프로그래밍설계방법론
설계 프로젝트

‘게임 : 숫자패드 순서 맞추기’

학과 소프트웨어학부
학번 2018044993
이름 임소윤

1. 게임설명

1) 게임방법

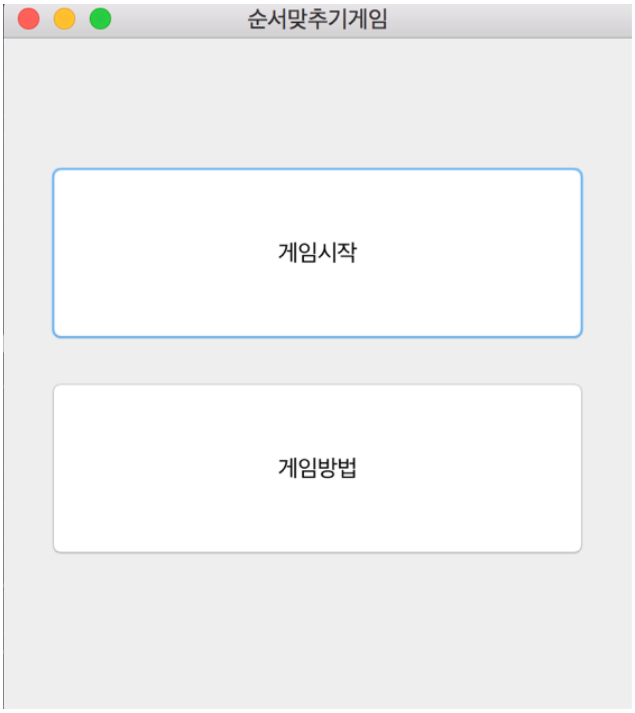
1	2	3
4	5	6
7	8	

① Easy Mode: 3*3 패드에 랜덤으로 배치된 1부터 8까지의 숫자를 위와 같은 순서대로 배열한다.

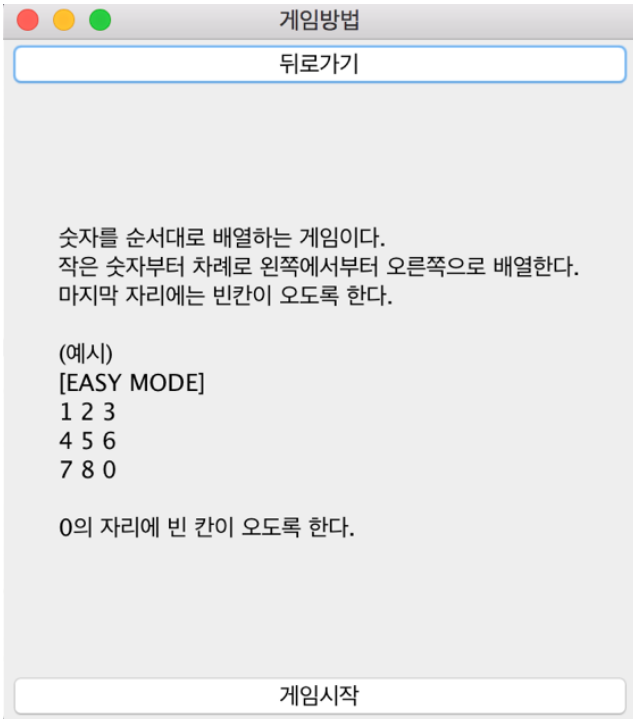
② Normal Mode: 4*4패드에 랜덤으로 배치된 1부터 15까지의 숫자를 easy mode와 같이 순서대로 배열한다.

③ Hard Mode: 5*5패드에 랜덤으로 배치된 1부터 24까지의 숫자를 easy mode, normal mode와 같이 순서대로 배열한다.

