사용자 1 이름");
		f2 =new JTextField("사용자 2 이름");
		f3 =new JTextField(); // 사용자 1 이름 입력받음
		f4 =new JTextField(); // 사용자 2 이름 입력받음
		f5 =new JTextField("바둑판 크기(20~50)");
		f6 =new JTextField();
		info =new JTextField("    ** 백이 먼저 돌을 놓습니다 **");
		order =new JButton("순서 정하기");

		// 프레임 설정
		this.setTitle("오목 게임");
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setSize(400, 400);
		this.add(inputOmok);
		inputOmok.setLayout(null);

		// 컴포넌트 위치 설정
		f1.setEditable(false);
		f2.setEditable(false);
		f5.setEditable(false);
		info.setEditable(false);
		info.setBackground(Color.red);
		info.setForeground(Color.white);
		exitGame.setBounds(120, 20, 140, 30);
		f1.setBounds(10,60, 150, 30);
		f2.setBounds(10,140, 150, 30);
		f3.setBounds(10,100, 150, 30);
		f4.setBounds(10,180, 150, 30);
		f5.setBounds(200,60, 150, 30);
		f6.setBounds(200,100, 150, 30);
		info.setBounds(100, 240, 180, 30);
		order.setBounds(120, 280, 140, 30);

		// 컴포넌트 부착
		inputOmok.add(exitGame);
		inputOmok.add(f1);
		inputOmok.add(f2);
		inputOmok.add(f3);
		inputOmok.add(f4);
		inputOmok.add(f5);
		inputOmok.add(f6);
		inputOmok.add(info);
```

```
                inputOmok.add(order);

                exitGame.addActionListener(this); // 게임 종료
                order.addActionListener(this); // 순서 정하기
                this.setVisible(true);
        }
        @Override
        publicvoid actionPerformed(ActionEvent e) {
                if(e.getSource() == exitGame) {
                        GameStartMenu game =new GameStartMenu();
                        this.setVisible(false);
                }
                if(e.getSource() == order) {
                        Gamer.g1 = f3.getText(); // 입력받은 이름 게이머 정보에 저장
                        Gamer.g2 = f4.getText();
                        Gamer.size = Integer.parseInt(f6.getText()) +2; // 판의 가장자리를 안 쓰는 것을 고려해서 잡
음

                        Order o =new Order();
                        this.setVisible(false);
                }
        }
}
```

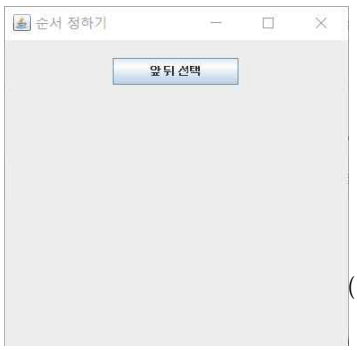
사용자 1, 2 이름과 바둑판 크기를 입력받는 부분으로 앞서 지뢰 찾기의 입력 프레임과 크게 다른 점이 없다. 각각 static 변수 g1, g2, size에 정보를 입력받는다. 참고로 여기서도 가장자리를 제외하기 위해 사이즈를 2 크게 설정한다.


```
class Order extends JFrame implementsActionListener{ // 선공 후공 순서 정하는 클래스
        JPanel p1;
        JButton b1, b2, b3;
        JTextField f1,f2;
        ImageIcon foreim, backim;
        JLabel p1lb, p2lb, firstlb;

        Order() {

        this.setTitle("순서 정하기");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(400, 400);

        b1 =new JButton("앞 뒤 선택");
        p1lb =new JLabel();
        foreim =new ImageIcon("fore.jpg");
        p2lb =new JLabel();
```

```
backim =new ImageIcon("back.jpg");
firstlb =new JLabel(); // 선공할 동전
p1 =new JPanel();
p1.setLayout(null);
f1 =new JTextField(Gamer.g1 +"player의 동전");
f2 =new JTextField(Gamer.g2 +"player의 동전" );
b2 =new JButton("선공은?");
b3 =new JButton("게임 시작");
Random rand =newRandom(); // 랜덤 객체 생성
boolean r = rand.nextBoolean(); // true or false를 랜덤으로 생성
boolean first = rand.nextBoolean(); // true면 앞면, false면
b1.setBounds(120, 20, 140, 30);
f1.setBounds(10,60, 150, 30);
f2.setBounds(210,60, 150, 30);
b2.setBounds(10,240,140,30);
p1lb.setBounds(30,60,200,200);
p2lb.setBounds(240,60,200,200);
firstlb.setBounds(200, 200, 180, 200);
b3.setBounds(10,300,140,30);

