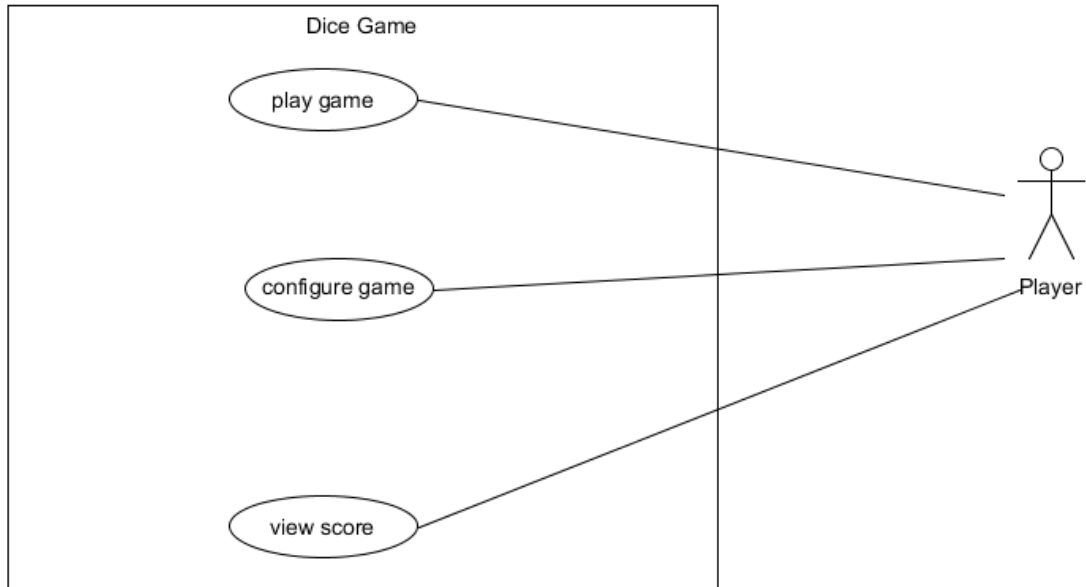


# DICE GAME

1592073 조성권

## 1. 유스케이스 다이어그램



## 2. 유스케이스 기술서

표 2-15 유스케이스 기술서(play game)

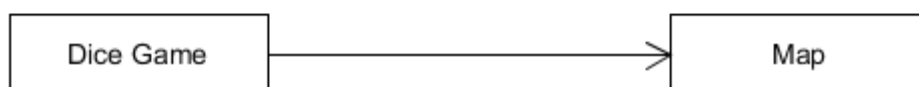
유스케이스명	play game
액터명	주 액터: 플레이어(Player)
개요	플레이어가 게임을 하기위해 시스템을 사용한다.
사전조건	
사후조건	• 플레이어의 스코어가 기록된다.
기본흐름	<ol style="list-style-type: none"> <li>1. 플레이어는 자신의 이름을 입력하고 게임을 시작한다.</li> <li>2. 시스템은 주사위 두 개를 준비한다. 하나는 플레이어가 사용할 주사위이고 다른 하나는 알파다이스가 사용할 주사위이다.</li> <li>3. 플레이어가 주사위를 던진다.</li> <li>4. 시스템은 주사위 숫자만큼 칸을 이동한다.</li> <li>5. 만약 도달한 칸이 이벤트 칸이라면 플레이어를 그 만큼 전진하거나 후퇴시킨다.</li> <li>6. 시스템은 다른 주사위를 던져 알파다이스를 주사위 숫자만큼 칸을 이동시킨다.</li> <li>7. 만약 도달한 칸이 이벤트 칸이라면 그 만큼 전진하거나 후퇴한다.</li> <li>8. 플레이어나 알파다이스가 목표 칸에 도달할 때 까지 「3번 단계-7번 단계」를 반복한다.</li> <li>9. 시스템은 승자를 알려주고 플레이어의 스코어를 기록한다.</li> </ol>

표 2-16 유스케이스 기술서(view score)

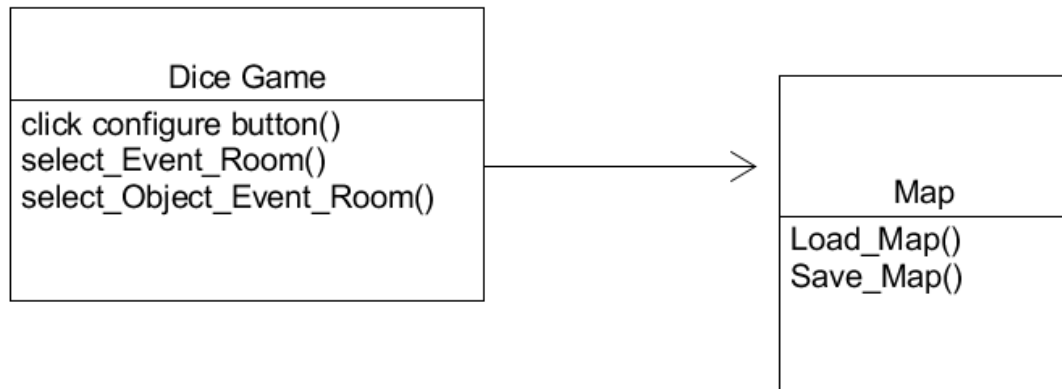
유스케이스명	view score
액터명	주 액터: 플레이어
개요	플레이어가 스코어를 조화하기 위해 시스템을 사용한다.
사전조건	play game
사후조건	
기본흐름	1. 플레이어는 스코어 목록을 요청한다. 2. 시스템은 스코어 목록을 보여준다.