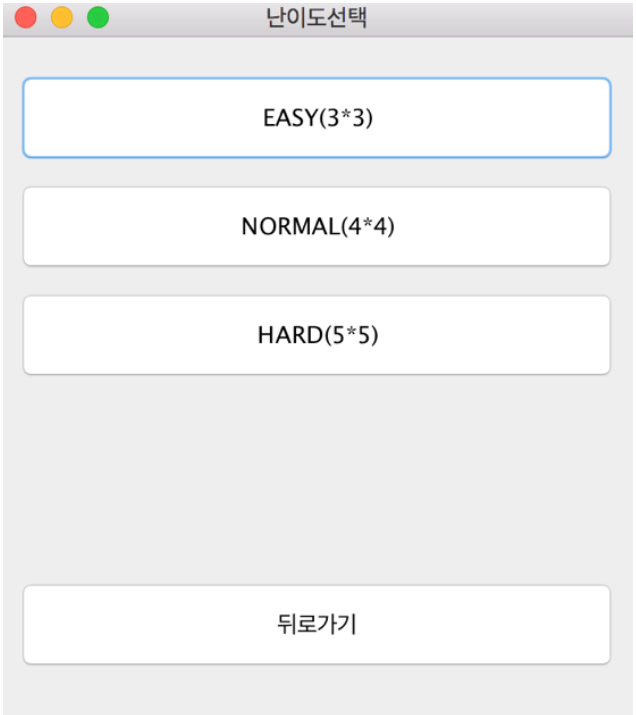
2) 게임화면



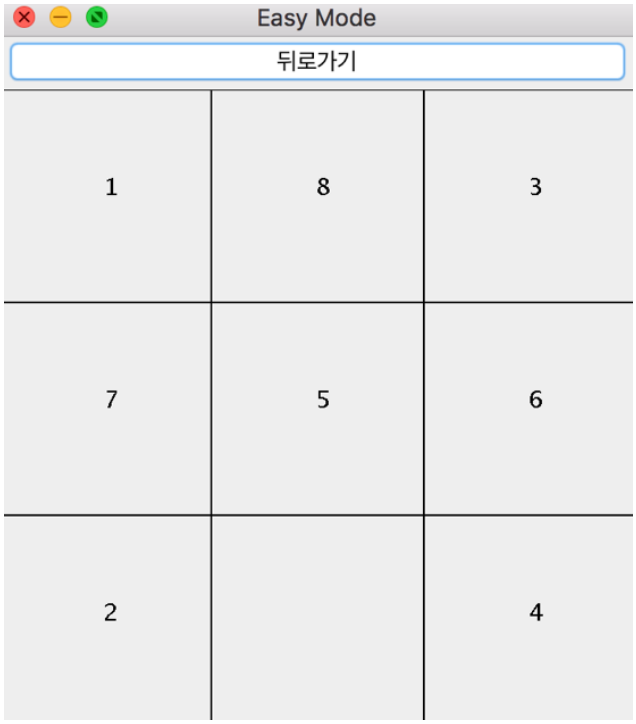
① 게임실행 시 첫 화면



② '게임방법' 클릭 시 화면 – '게임방법'화면



③ '게임시작' 클릭 시 화면 – '난이도선택'화면



④ 'EASY(3*3)' 클릭 시 화면 – 'Easy Mode'화면

⑤ 'NORMAL(4*4)' 클릭 시 화면 – 'Normal Mode'화면

⑥ 'HARD(5*5)' 클릭 시 화면 – 'Hard Mode'화면

결과: 모든 기능이 GUI를 통해 실행이 가능함을 볼 수 있다.

2. 기능 설명

1) 코드 설명

① 게임 화면(Easy/Normal/Hard)

```
public class Easy extends JFrame {
    private JButton back = new JButton( text: "뒤로가기");
    private Easymode easymode = new Easymode();

    public Easy(){
        setTitle("Easy Mode");
        setSize( width: 351,  height: 400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add( name: "North",back);
        back.addActionListener(new ActionListener() {
```

뒤로가기 버튼인 back과 게임화면인 easymode를 선언해준다.