if(r) { // 랜덤으로 앞면 뒷면을 해당 플레이어 아래에 보여줌
        p1lb.setIcon(foreim); // 1번이 앞면
        p2lb.setIcon(backim); // 2번이 뒷면
}
else {
        p1lb.setIcon(backim);
        p2lb.setIcon(foreim);
}
if(first) { // 앞면이면
        firstlb.setIcon(foreim); // 앞면이 선공
}
else { // 뒷면이면
        firstlb.setIcon(backim);
}

b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
p1.add(f1);
p1.add(f2);
p1.add(p1lb);
p1.add(p2lb);
p1.add(b1);
p1.add(b2);
p1.add(firstlb);
p1.add(b3);

f1.setVisible(false); // 버튼이 눌린 후 보여지기 위해서 비가시화로 설정한 컴포넌트들
f2.setVisible(false);
p1lb.setVisible(false);
p2lb.setVisible(false);
b2.setVisible(false);
firstlb.setVisible(false);
b3.setVisible(false);
```

```
        this.add(p1);
        this.setVisible(true);
        }
        publicvoid actionPerformed(ActionEvent e) {
                if(e.getSource() == b1) { // 앞, 뒤를 선택
                        f1.setVisible(true); f2.setVisible(true);
                        p1lb.setVisible(true); p2lb.setVisible(true);
                        b2.setVisible(true);
                }
                if(e.getSource() == b2) { // 선공을 정함
                        firstlb.setVisible(true); b3.setVisible(true);
                        if(firstlb.getIcon() == p1lb.getIcon()) { // firstlb와 같은 이미지를 가진 플레이어가 선공
                                System.out.println("p1이 선공");
                                Gamer.turn =false;
                        }
                        elseif(firstlb.getIcon() == p2lb.getIcon()) {
                                System.out.println("p2이 선공");
                                String tmp = Gamer.g1; // 선공의 이름을 g1로, 후공의 이름을 g2로 바꿈
                                Gamer.g1 = Gamer.g2;
                                Gamer.g2 = tmp;
                                Gamer.turn =false;
                        }
                }
                if(e.getSource() == b3) { // 게임 시작
                        this.dispose();
                        OmokFrame of =new OmokFrame(Gamer.size); // 오목 게임 실행
                }
        }
}
```

선/후공을 정하는 프레임이다. 랜덤 함수를 사용해 player1과 player2의 이름 아래에 동전 앞/뒤 이미지를 띄운다.



(앞 뒤 선택을 누르면->)

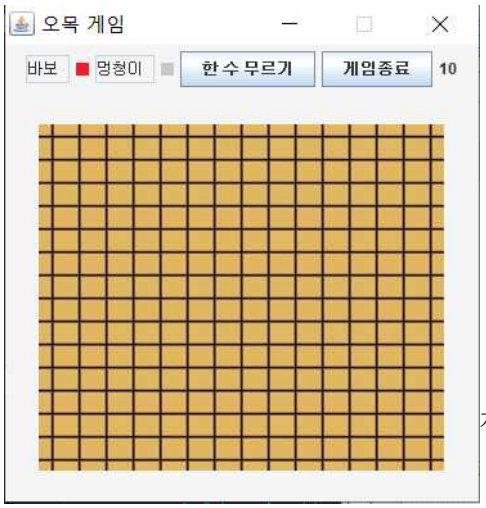
그리고 또 랜덤함수를 이용해 firstlb라는 라벨에 선공을 취할 버튼의 이미지를 정하는데, b2버튼(선공은?)을 누르면 랜덤으로 동전 앞/뒤가 이미지로 나온다. 그리고 static 변수 g1에는 무조건 선공이, g2에는 무조건 후공이 들어가야 하므로 만약 player2가 후공이면 g1과 g2를 서로 바꿔준다. 마지막으로 b3(게임 시작)을 누르면 게임 창이 뜨고 현재 창은 dispose된다. 그리고 제일 중요한 게임 턴은 선공(백돌)일 때 static 변수 turn이 false, 후공일 때 false이다.