유스케이스명	configure game
액터명	주 액터:player
개요	플레이어가 맵 정보를 불러오기 위해 시스템을 이용한다.
사전조건	
사후조건	
기본흐름	1.configure 버튼을 클릭 2.주사위 게임의 맵의 구조를 불러온다. 3.이벤트 칸을 선택한다. 4.이벤트 칸의 목표 칸을 선택한다. 5.이벤트를 맵의 구조에 저장한다.
맵, 주사위 게임	클래스 후보
구조	맵 속성
이벤트 칸, 목표 칸	주사위 게임 속성

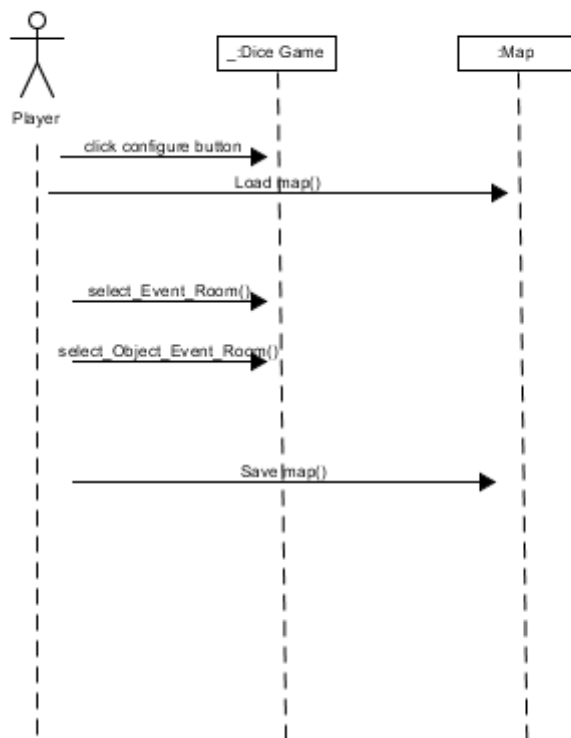
분석 클래스 다이어그램



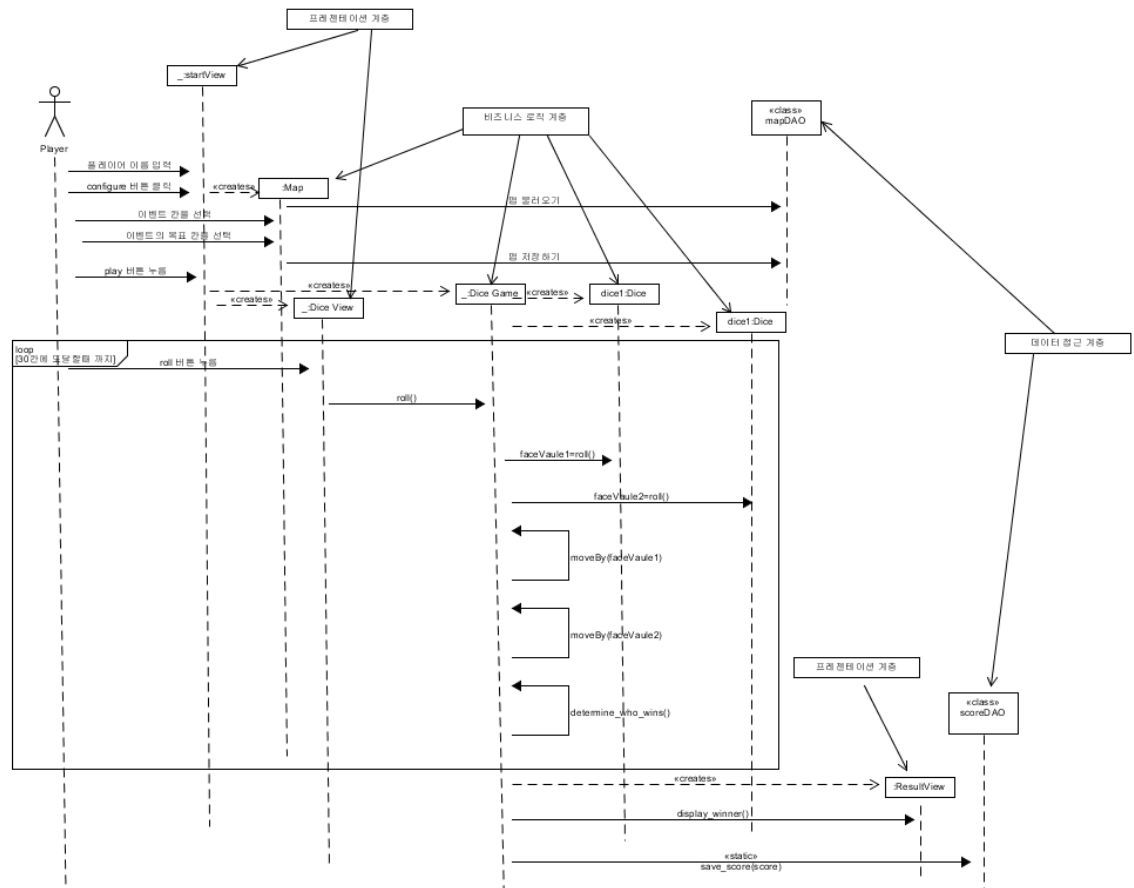