프레임이 열렸을 때 프레임 제목이 'Easy Mode'로 나오도록 하고, 프레임의 크기는 가로 351, 세로 400으로 정한다. 프레임은 화면 중앙에 뜨도록 하고, 종료 버튼을 눌렀을 때 화면에서만 사라지는 것이 아닌 프로세스 상에서도 종료되도록 한다. Back버튼을 프레임 화면 맨 위에 추가한다.

```
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ea) {
                try{
                    dispose();
                    backFrame();
                }catch(Exception e){}
            }
        });
```

Back버튼을 클릭했을 때 현재 떠 있는 프레임 창을 종료하고 backFrame이 뜨게 한다.

```
        easymode.setLayout(new BoxLayout(easymode,BoxLayout.Y_AXIS));
        add( name: "Center",easymode);
        setVisible(true);
    }
```

(easymode 패널을 아래에 따로 선언해주었다.)

easymode를 프레임화면 중앙에 추가하고, easymode프레임이 화면에 보이게 한다.

```
    void backFrame(){
        Select select = new Select();
        select.setVisible(true);
    }
```

위에서 설명한 것처럼 back버튼을 눌렀을 때 켤 backFrame을 선언해준다.

난이도 선택 프레임인 select프레임을 열게 하고, select프레임이 화면에 보이도록 한다.

```
    public static void main(String[] args) {
        Easy easy = new Easy();
    }
}
```

Easy클래스를 실행한다.

```
class Easymode extends JPanel implements MouseListener {

    public void mouseReleased(MouseEvent e) { }
    public void mouseExited(MouseEvent e) { }
    public void mouseEntered(MouseEvent e) { }
    public void mouseClicked(MouseEvent e) { }
```

위에서 말한 Easymode를 선언해준다.

-mouseReleased : 마우스가 눌렸다가 놓아졌을 때

-mouseExited : 마우스가 뷰 영역을 나갔을 때

-mouseEnterd : 마우스가 뷰 영역으로 들어왔을 때

-mouseClicked : 마우스 버튼이 클릭 된 경우

```
public int[][] data = {{1,2,3},{4,5,6},{7,8,0}};
public Random rand;
public Easymode() {
    addMouseListener( l: this);
    rand = new Random();
    for (int i = 0; i < 100; i++) {
        init();
    }
}
```

정답 숫자 배열을 정의해주고, 게임 패드에서 숫자는 랜덤으로 배열되도록 한다.

```
public void mousePressed(MouseEvent e) {
    int size = 117;
    int x = e.getX() / size;
    int y = e.getY() / size;
    move(x, y);
    checkanswer();
}
```

마우스로 클릭하는 부분의 크기를 117로 지정한다. x와 y를 사이즈로 나눠주면 누른 곳의 숫자 패드가 (x, y)로 이동한다.
숫자 패드를 이동할 때마다 정답인지 확인한다.

```
public void paint(Graphics g) {
    paintComponents(g);
    int size = 117;
    for (int y = 0; y < 3; y++) {
        for (int x = 0; x < 3; x++) {
            String str = Integer.toString(data[y][x]);
            int dx = x * size;
            int dy = y * size;
            g.drawRect(dx, dy, size, size);
            if (data[y][x] != 0) {
                g.drawString(str,  x: dx + size / 2,  y: dy + size / 2);
            }
        }
    }
}
```

숫자 패드 한 개의 크기를 마우스 클릭 범위와 똑같이 117로 지정한다.
Java에서는 y축을 먼저 쓰기 때문에 y에 대해 먼저 써준다. 현재 easy모드가 3*3이기 때문에 y와 x의 범위를 0부터 3까지로 정한다. 사각형을 출력한다. (x좌표, y좌표, 너비, 높이 순)
만약 숫자 패드의 자리가 0, 즉 빈칸이 아니라면, 문자열을 입력해준다.(출력할 문자열, 문자열을 출력할 x좌표, 문자열을 출력할 y좌표 순)