```
class OmokFrame extends JFrame implementsActionListener{ // 오목판이 있는 게임 프레임
        JPanel omokP =new JPanel();
        JButton [][] bt;
        JPanel info;
        JTextField name1, name2; // 순서대로 선공, 후공
        ImageIcon w,b;
        ImageIcon p,t,nt;
        JLabel t1, t2;
        JButton mool, exit;
        JButton backSoo; // 무를 수
        JLabel time; // 시간이 표시될 라벨
        int sec; // 시간
        int sizeforIndex; // 인덱스를 위한 사이즈

        int mCountb =1; // 무를 수 있는 횟수 한 번
        int mCountw =1;
        int indexa =0; // 클릭된 버튼의 인덱스 저장
        int indexb =0;
        Timer timer;
        JTextField end; // 게임의 끝을 표시할 문구
        Font f;

        OmokFrame(int a){ // 사이즈를 매개변수로 받는 생성자

                sizeforIndex = a-1; // 인덱스를 위한 사이즈, 사이즈-1
                sec =20; // 20초의 시간을 줌
                timer =new Timer(1000, newActionListener () { // 타이머 설정
                        publicvoid actionPerformed(ActionEvent e) {
                                if(sec<0) { // 타이머가 0이 되는 순간
                                        timer.stop(); // 타이머 리셋
                                        sec =20;
                                        timer.restart();
                                        if(Gamer.turn) { // 검은색 차례였다면
                                                Gamer.turn =false; // 흰색으로 턴이 넘어감
                                                t1.setIcon(t); // 흰 돌을 빨간 불로
                                                t2.setIcon(nt);
                                        }
                                        else { // 흰색 차례였다면
                                                Gamer.turn =true; // 검은색으로 턴이 넘어감
                                                t1.setIcon(nt);
                                                t2.setIcon(t); // 검은 돌을 빨간 불로
                                        }
                                }
                                time.setText(Integer.toString(sec));
                                sec -=1; // 1초마다 1씩 줄어듬
                        }
                }); // 타이머 설정
                timer.start(); // 타이머 실행

                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 종료 버튼
                this.setTitle("오목 게임");
                this.setLayout(new BorderLayout()); // BorderLayout으로 프레임 설정
                this.setSize(a*22,a*23); // 매개변수로 받은 a 값에 따라 프레임 크기를 조절
                this.setResizable(false); // 프레임 사이즈 변경 불가
```

```
Gamer.bpan =newboolean[a][a]; // 버튼의 상태가 저장될 배열
Gamer.wpan =newboolean[a][a];

omokP =new JPanel(); // 오목 패널
omokP.setLayout(new GridLayout(a,a)); // a*a 개의 버튼을 갖는 패널
bt =new JButton[a][a]; // 버튼의 a행 a열의 이차원 배열 생성
w =new ImageIcon("wdol.jpg");
b =new ImageIcon("bdol.jpg");
p =new ImageIcon("pandegi.jpg");

info =new JPanel(); // 시간 등 정보가 표시되는 패널
name1 =new JTextField(Gamer.g1+"  "); name1.setEditable(false); // 선공 이름
name2 =new JTextField(Gamer.g2+"  "); name2.setEditable(false); // 후공 이름
t =new ImageIcon("turn.jpg"); // 턴이 돌아오면 빨간불
nt =new ImageIcon("noturn.jpg"); // 턴이 아니면 회색불
t1 =new JLabel(); t1.setIcon(t); // 선공은 빨간불
t2 =new JLabel(); t2.setIcon(nt); // 후공은 희색불
mool =new JButton("한 수 무르기"); // 물리기 버튼
exit=new JButton("게임종료"); // 게임 종료 버튼
time =new JLabel("시간");
end =new JTextField("끝");
f =new Font("San Serif", Font.BOLD, 13);

info.add(name1);
info.add(t1); // 선공 상태
info.add(name2);
info.add(t2); // 후공 상태
info.add(mool);
info.add(exit);
info.add(time);
info.add(end); // 게임 끝나면 안내 문구 표시

end.setFont(f);
end.setBackground(Color.red);
end.setBorder(null);
end.setForeground(Color.white);
end.setVisible(false);

for(int i =0; i <a; i++) { // 행만큼 반복
        for(int j =0; j <a; j++) { // 열만큼 반복
                bt[i][j] =new JButton(); // 버튼 생성, 버튼 이미지 부착
                omokP.add(bt[i][j]); // 패널에 부착
                if(i>0&&i<a-1&& j>0&& j<a-1) {
                bt[i][j].setIcon(p); // 오목 판 이미지를 붙임
                bt[i][j].setBorderPainted(false); // 버튼 경계선 안 보이도록 설정
                bt[i][j].addActionListener(this); // 모든 버튼에 이벤트 설정
                }
                else {
                        bt[i][j].setVisible(false); // 가장자리 버튼들 안 보이도록 설정
                }
        }
}
```