책임이 할당된 클래스 다이어그램



configure game 분석 순차 다이어그램



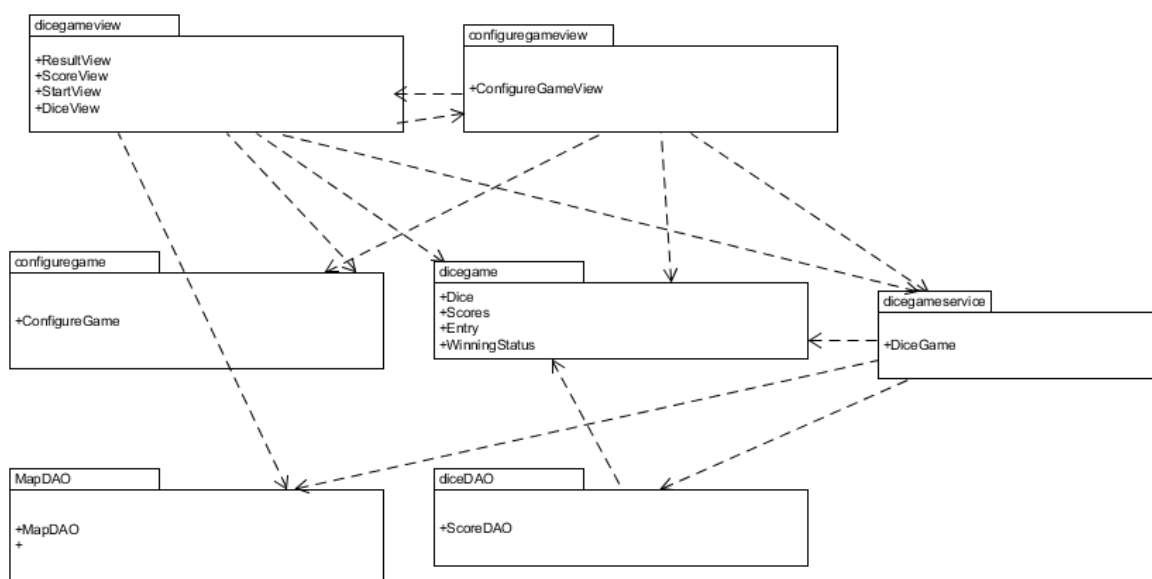
## configure game 설계 순차 다이어그램



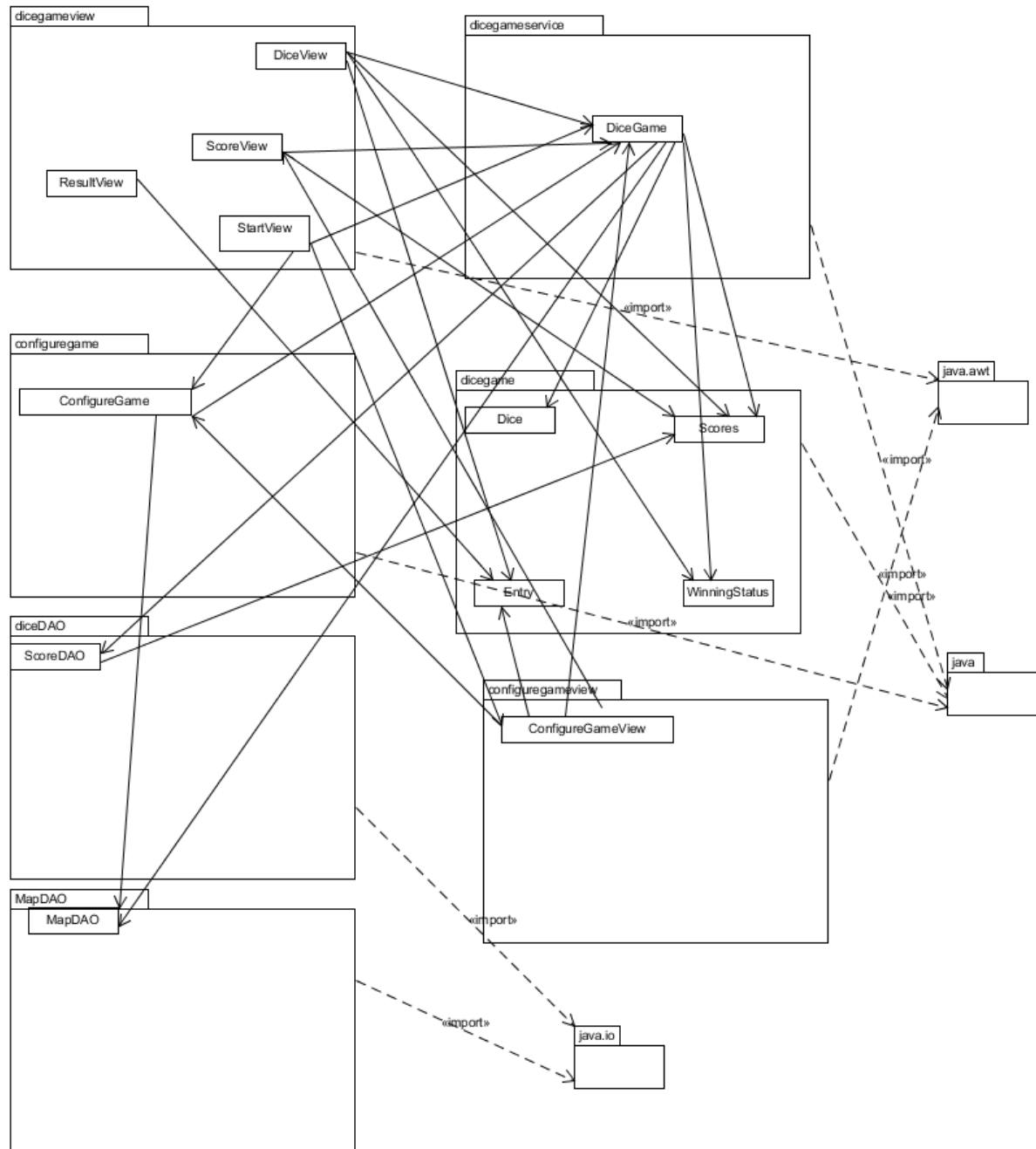
## 설계 및 구현 매커니즘의 결정

StartView,DiceView,ResultView,ScoreView,ConfigureView	Java AWT
DiceGame,Dice,Scores,Entry,ConfigureGame	Java
ScoreDAO,MapDAO	Java 파일 매커니즘