좌표의 숫자가 위에서 선언해주었던 정답 패드의 숫자 n과 같지 않으면 다시 정답을 확인하도록 한다. 정답 패드와 숫자가 똑같이 배열되면 'success!'라는 제목을 가진 알림창을 이용해 '*^^*'라는 메세지를 출력한다.

```java
public void move(int x, int y) {
    try {
        if ((y - 1) >= 0) {
            if (data[y - 1][x] == 0) {
                data[y - 1][x] = data[y][x];
                data[y][x] = 0;
                repaint();
                return;
            }
        }
        if ((y + 1) <= 2) {
            if (data[y + 1][x] == 0) {
                data[y + 1][x] = data[y][x];
                data[y][x] = 0;
                repaint();
                return;
            }
        }
        if ((x + 1) <= 2) {
            if (data[y][x + 1] == 0) {
                data[y][x + 1] = data[y][x];
                data[y][x] = 0;
                repaint();
                return;
            }
        }
        if ((x - 1) >= 0) {
            if (data[y][x - 1] == 0) {
                data[y][x - 1] = data[y][x];
                data[y][x] = 0;
                repaint();
                return;
            }
        }
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
```

-위에 빈칸이 있을 때: [y-1][x]가 빈칸이라면 [y][x]에 있던 패드를 빈칸으로 옮기고 [y][x]패드에 빈칸을 생성한다.

-아래에 빈칸이 있을 때: [y+1][x]가 빈칸이라면 [y][x]에 있던 패드를 빈칸으로 옮기고 [y][x]패드에 빈칸을 생성한다.

-오른쪽에 빈칸이 있을 때: [y][x+1]가 빈칸이라면 [y][x]에 있던 패드를 빈칸으로 옮기고 [y][x]패드에 빈칸을 생성한다.

-왼쪽에 빈칸이 있을 때: [y][x-1]가 빈칸이라면 [y][x]에 있던 패드를 빈칸으로 옮기고 [y][x]패드에 빈칸을 생성한다.

```
public void init() {
    int count, x, y, ran;
    int temp;
    for (count = 0; count < 100; count++) {
        x = rand.nextInt( bound: 3);
        y = rand.nextInt( bound: 3);
        ran = rand.nextInt( bound: 4);
        try {
            if ((x + 1) <= 2 || (x - 1) >= 0 || (y + 1) <= 2 || (y - 1) >= 0) {
                if (ran == 0) {
                    if ((x - 1) >= 0) {
                        temp = data[y][x - 1];
                        data[y][x - 1] = data[y][x];
                        data[y][x] = temp;
                        repaint();
                        return;
                    }
                }
                if (ran == 1) {
                    if ((x + 1) < 3) {
                        temp = data[y][x + 1];
                        data[y][x + 1] = data[y][x];
                        data[y][x] = temp;
                        repaint();
                        return;
                    }
                }
                if (ran == 2) {
                    if ((y - 1) >= 0) {
                        temp = data[y - 1][x];
                        data[y - 1][x] = data[y][x];
                        data[y][x] = temp;
                        repaint();
                        return;
                    }
                }
                if (ran == 3) {
                    if ((y + 1) < 3) {
                        temp = data[y + 1][x];
                        data[y + 1][x] = data[y][x];
                        data[y][x] = temp;
                        return;
                    }
                }
            }
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

랜덤으로 숫자를 배열해주는 init함수를 생성한다. x, y는 easy모드가 3*3이므로 랜덤으로 3을 써주고, ran은 위 아래 양 옆 네 방향 중 하나를 랜덤으로 하는 것이므로 4를 써준다. 순서대로 왼쪽, 오른쪽, 위, 아래 순서이고 랜덤으로 숫자를 배열해주도록 한다.

(Normal, Hard클래스도 이와 같고 변수의 이름과 숫자만 다르므로 설명을 생략한다.)

② **시작 화면**