```
                mool.addActionListener(this); // 이벤트 추가
                exit.addActionListener(this);

                add(omokP, BorderLayout.CENTER); // 오목판 패널을 프레임 중앙에 부착
                add(info, BorderLayout.NORTH); // 정보가 표시되는 패널을 프레임 위쪽에 부착
                setVisible(true); // 가시화
        }
```

게임 시작을 누르면 뜨는 오목판 프레임이다. 여러 변수들이 선언되어있는데 중요한 변수로는 mCountb, mCountw로, 무를 수 있는 횟수 1번을 의미한다. 생성자로 버튼의 개수 a를 받는데, a는 현재 입력받은 사이즈보다 2가 크다. 그래서 sizeforindex라는 변수에 a-1값을 넣어준다. 지뢰 찾기의 realSize와 같은 역할을 하는 변수이다. 시간은 20초를 설정하고, 1초에 1씩 줄어들게 된다. 플레이어의 이름을 선공 후공 순서대로 바둑판 위에 표시하고 그 옆에 현재 턴인 사람의 경우에 빨간 불이 들어오도록 이미지를 넣었다.(추가사항 뒤에 정리해놓음) 또 한 수 무르기 버튼과 게임 종료 버튼이 존재한다. 타이머 구현부 내에서 변수 sec가 0이 되면 타이머가 멈췄다가 다시 20으로 바뀌고 턴이 바뀌게 된다. 또 특이사항으로는 버튼 생성당시에 조건문을 통해 가장자리를 제외한 버튼들에만 이미지를 붙이고 버튼 벤트를 설정한 것이다. 가장자리 버튼은 setVisible(false)로 처리했다.



게임 창, 현재 player1(선공, 백돌) 차례.

```
publicvoid actionPerformed(ActionEvent e) {

                timer.stop(); // 버튼이 눌리면 타이머 초기화
                sec =20;
                timer.restart();

                if(e.getSource() ==exit) { // 종료버튼 누르면
                        Gamer.clear(); // 게임 정보 초기화
                        this.dispose(); // 게임창 종료
                        GameStartMenu g =new GameStartMenu();
                }
                if(e.getSource() == mool) { // 한수 무르기 누르면
                        if(Gamer.turn) { // 검은 돌 차례였으면
                                if(mCountw>0&& backSoo !=null) { // 무르기 사용 횟수가 존재할 때만, backSoo가 null이면 방금 전에 무르기를 했다는 것(연속 무르기 방지)
                                        mCountw--; // 흰 돌이 무르기 사용했으므로 카운트 감소
                                        backSoo.setIcon(p); // 최근에 눌린 버튼을 다시 오목판 그림으로 바꾸고
                                        backSoo =null; // backSoo를 null로 만들어 방금 전에 무르기를 했다는 사실을 표시

                                        Gamer.turn =false; // 다시 흰 돌 차례로 바꿔줌
                                        Gamer.sooW--; // 흰 돌의 수를  하나 줄임
                                        t1.setIcon(t); // 흰 돌을 빨간 불로
                                        t2.setIcon(nt);
                                        Gamer.wpan[indexa][indexb] =false; // 흰돌에 눌린 것으로 표시된 것을 다시 초기화
```