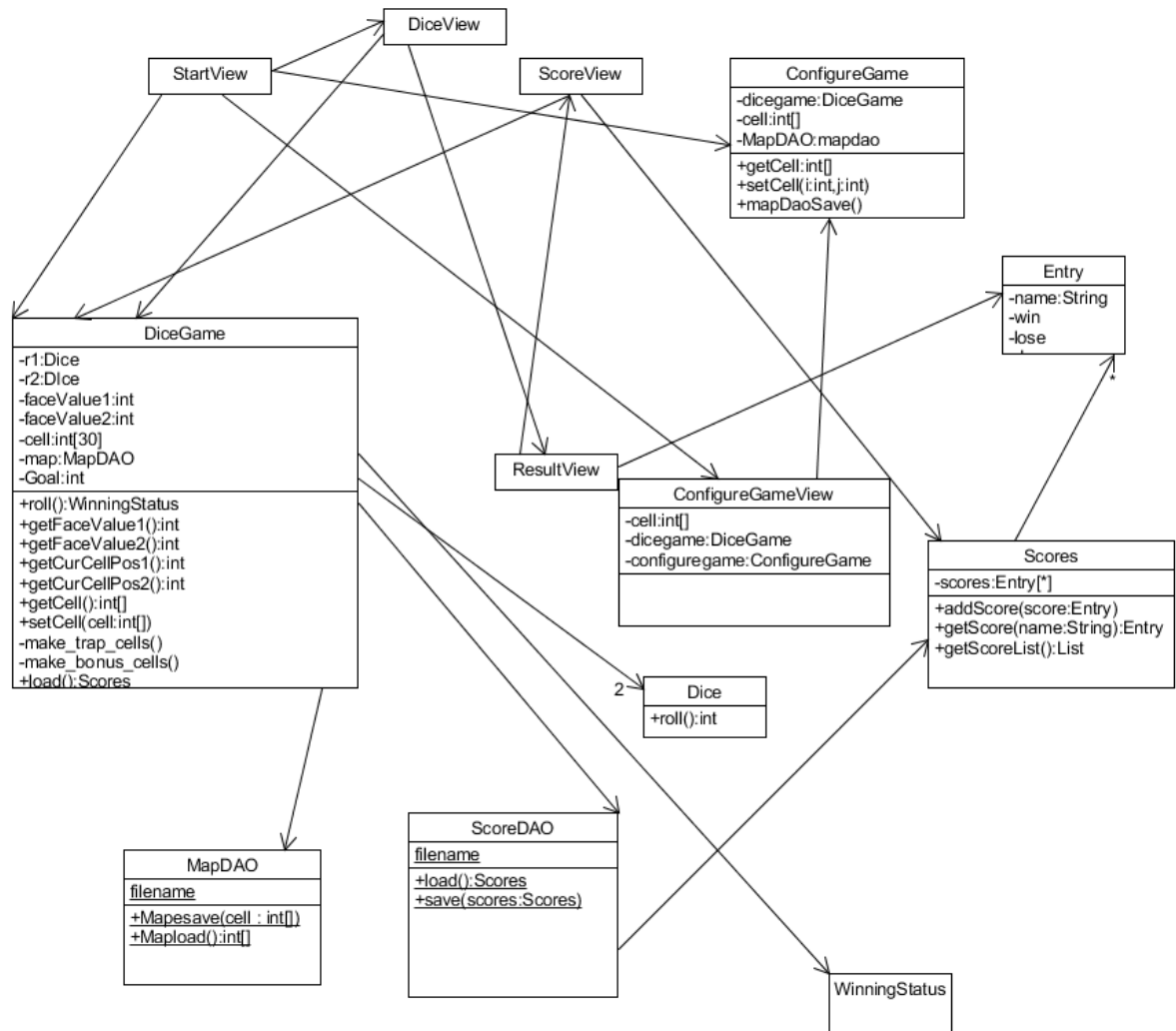
## 분석 패키지 다이어그램



## 설계 패키지 다이어그램



# 주사위 게임 설계 클래스 다이어그램





코드 및 실행

mapDAO

```
package mapDAO;
```

```
import java.io.BufferedReader;
```

```
import java.io.File;
```

```
import java.io.FileNotFoundException;
```

```
import java.io.FileOutputStream;
```

```
import java.io.FileReader;
```

```
import java.io.IOException;
```

```
public class MapDAO {
```

```
    static String filename = "map.txt";
```

```
    public static void Mapsave(int[] cell) throws Exception {
```

```
        FileOutputStream output = new FileOutputStream("E:/app/조성권  
/DiceGame/bin/map.txt");
```

```
        for (int i = 0; i < 30; i++) {
```

```
            String data = cell[i] + "WrWn";
```

```
            output.write(data.getBytes());
```

```
        }
```

```
        output.close();
```

```
    }
```

```
public static int[] Mapload() throws IOException {  
  
    int[] cell;  
  
    try {  
  
        BufferedReader br = new BufferedReader(new FileReader("E:/app/조성권  
/DiceGame/bin/map.txt"));  
  
        cell = new int[30];  
  
        while (true) {  
  
            int i = 0;  
  
            String line = br.readLine();  
  
            if (line == null)  
  
                break;  
  
            cell[i++] = Integer.valueOf(line);  
  
        }  
  
        br.close();  
  
        return cell;  
  
    } catch (FileNotFoundException e) {  
  
        File mapFile = new File("map.txt");  
  
        return null;  
  
    }  
  
}
```

## StartView

```
package dicegameview;

import java.awt.Button;

import java.awt.FlowLayout;

import java.awt.Frame;

import java.awt.Panel;

import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import configuregame.ConfigureGame;

import configuregameview.ConfigureGameView;

import dicegameservice.DiceGame;

public class StartView extends Frame {

    TextField nametf;

    String name;

    DiceGame dicegame;

    ConfigureGame configuregame;

    Panel p1, p2;
```

```
public StartView(String title) {  
  
    super(title);  
  
    Button configBut = new Button("Configure");  
  