```
public class Start extends JFrame {
    private JButton start = new JButton( text: "게임시작");
    private JButton how = new JButton( text: "게임방법");

    public Start() {
        setTitle("순서맞추기게임");
        setSize( width: 351, height: 400);
        setLayout(null);
        setLocationRelativeTo(null);
```

난이도 선택으로 이동하는 start버튼과 게임 방법으로 이동하는 how버튼을 추가한다. "순서맞추기게임'이라는 제목을 프레임에 달고, 크기는 너비 351, 높이 400으로 지정하며 프레임은 화면 중앙에 뜨도록 하고, 종료 버튼을 눌렀을 때 화면에서만 사라지는 것이 아닌 프로세스 상에서도 종료되도록 한다.

```
        add(start); start.setSize( width: 300, height: 100); start.setLocation( x: 25, y: 70);
        start.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ea) {
                try{
                    dispose();
                    nextFrame();
                }catch(Exception e){}
            }
        });

        add(how); how.setSize( width: 300, height: 100); how.setLocation( x: 25, y: 190);
        how.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ea) {
                try{
                    dispose();
                    howFrame();
                }catch(Exception e){}
            }
        });

        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
```

Start버튼의 크기는 너비300, 높이100으로 지정하고 위치는 x:25, y:70으로 지정한다. Start버튼을 누르면 게임 첫 시작 프레임이 종료되고 다음 프레임인 난이도 선택 프레임이 켜지도록 한다.
how버튼의 크기는 너비300, 높이100으로 지정하고 위치는 x:25, y:190으로 지정한다. how버튼을 누르면 게임 첫 시작 프레임이 종료되고 게임설명 프레임이 켜지도록 한다.

```
    void nextFrame(){
        Select select = new Select();
        select.setVisible(true);
    }

    void howFrame(){
        How how = new How();
        how.setVisible(true);
    }

    public static void main(String[] args) { Start start = new Start(); }
}
```

위에서 설명한 것처럼 start버튼을 눌렀을 때 켤 nextFrame을 선언해준다. 난이도 선택 프레임인 select프레임을 열게 하고, select프레임이 화면에 보이도록 한다.

위에서 설명한 것처럼 how버튼을 눌렀을 때 켤 howFrame을 선언해준다. 난이도 선택 프레임인 게임설명 프레임을 열게 하고, 게임설명 프레임이 화면에 보이도록 한다.

③ 게임 방법 화면

```
public class How extends JFrame{
    private JButton back = new JButton( text: "뒤로가기");
    private JButton start = new JButton( text: "게임시작");
    JLabel text = new JLabel( text: "<html>숫자를 순서대로 배열하는 게임이다.<br>작은 숫자부터 차례로 왼쪽에서부터 오른쪽으로 배열한다.<br>" +
            "마지막 자리에는 빈칸이 오도록 한다.<br>" +
            "<br>(예시)<br>[EASY MODE]<br>1 2 3<br>4 5 6<br>7 8 0<br><br>0의 자리에 빈 칸이 오도록 한다.</html>", SwingConstants.CENTER);
```

게임 실행 첫 화면으로 이동하는 back버튼과 난이도 선택으로 이동하는 start버튼을 추가한다.

Java의 JLabel 줄바꿈하기 위해선 '₩n'만으로는 해결되지 않는다. JLabel에서 html을 그대로 보여줄 수 있기 때문에 html을 활용해 줄바꿈을 해준다.