```
                                        }
                                }
                                else { // 흰 돌 차례였으면
                                        if(mCountb>0&& backSoo !=null) { // 무르기 사용 횟수가 존재할 때만, 연속 무르기 방지
                                        mCountb--; // 검은돌이 무르기 사용했으므로 카운트 감소
                                        backSoo.setIcon(p); // 최근에 눌린 버튼을 다시 오목판 그림으로 바꾸고
                                        backSoo =null; // backSoo를 null로 만들어 방금 전에 무르기를 했다는 사실을 표시
                                        Gamer.turn =true; // 다시 검은 돌 차례로 바꿔줌
                                        Gamer.sooB--; // 검은 돌의 수를 하나 줄임
                                        t1.setIcon(nt);
                                        t2.setIcon(t); // 검은 돌을 빨간 불로
                                        Gamer.bpan[indexa][indexb] =false; // 검은돌에 눌린 것으로 표시된 것을 다시 초기화
                                        }
                                }
                        }
```

버튼을 눌렀을 경우 이벤트 처리이다. 처음 버튼을 누를 때마다 timer가 20으로 초기화된다. exit을 누르면 static 변수를 초기화하고 게임창을 닫은 뒤 게임 선택창으로 돌아가고, 무르기를 누르면 검은돌/흰돌 차례에 따라 다른 작업이 실행된다. 하나의 예로 현재 검은 돌 차례일 때 무르기를 누르면 흰 돌을 무르는 것이므로 mCountw를 하나 줄인다. backSoo라는 가장 최근에 놓았던 돌의 정보가 저장된 버튼을 다시 오목판 그림으로 바꾸고 backSoo는 초기화한다. 그리고 무르기 작업은 mCountb/w가 >0일 경우와 backSoo가 null이 아닐 때에만 실행되므로 한 게임에 한 번만 가능하며 검은돌과 흰돌이 연속 무르기가 불가능하다.(예외처리) 그리고 흰 돌의 눌린 버튼을 저장하는 wpan의 최근 눌린 인덱스가 저장된 indexa,indexb를 다시 false로 바꾼다. 턴을 변경한다.

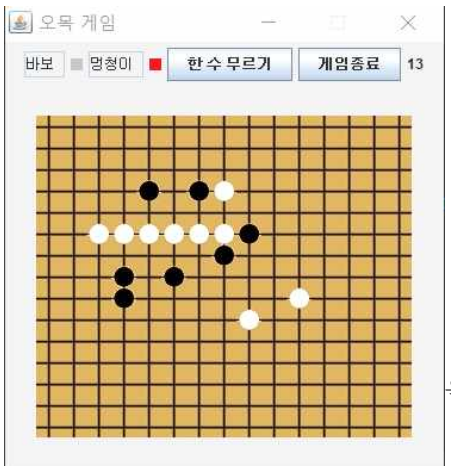
```
        for(int i =0;i<Gamer.size;i++) { // 돌을 놓는 경우
                        for(int j =0;j<Gamer.size;j++) {

                        if(!Gamer.bpan[i][j] &&!Gamer.wpan[i][j]) { // 이미 눌린 버튼이면 넘어감
                                if(e.getSource() == bt[i][j]) { // 특정 버튼이 눌렸는데
                                        backSoo = bt[i][j]; // 가장 최근에 눌린 버튼 저장
                                        indexa = i; indexb = j; // 인덱스 저장
                                        if(!Gamer.turn) { // 흰 돌이 놓을 차례였으면
                                                bt[i][j].setIcon(w); // 흰 돌이 놓여진 그림으로 바꿈
                                                Gamer.wpan[i][j] =true; // 흰 돌이 놓여진 위치 저장
                                                Gamer.turn =true; // 턴을 검은 돌에게 넘겨줌
                                                Gamer.sooW++; // 흰 돌 수 증가
                                                t1.setIcon(nt);
                                                t2.setIcon(t); // 검은 돌을 빨간 불로
                                                ////////////////////////////////////////////////이기는 경우
                                                //// 상하 좌우로 오목인 경우
                                                int nu = i; // 연결된 제일 위 인덱스
                                                while(Gamer.wpan[nu][j]) {
                                                        nu--;
                                                } nu++;
                                                int nd = i; // 연결된 제일 아래 인덱스
                                                while(Gamer.wpan[nd][j]) {
                                                        nd++;
                                                } nd--;
                                                int nr = j; // 연결된 제일 오른쪽 인덱스
                                                while(Gamer.wpan[i][nr]) {
                                                        nr++;
                                                } nr--;
                                                int nl = j; // 연결된 제일 왼쪽 인덱스
```