    Button playBut = new Button("Play");  
  
    Button exitBut = new Button("Exit");  
  
  
    p1 = new Panel();  
  
    p2 = new Panel();  
  
  
    nametf = new TextField(20);  
  
    setLayout(new FlowLayout());  
  
  
    p1.add(nametf);  
  
    p1.add(playBut);  
  
    add(p1);  
  
  
    p2.add(configBut);  
  
    p2.add(exitBut);  
  
    add(p2);  
  
  
    playBut.addActionListener(new ActionListener() {  
  
        @Override
```

```

public void actionPerformed(ActionEvent e) {

    if (dicegame == null)

        try {

            dicegame = new DiceGame();

        } catch (Exception e1) {

            // TODO Auto-generated catch block

            e1.printStackTrace();

        }

    name = nametf.getText();

    new DiceView(name, dicegame);

    setVisible(false);

    dispose();

}

});

```

// configure를 하게 하는 이벤트 시작

```

configBut.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        if (dicegame == null)

            try {

                dicegame = new DiceGame();

            } catch (Exception e2) {

```

```

        // TODO Auto-generated catch block
        e2.printStackTrace();
    }

    if(configuregame == null)

        try {

            configuregame = new
ConfigureGame(dicegame);

        } catch (Exception e1) {

            // TODO Auto-generated catch block
            e1.printStackTrace();

        }

    try {

        new ConfigureGameView(dicegame);

    } catch (Exception e1) {

        // TODO Auto-generated catch block
        e1.printStackTrace();

    }

}

});

exitBut.addActionListener(new ActionListener() {

```

```
        @Override

        public void actionPerformed(ActionEvent arg0) {

            System.exit(0);

        }

    });

    setSize(300, 150);

    setVisible(true);

}

}
```

ScoreView

```
package dicegameview;

import java.awt.BorderLayout;

import java.awt.Button;

import java.awt.Frame;

import java.awt.List;

import java.awt.Panel;

import java.awt.ScrollPane;

import java.awt.event.ActionEvent;

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