```
public How() {
    setTitle("게임방법");
    setSize( width: 351,  height: 400);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    add( name: "North", back);
    back.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ea) {
            try {
                dispose();
                backFrame();
            } catch (Exception e) { }
        }
    });
    add( name: "South", start);
    start.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ea) {
            try {
                dispose();
                selectFrame();
            } catch (Exception e) { }
        }
    });

    add( name: "Center", text);
    setVisible(true);
}
```

"게임방법'이라는 제목을 프레임에 달고, 크기는 너비 351, 높이 400으로 지정하며 프레임은 화면 중앙에 뜨도록 하고, 종료 버튼을 눌렀을 때 화면에서만 사라지는 것이 아닌 프로세스 상에서도 종료되도록 한다.

Back버튼을 화면 프레임 위에 위치하도록 하고, 클릭하면 게임설명 프레임이 종료되고 이전 프레임인 게임 시작 첫 화면 프레임이 켜지도록 한다.

start버튼을 화면 프레임 아래에 위치하도록 하고, 클릭하면 게임설명 프레임이 종료되고 난이도 선택 프레임이 켜지도록 한다.

Text를 프레임 중간에 위치하도록 한다.

```
void backFrame() {
    Start start = new Start();
    start.setVisible(true);
}
void selectFrame(){
    Select select = new Select();
    select.setVisible(true);
}
    public static void main(String[] args) { How how = new How(); }
}
```

위에서 설명한 것처럼 back버튼을 눌렀을 때 켤 backFrame을 선언해준다. 게임 첫 시작 화면 프레임을 열게 하고, 게임 첫 시작 화면 프레임이 화면에 보이도록 한다.

위에서 설명한 것처럼 start버튼을 눌렀을 때 켤 selectFrame을 선언해준다. 난이도 선택 프레임을 열게 하고, 난이도 선택 프레임이 화면에 보이도록 한다.

④ 난이도 선택 화면

```
public class Select extends JFrame {
    private JButton easy = new JButton( text: "EASY(3*3)");
    private JButton normal = new JButton( text: "NORMAL(4*4)");
    private JButton hard = new JButton( text: "HARD(5*5)");
    private JButton back = new JButton( text: "뒤로가기");

    public Select() {
        this.setTitle("난이도선택");
        this.setSize( width: 351, height: 400);
        this.setLayout(null);
        this.setLocationRelativeTo(null);
```

비 351, 높이 400으로 지정하며 프레임은 화면 중앙에 뜨도록 하고, 종료 버튼을 눌렀을 때 화면에서만 사라지는 것이 아닌 프로세스 상에서도 종료되도록 한다.

```
        add(easy); easy.setSize( width: 330, height: 50); easy.setLocation( x: 10, y: 20);
        add(normal); normal.setSize( width: 330, height: 50); normal.setLocation( x: 10, y: 80);
        add(hard); hard.setSize( width: 330, height: 50); hard.setLocation( x: 10,  y: 140);
        add(back); back.setSize( width: 330, height: 50); back.setLocation( x: 10,  y: 300);

    easy.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ea) {
            try{
                dispose();
                easyFrame();
            }catch(Exception a){}
        }
    });

    normal.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent no) {
            try{
                dispose();
                normalFrame();;
            }catch(Exception a){}
        }
    });

    hard.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent h) {
            try{
                dispose();
                hardFrame();
            }catch(Exception a){}
        }
    });
    back.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ea) {
            try{
                dispose();
                backFrame();
            }catch(Exception e){}
        }
    });

    this.setVisible(true);
    this.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
```

easy버튼의 크기는 너비330, 높이50으로 지정하고 위치는 x:10, y:20으로 지정한다. easy버튼을 누르면 난이도 선택 프레임이 종료되고 다음 프레임인 easymode게임 프레임이 켜지도록 한다.

normal버튼의 크기는 너비330, 높이50으로 지정하고 위치는 x:10, y:80으로 지정한다. normal버튼을 누르면 난이도 선택 프레임이 종료되고 다음 프레임인 normalmode게임 프레임이 켜지도록

한다.
hard버튼의 크기는 너비330, 높이50으로 지정하고 위치는 x:10, y:140으로 지정한다. hard버튼을 누르면 난이도 선택 프레임이 종료되고 다음 프레임인 hardmode게임 프레임이 켜지도록 한다.
back버튼의 크기는 너비330, 높이50으로 지정하고 위치는 x:10, y:300으로 지정한다. back버튼을 누르면 난이도 선택 프레임이 종료되고 이전 프레임인 게임시작화면 프레임이 켜지도록 한다.