```
                while(Gamer.wpan[i][nl]) {
                        nl--;
                }nl++;
                if((nr-nl==4) && (nr-nl !=5)) { // 좌우로 오목인 경우, 육목 제외
                        time.setVisible(false); // 시간은 안 보일록
                        end.setVisible(true); // 끝났음을 표시
                        Gamer.finish =true;
                        Gamer.winner = Gamer.g1; // 선공(백돌) 승리
                        Gamer.winSoo = Gamer.sooW; return;
                }
                elseif((nd-nu ==4)&&(nd-nu !=5)) { // 상하로 오목인 경우, 육목
제외
                        time.setVisible(false); // 시간은 안 보일록
                        end.setVisible(true); // 끝났음을 표시
                        Gamer.finish =true;
                        Gamer.winner = Gamer.g1; // 선공(백돌) 승리
                        Gamer.winSoo = Gamer.sooW;  return;
                }
                //// 대각선으로 오목인 경우
                // 오른쪽 아래
                int rdx = j; int rdy = i; int lux = j; int luy = i;
                while(Gamer.wpan[rdy][rdx]) {
                        rdx++; rdy++;
                } rdx--; rdy--;

                while(Gamer.wpan[luy][lux]) { // 왼쪽 위
                        lux--; luy--;
                }lux++; luy++;
                if((rdx-lux ==4)&&(rdx-lux !=5)) { // 어차피 x와 y값이 같이 커지
므로 x값 하나만 비교해도 됨
                        time.setVisible(false); // 시간은 안 보일록
                        end.setVisible(true); // 끝났음을 표시
                        Gamer.finish =true;
                        Gamer.winner = Gamer.g1; // 선공(백돌) 승리
                        Gamer.winSoo = Gamer.sooW;  return;
                }
                // 오른쪽 위
                int rux = j; int ruy = i; int ldx = j; int ldy = i;
                while(Gamer.wpan[ruy][rux]) {
                        rux++; ruy--;
                }rux--; ruy++;
                while(Gamer.wpan[ldy][ldx]) { // 왼쪽 아래
                        ldx--; ldy++;
                }ldx++; ldy--;
                if((rux-ldx ==4)&&(rux-ldx !=5)) { // 어차피 x와 y값이 같이 커지
므로 x값 하나만 비교해도 됨
                        time.setVisible(false); // 시간은 안 보일록
                        end.setVisible(true); // 끝났음을 표시
                        Gamer.finish =true;
                        Gamer.winner = Gamer.g1; // 선공(백돌) 승리
                        Gamer.winSoo = Gamer.sooW; return;
                        }
                }
        else { // 검은돌을 놓을 차례였으면
```

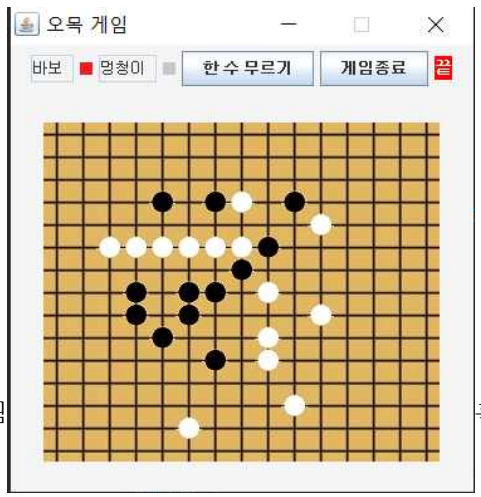
```
bt[i][j].setIcon(b); // 검은 돌이 놓여진 그림으로 바꿈
Gamer.bpan[i][j] =true; // 검은 돌이 놓여진 위치 저장
Gamer.turn =false; // 턴을 흰 돌에게 넘겨줌
Gamer.sooB++; // 검은 돌 수 증가
t1.setIcon(t); // 흰 돌을 빨간 불로
t2.setIcon(nt);
//////////////////////////////////////////////////이기는 경우

//// 상하 좌우로 오목인 경우
int nu = i; // 연결된 제일 위 인덱스
while(Gamer.bpan[nu][j]) {
        nu--;
}nu++;
int nd = i; // 연결된 제일 아래 인덱스
while(Gamer.bpan[nd][j]) {
        nd++;
}nd--;
int nr = j; // 연결된 제일 오른쪽 인덱스
while(Gamer.bpan[i][nr]) {
        nr++;
}nr--;
int nl = j; // 연결된 제일 왼쪽 인덱스
while(Gamer.bpan[i][nl]) {
        nl--;
}nl++;
if((nr-nl==4) && (nr-nl !=5)) { // 좌우로 오목인 경우, 육목 제외
        time.setVisible(false); // 시간은 안 보인록
        end.setVisible(true); // 끝났음을 표시
        Gamer.finish =true;
        Gamer.winner = Gamer.g2; // 후공(흑돌) 승리
        Gamer.winSoo = Gamer.sooB;  return;
}
elseif((nd-nu ==4)&&(nd-nu !=5)) { // 상하로 오목인 경우, 육목 제외