```
import dicegame.Scores;
```

```
import dicegameservice.DiceGame;
```

```
public class ScoreView extends Frame {
```

```
    ScrollPane scroll;
```

```
    DiceGame dicegame;
```

```
    Panel p1, p2;
```

```
    Button exitBut;
```

```
    public ScoreView() throws Exception {
```

```
        super("Score View");
```

```
        dicegame = new DiceGame();
```

```
        p1 = new Panel();
```

```
        p2 = new Panel();
```

```
        ScrollPane scroll = new ScrollPane();
```

```
        exitBut = new Button("Exit");
```

```
        Scores scores = dicegame.load();
```

```
        final List list = scores.getList();
```

```
        scroll.add(list);
```

```
        p1.add(scroll);
```

```
        p2.add(exitBut);
```

```
        add(p1, BorderLayout.NORTH);
```



```
add(p2, BorderLayout.CENTER);
```

```
scroll.setSize(250, 100);
```

```
setSize(300, 200);
```

```
setVisible(true);
```

```
exitBut.addActionListener(new ActionListener() {
```

```
    @Override
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        setVisible(false);
```

```
        dispose();
```

```
    }
```

```
});
```

```
}
```

```
}
```

```
ResultView
```

```
package dicegameview;
```

```
import java.awt.Button;
```

```
import java.awt.FlowLayout;
```

```
import java.awt.Frame;
```

```
import java.awt.Label;
```

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

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

```
import dicegame.Entry;
```

```
public class ResultView extends Frame {
```

```
    private Label message;
```

```
    public ResultView(final String msg, final Entry entry) {
```

```
        super("Result View");
```

```
        message = new Label(msg);
```

```
        Button scoreBut = new Button("Score");
```

```
        Button cancelBut = new Button("Exit");
```

```
        setLayout(new FlowLayout());
```

```
        add(message);
```

```
        add(scoreBut);
```

```
        add(cancelBut);
```

```
        message.setText(msg);
```

```
        scoreBut.addActionListener(new ActionListener() {
```

```

@Override

public void actionPerformed(ActionEvent e) {

    try {

        new ScoreView();

    } catch (Exception e1) {

        // TODO Auto-generated catch block

        e1.printStackTrace();

    }

}

});

```

```

cancelBut.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        setVisible(false);

        dispose();

    }

});

```

```

setSize(500, 100);

setVisible(true);

```

```

}

```

```
}
```

DiceView

```
package dicegameview;

import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import dicegame.Entry;
import dicegame.Scores;
import dicegame.WinningStatus;
import dicegameservice.DiceGame;

public class DiceView extends Frame {

    private TextField dice1;
    private TextField dice2;
    private Label cell1;
    private Label cell2;
    private DiceGame dicegame;

    Panel p1, p2;

    public DiceView(final String playerName, final DiceGame dicegame) {
        super("Dice View");
        this.dicegame = dicegame;

        dice1 = new TextField(5);
        dice2 = new TextField(5);
        cell1 = new Label();
        cell2 = new Label();

        p1 = new Panel();
        p2 = new Panel();

        final Button rollBut = new Button("Roll");
        p1.add(rollBut);
        Button cancelBut = new Button("Exit");
        setLayout(new FlowLayout());
        p1.add(new Label(playerName + " face value"));
        p1.add(dice1);
        p1.add(new Label("AlphaDice face value"));
        p1.add(dice2);
```

```

p1.add(rollBut);
p1.add(cancelBut);

p2.setLayout(new GridLayout(2, 2));
p2.add(new Label(playerName));
p2.add(cell1);
p2.add(new Label("AlphaDice"));
p2.add(cell2);

setLayout(new BorderLayout());
add(p1, BorderLayout.EAST);
add(p2, BorderLayout.WEST);

rollBut.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        WinningStatus ws = dicegame.roll();

dice1.setText(String.valueOf(dicegame.getFaceValue1()));

dice2.setText(String.valueOf(dicegame.getFaceValue2()));
        if (ws == WinningStatus.NotYet) {

cell1.setText(String.valueOf(dicegame.getCurCellPos1()));

cell2.setText(String.valueOf(dicegame.getCurCellPos2()));
            } else {
                String message;
                Scores scores = dicegame.load();
                Entry entry;
                if (scores == null) {
                    scores = new Scores();
                    entry = new Entry(playerName, 0, 0, 0);
                    scores.addScore(entry);
                }

                entry = scores.getEntry(playerName);

                if (ws == WinningStatus.Draw) {
                    entry.setDraw(entry.getDraw() + 1);
                    message = "Draw";
                } else if (ws == WinningStatus.Player) {
                    entry.setWin(entry.getWin() + 1);
                    message = playerName + " wins";
                } else {
                    entry.setLose(entry.getLose() + 1);
                    message = "AlphaDice wins";
                }

                dicegame.save(scores);
                new ResultView(message, entry);
                setVisible(false);
                dispose();
            }
        }
    });

```

```

cancelBut.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        setVisible(false);
        dispose();
    }