```
    void easyFrame(){
        Easy easy = new Easy();
        easy.setVisible(true);
    }

    void normalFrame(){
        Normal normal = new Normal();
        normal.setVisible(true);
    }

    void hardFrame(){
        Hard hard = new Hard();
        hard.setVisible(true);
    }
    void backFrame(){
        Start start = new Start();
        start.setVisible(true);
    }
    public static void main(String[] args) { Select select = new Select(); }
}
```

위에서 설명한 것처럼 Easy버튼을 눌렀을 때 켤 easyFrame을 선언해준다. Easymode 게임 프레임을 열게 하고, easymode 게임 프레임이 화면에 보이도록 한다.
위에서 설명한 것처럼 normal버튼을 눌렀을 때 켤 normalFrame을 선언해준다. normalmode 게임 프레임을 열게 하고, normalmode 게임 프레임이 화면에 보이도록 한다.
위에서 설명한 것처럼 hard버튼을 눌렀을 때 켤 hardFrame을 선언해준다. hardmode 게임 프레임을 열게 하고, hardmode 게임 프레임이 화면에 보이도록 한다.
위에서 설명한 것처럼 back버튼을 눌렀을 때 켤 backFrame을 선언해준다. 게임 첫 시작 화면 프레임을 열게 하고, 게임 첫 시작 화면 프레임이 화면에 보이도록 한다.

2) 기능 설명

-난이도 조절
:난이도에 따라 패드의 개수를 다르게 설정한다. Easy모드의 경우 3*3으로 9개, normal모드의 경우 4*4로 16개, hard모드의 경우 5*5로 25개로 설정한다.

-버튼 추가
:난이도를 선택하여 게임을 진행할 수 있도록 게임시작 버튼, 난이도 선택 버튼, 뒤로 가기 버튼을 추가한다. 더불어 게임을 이해하지 못할 사람들을 위해 게임 방법에 대한 프레임을 담은 버튼을 추가한다.

3) 예외 케이스

1.