        time.setVisible(false); // 시간은 안 보인록
        end.setVisible(true); // 끝났음을 표시
        Gamer.finish =true;
        Gamer.winner = Gamer.g2; // 후공(흑돌) 승리
        Gamer.winSoo = Gamer.sooB;  return;
}
//// 대각선으로 오목인 경우
// 오른쪽 아래
int rdx = j; int rdy = i; int lux = j; int luy = i;
while(Gamer.bpan[rdy][rdx]) {
        rdx++; rdy++;
} rdx--; rdy--;
while(Gamer.bpan[luy][lux]) { // 왼쪽 위
        lux--; luy--;
}lux++; luy++;
if((rdx-lux ==4)&&(rdx-lux !=5)) { // 어차피 x와 y값이 같이 커지므로 x값 하나만 비교해도 됨

        time.setVisible(false); // 시간은 안 보인록
        end.setVisible(true); // 끝났음을 표시
        Gamer.finish =true;
```

```
                    Gamer.winner = Gamer.g2; // 후공(흑돌) 승리
                    Gamer.winSoo = Gamer.sooB;
                     return;
                }
                // 오른쪽 위
                int rux = j; int ruy = i; int ldx = j; int ldy = i;
                while(Gamer.bpan[ruy][rux]) {
                    rux++; ruy--;
                }rux--; ruy++;

                while(Gamer.bpan[ldy][ldx]) { // 왼쪽 아래
                    ldx--; ldy++;
                }ldx++; ldy--;
                if((rux-ldx ==4)&&(rux-ldx !=5)) { // 어차피 x와 y값이 같이 커지
므로 x값 하나만 비교해도 됨
                    time.setVisible(false); // 시간은 안 보일록
                    end.setVisible(true); // 끝났음을 표시
                    Gamer.finish =true;
                    Gamer.winner = Gamer.g2; // 후공(흑돌) 승리
                    Gamer.winSoo = Gamer.sooB;
                     return;
                    }
                }
```

이 부분은 가장 중요한 돌을 놓고 이기는 경우를 표현한 코드인데, 기본적으로 돌을 놓으면(버튼을 클릭하면) turn에 따라 백돌 혹은 흑돌로 버튼 이미지를 바꾼다. 예외처리로 이미 눌린 버튼이면 버튼 이벤트 자체를 넘어가버리고, backSoo와 indexa, indexb에 눌린 버튼의 정보를 저장한다. 어차피 검은돌이나 흰돌이나 코드는 b,w만 다르고 같기 때문에 흰돌을 예로 들어 설명한다. static 변수 wpan[][]을 true로 바꿔서 해당 위치 버튼이 눌렸음을 표시하고, static 변수 turn을 true로 바꿔주고(백돌이 선공, false이므로) static 변수 sooW++를 통해 수를 증가시킨다. 그리고 검은 돌을 빨간 불 들어오도록 해준다. 그 뒤에 나오는 코드가 바로 승리 조건(육목 제외 오목인 경우)이다. 코드를 보면 int nu=i;와 같은 변수 선언과 while()문이 잔뜩 나오는데, 이 작업은 방금 누른 버튼 위 위치부터 오른쪽 끝, 왼쪽 끝, 위쪽 끝, 아래쪽 끝 등을 주욱 따라가는 코드이다. wpan[][]이 true인 경우 wpan[nu][j]에서 nu--;를 해주는 식으로 제일 위 인덱스를 구하는 방식이다. 마지막에 nu를 ++해주는 이유는 마지막 실행된 --를 캔슬해주기 위함이다. 이 방식으로 오른쪽 끝 인덱스에서 왼쪽 끝 인덱스를 뺐을 때 4이고(1,2,3,4,5이면 빼면 4) 5가 아닐 때 승리하고, 똑같이 아래와 위의 차이, 오른쪽 아래 끝과 왼쪽 위 끝의 차이, 오른쪽 위 끝과 왼쪽 아래 끝의 차이가 5가 아니고 4일 때(육목 제외 오목일 때) 승리하도록 설정했다. 이것을 때는 시간 표시하는 곳을 안 보이도록 하고 end라는 텍스트 필드를 보이도록 해서 끝났음을 알린다. 백돌일 경우 winner static 변수에 g1 static 변수를, winSoo static 변수에 sooW static 변수를 넣고 finsh를 true로 만든 후 버튼 이벤트가 끝난다. 이후 아무 버튼이나 한 번 더 클릭하면 finish에 의해 다음 코드가 실행된다.



육목은 승리가 안 됨



흑돌 승리시 끝