});

setSize(600, 100);
setVisible(true);
}
}

```

DiceGame

```
package dicegameservice;
```

```
import diceDAO.ScoreDAO;
```

```
import dicegame.Dice;
```

```
import dicegame.Scores;
```

```
import dicegame.WinningStatus;
```

```
import mapDAO.MapDAO;
```

```
public class DiceGame {
```

```
    private Dice r1;
```

```
    private Dice r2;
```

```
    private int faceValue1;
```

```
    private int faceValue2;
```

```
    private int [] cell;
```

```
    private int curCell1;
```

```
private int curCell2;
```

```
private MapDAO map;
```

```
final private int Goal = 29;
```

```
public DiceGame() throws Exception {
```

```
    r1 = new Dice();
```

```
    r2 = new Dice();
```

```
    faceValue1 = faceValue2 = curCell1 = curCell2 = 0;
```

```
    cell = new int [30];
```

```
    for (int i=0;i<30;i++)
```

```
        cell[i] = i;
```

```
    make_trap_cells();
```

```
    make_bonus_cells();
```

```
    map.Mapsave(cell);
```

```
    // 저장 된 cell의 내용을 생성하면서 불러오고 셋팅하는 역할
```

```
}
```

```
public WinningStatus roll() {
```

```
    faceValue1 = r1.roll();
```

```
    faceValue2 = r2.roll();
```

```
curCell1 += faceValue1;
```

```
curCell2 += faceValue2;
```

```
if (curCell1 >= Goal && curCell2 >= Goal)
```

```
    return WinningStatus.Draw;
```

```
else if (curCell1 >= Goal && curCell2 < Goal)
```

```
    return WinningStatus.Player;
```

```
else if (curCell1 < Goal && curCell2 >= Goal)
```

```
    return WinningStatus.AlphaDice;
```

```
else {
```

```
    if (curCell1 != cell[curCell1])
```

```
        curCell1 = cell[curCell1];
```

```
    if (curCell2 != cell[curCell2])
```

```
        curCell2 = cell[curCell2];
```

```
    return WinningStatus.NotYet;
```

```
}
```

```
}
```

```
public int getFaceValue1() {
```

```
    return faceValue1;
```

```
}
```

```
public int getFaceValue2() {
```



```
        return faceValue2;
    }
}
```

```
public int getCurCellPos1() {
    return curCell1;
}
```

```
public int getCurCellPos2() {
    return curCell2;
}
```

// dicegame의 cell을 다른 곳에서 사용하도록 하게끔 해줌

```
public int[] getCell() {
    return cell;
}
```

// 바뀐 cell의 내용을 가져오는 함수

```
public void setCell(int [] cell) {
    this.cell = cell;
}
```

```
private void make_trap_cells() {
    cell[10] = 0;
}
```

```
        cell[28] = 0;

        cell[8] = 3;

        cell[15] = 5;

        cell[21] = 12;

        cell[25] = 17;

    }
```

```
private void make_bonus_cells() {

    cell[11] = 20;

    cell[26] = 27;

    cell[9] = 14;

    cell[16] = 22;

}
```

```
public Scores load() {

    try {

        return ScoreDAO.load();

    } catch (Exception e) {

        e.printStackTrace();

    }

    return null;

}
```

```

        public void save(Scores scores) {

            try {

                ScoreDAO.save(scores);

            } catch (Exception e) {

                e.printStackTrace();

            }

        }

    }
}

```

WinningStatus

```

package dicegame;

public enum WinningStatus {
    NotYet, Player, Draw, AlphaDice
}

```

Scores

```

package dicegame;

```

```

import java.util.*;

```

```

import java.awt.*;

```

```

import java.awt.List;

```

```

import java.io.*;

```

```

public class Scores implements Serializable {

```

```

    private ArrayList<Entry> scores;

```

```
public Scores() {  
  
    scores = new ArrayList<Entry>();  
  
}
```

```
public void addScore(Entry score) {  
  
    scores.add(score);  
  
}
```

```
public Entry getEntry (String playerName) {  
  
    Iterator<Entry> it = scores.iterator();  
  
    Entry entry;  
  
    while (it.hasNext()) {  
  
        entry = it.next();  
  
        if (entry.getName().equals(playerName))  
  
            return entry;  
  
    }  
  
    entry = new Entry(playerName, 0, 0, 0);  
  
    addScore(entry);  
  
    return entry;  
  
}
```

```
public List getList() {  
  
    final List list = new List();
```

```

        Iterator<Entry> it = scores.iterator();

        Entry entry;

        while (it.hasNext()) {

            entry = it.next();

            list.add(entry.getName() + " : " + entry.getWin() + " wins " +
entry.getLose() + " loses " + entry.getDraw() + " draws ");

        }

        return list;

    }

}

```

Entry

```

package dicegame;

import java.io.*;

public class Entry implements Serializable {
    private String name;
    private int win;
    private int lose;
    private int draw;

    public Entry(String playerName, int w, int l, int d) {
        this.name = playerName;
        this.win = w;
        this.lose = l;
        this.draw = d;
    }

    public Entry() {

    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

```

    public int getWin() {
        return win;
    }

    public void setWin(int win) {
        this.win = win;
    }

    public int getLose() {
        return lose;
    }

    public void setLose(int lose) {
        this.lose = lose;
    }

    public int getDraw() {
        return draw;
    }

    public void setDraw(int draw) {
        this.draw = draw;
    }
}

```

Dice

```

package dicegame;

public class Dice {

    public int roll () {
        return (int)(Math.random()*6 + 1);
    }
}

```

scoreDAO

```

package diceDAO;

```

```

import java.io.*;

```

```

import dicegame.Scores;