```
easy.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ea) {
        try{
            dispose();
            easyFrame();
        }catch(Exception a){}
    }
});
```

위 사진과 같은 try-catch문에서 try문 안에 들어있는 내용이 예외 처리 될 경우 catch문을 실행한다. 위와 같은 형식의 try-catch문에서 catch문에는 아무런 내용이 없으므로 Try문이 예외 처리 될 경우 아무것도 실행하지 않는다.

2.

Easy/normal/hard mode에 있는 move와 init에서도 예외 처리되는 경우가 있다. 위,아래,양 옆에 빈 칸이 없는 경우, 위, 아래, 양 옆이 없는 경우 에러 코드를 출력하도록 한다.

3. 클래스설명

1) 클래스 명세표

클래스 명	Start
필드변수	start : JButton how : JButton
메소드	nextFrame() : void howFrame() : void main(args:String[]) : void
MVC	MV

클래스 명	How
필드변수	start : JButton back : JButton text : JLabel
메소드	backFrame() : void selectFrame() : void main(args:String[]) : void
MVC	MV

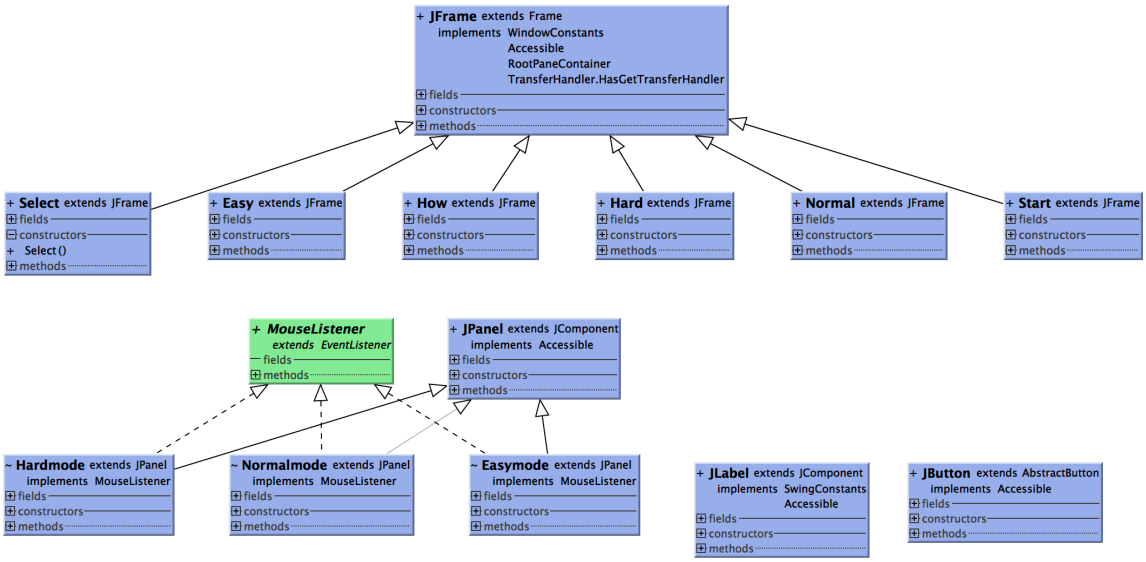
클래스 명	Select
필드변수	easy : JButton normal : JButton hard : JButton back : JButton
메소드	easyFrame() : void normalFrame() : void hardFrame() : void backFrame() : void main(args:String[]) : void
MVC	MV

클래스 명	Easy
필드변수	back : JButton easymode : Easymode
메소드	backFrame() : void main(args:String[]) : void
MVC	MVC

클래스 명	Normal
필드변수	back : JButton normalmode : Normalmode
메소드	backFrame() : void main(args:String[]) : void
MVC	MVC

클래스 명	Hard
필드변수	back : JButton hardmode : Hardmode
메소드	backFrame() : void main(args:String[]) : void
MVC	MVC

2) 클래스 구조도



3) 추상 클래스 및 인터페이스 사용 내역

인터페이스 사용 내역
Accessible
SwingConstants
MouseListener

4. **아쉬운 점**

처음에 팀을 하기로 했던 친구가 바쁘다는 핑계로 계속해서 만남을 미루어서 혼자서만 일을 진행하다가 결국엔 팀을 해체하고 혼자 게임 구성 및 코딩까지 다 진행해야만 했다. 모든걸 처음부터 다시 하려니 시간이 별로 없었다. 타이머 기능과 랭킹 기능을 넣으려고 했는데 그러기엔 시간이 많이 부족해서 많이 아쉬웠다.

결론

- 난이도를 세 개로 나눔으로써 게임 사용자들이 다양한 난이도의 게임을 실현해볼 수 있게 하는 효과를 가져옴.
- 버튼을 추가해 게임 시작 페이지, 난이도 선택 페이지, 게임 페이지로 나눔으로써 편의성을 높임.
- 간단한 프레임 구조와 그에 따른 패널 구조로, 유지하기도 편리하고 난이도 별로 나누어져 있어 수정하기 편함. 프레임을 추가하면 새로운 게임을 추가한 후 실행할 수 있어

한 개로 묶어 코드를 작성한 것보다 나누어 작성한 것이 더 편했음.
- '게임 시작 – (게임방법) - 난이도 선택 – 게임 화면'의 구조가 잘 연결되었고, 게임 실행도 오류없이 잘 됨.