```
        if(Gamer.finish) { ////////////////////////////////////////////// 게임이 누군가의 승리로 끝나면

        String s = Integer.toString(Gamer.winSoo)+" "+Gamer.winner;
```

```
                                                try {
                                                        GameRun.readOmok(); // 순위를 불러와서
                                                } catch (IOException e1) {
                                                        // TODO Auto-generated catch block, IOException을
해결하기 위한 try-catch
                                                        e1.printStackTrace();
                                                }
                                                GameRun.putOmok(s); // 새로운 순위를 넣고
                                                try {
                                                GameRun.WriteOmok(); // 다시 파일에 순위를 써넣는다.
                                                } catch (IOException e1) {
                                                // TODO Auto-generated catch block
                                                        e1.printStackTrace();
                                                }
                                                Win lol =new Win();

                                                dispose(); // 창 닫기
                                                }
                                        }
                                }
                        }
                }
}
```

게임이 누군가의 승리로 끝났을 때 버튼을 아무데나 클릭하면 이 finish 부분이 실행되는데, 앞서 지뢰 찾기처럼 문자열 s에 띄어쓰기를 기준으로 이긴 수와 이름을 저장하고 읽기, 넣기, 쓰기를 실행한다. 그리고 Win 객체를 실행하고 창을 닫는다.

```
class Win extends JFrame{
        JPanel p;
        JTextField winner;
        Font f;
        JButton exit;
        JLabel winlb;
        ImageIcon winim;
        Win(){
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 종료 버튼
                this.setTitle("오목 게임 승자");
                this.setSize(400,400); // 매개변수로 받은 a 값에 따라 프레임 크기를 조절
                this.setResizable(false); // 프레임 사이즈 변경 불가

                p =new JPanel();
                p.setLayout(null);

                winner =new JTextField();
                f =new Font("San Serif", Font.BOLD, 30);
                exit=new JButton("게임 종료"); // 게임 종료 버튼
                winlb =new JLabel();
                winim =new ImageIcon("win.jpg"); // 승리 이미지
                winlb.setIcon(winim);

                if(Gamer.winSoo == Gamer.sooW) { // 백돌이 이겼으면
                        winner.setText("백돌 "+Gamer.winner+Gamer.winSoo+" 수, 승리!!");
                }
                else {
                        winner.setText("흑돌 "+Gamer.winner+Gamer.winSoo+" 수, 승리!!");
```

```
                }
                winner.setHorizontalAlignment(JTextField.CENTER); // 텍스트 가운데정렬
                winner.setFont(f);
                winner.setBackground(Color.RED);
                winner.setForeground(Color.green);

                winner.setBounds(10,10,400,60);
                winlb.setBounds(80, 50, 300, 300);
                exit.setBounds(150, 320, 90,30);

                exit.addActionListener(newActionListener() {
                        publicvoid actionPerformed(ActionEvent e) {
                                Gamer.clear(); // 게임 정보 초기화
                                dispose(); // 창 닫기
                                GameStartMenu gm =new GameStartMenu();
                        }
                });
                p.add(winner);
                p.add(winlb);
                p.add(exit);
                this.add(p);
                this.setVisible(true);
        }
}
```

finish 실행 후에 나오는 Win 클래스이다. 텍스트 필드로 winSoo가 sooW랑 같다면 백돌, 아니면 흑돌이 승리했고, 이름과 이긴 수를 표시해준다. 그리고 그 아래에 팡파레 이미지를 추가했다. 이 부분은 단순히 텍스트, 이미지 출력에 게임 종료 버튼 하나밖에 없으므로 버튼 actionListener도 익명 클래스로 선언했다.

승리 화면

추가한 사항 :

1. 마우스 리스너(마우스 어댑터)를 사용해 게임 버튼 위에 마우스를 올리면 게임 그림이 뜬다.
2. 지뢰찾기나 오목에서 게임 승리 시에 눈에 잘 보이는 색으로 게임이 끝났음을 알리는 효과를 넣음
3. 오목 게임에서 현재 턴을 가지고 있는 사람의 옆에 빨간 불이 들어오도록 이미지 넣음
4. 오목 승리 화면에 팡파레 이미지 넣음