```

```

public class ScoreDAO {

```

```
static String filename = "score.dat";
```

```
public static void save(Scores scores) throws Exception {
```

```
    FileOutputStream ostream = new FileOutputStream(filename);
```

```
    ObjectOutputStream p = new ObjectOutputStream(ostream);
```

```
    p.writeObject(scores);
```

```
    p.flush();
```

```
    ostream.close();
```

```
}
```

```
public static Scores load() throws Exception {
```

```
    FileInputStream istream;
```

```
    try {
```

```
        istream = new FileInputStream(filename);
```

```
        ObjectInputStream q = new ObjectInputStream(istream);
```

```
        Scores scr = (Scores)q.readObject();
```

```
        istream.close();
```

```
        return scr;
```

```
    } catch (FileNotFoundException e) {
```

```
        File scoreFile = new File("score.dat");
```

```
    }
```

```
    return null;
```

```
    }  
}
```

ConfigureGameView

```
package configuregameview;
```

```
import java.awt.BorderLayout;
```

```
import java.awt.Button;
```

```
import java.awt.FlowLayout;
```

```
import java.awt.Frame;
```

```
import java.awt.Label;
```

```
import java.awt.Panel;
```

```
import java.awt.TextField;
```

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

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

```
import java.io.IOException;
```

```
import configuregame.ConfigureGame;
```

```
import dicegame.Entry;
```

```
import dicegameservice.DiceGame;
```

```
import dicegameview.ScoreView;
```

```
public class ConfigureGameView extends Frame {
```



```
private Label firstNumber;
```

```
private Label secondNumber;
```

```
private TextField firstText;
```

```
private TextField secondText;
```

```
private ConfigureGame configuregame;
```

```
private DiceGame dicegame;
```

```
private int[] cell;
```

```
int num1=0,num2=0;
```

```
Panel p1,p2;
```

```
public ConfigureGameView(final DiceGame dicegame) throws Exception {
```

```
    super("ConfigureGameView");
```

```
    this.dicegame = dicegame;
```

```
    p1 = new Panel();
```

```
    Button changeBut = new Button("Change");
```

```
    Button cancelBut = new Button("Exit");
```

```
    firstNumber = new Label("이벤트 칸");
```

```
    secondNumber = new Label("목표 칸");
```

```
firstText = new TextField(3);
```

```
secondText = new TextField(3);
```

```
setLayout(new FlowLayout());
```

```
p1.add(firstNumber);
```

```
p1.add(firstText);
```

```
p1.add(secondNumber);
```

```
p1.add(secondText);
```

```
p1.add(changeBut);
```

```
p1.add(cancelBut);
```

```
setLayout(new BorderLayout());
```

```
add(p1);
```

```
configuregame = new ConfigureGame(dicegame);
```

```
changeBut.addActionListener(new ActionListener() {
```

```
    @Override
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        num1 = Integer.parseInt(firstText.getText());
```

```
        num2 = Integer.parseInt(secondText.getText());
```

```
        configuregame.setCell(num1,num2);
```

```
        firstText.setText("");

        secondText.setText("");

        try {

            configuregame.mapDaoSave();

        } catch (Exception e1) {

            // TODO Auto-generated catch block

            e1.printStackTrace();

        }

    }

});
```

```
cancelBut.addActionListener(new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        setVisible(false);

        dispose();

    }

});
```

```
setSize(500, 100);

setVisible(true);
```

```
}
```

```
}
```

ConfigureGame

```
package configuregame;
```

```
import java.io.IOException;
```

```
import dicegameservice.DiceGame;
```

```
import mapDAO.MapDAO;
```

```
public class ConfigureGame {
```

```
    private DiceGame dicegame;
```

```
    MapDAO mapdao;
```

```
    int[] cell;
```

```
    public ConfigureGame(DiceGame dicegame) throws IOException{
```

```
        this.dicegame = dicegame;
```

```
        cell = mapdao.Mapload();
```

```
    }
```

```
    public int[] getCell(){
```

```
        return dicegame.getCell();
```

```

    }

    public void setCell(int i, int j){

        cell = getCell();

        cell[i] = j;

        dicegame.setCell(cell);

    }

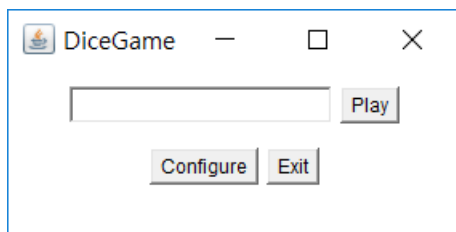
    public void mapDaoSave() throws Exception{

        mapdao.Mapsave(cell);

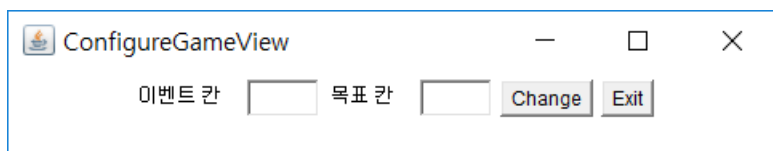
    }

}

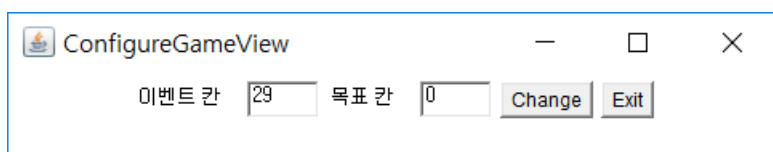
```



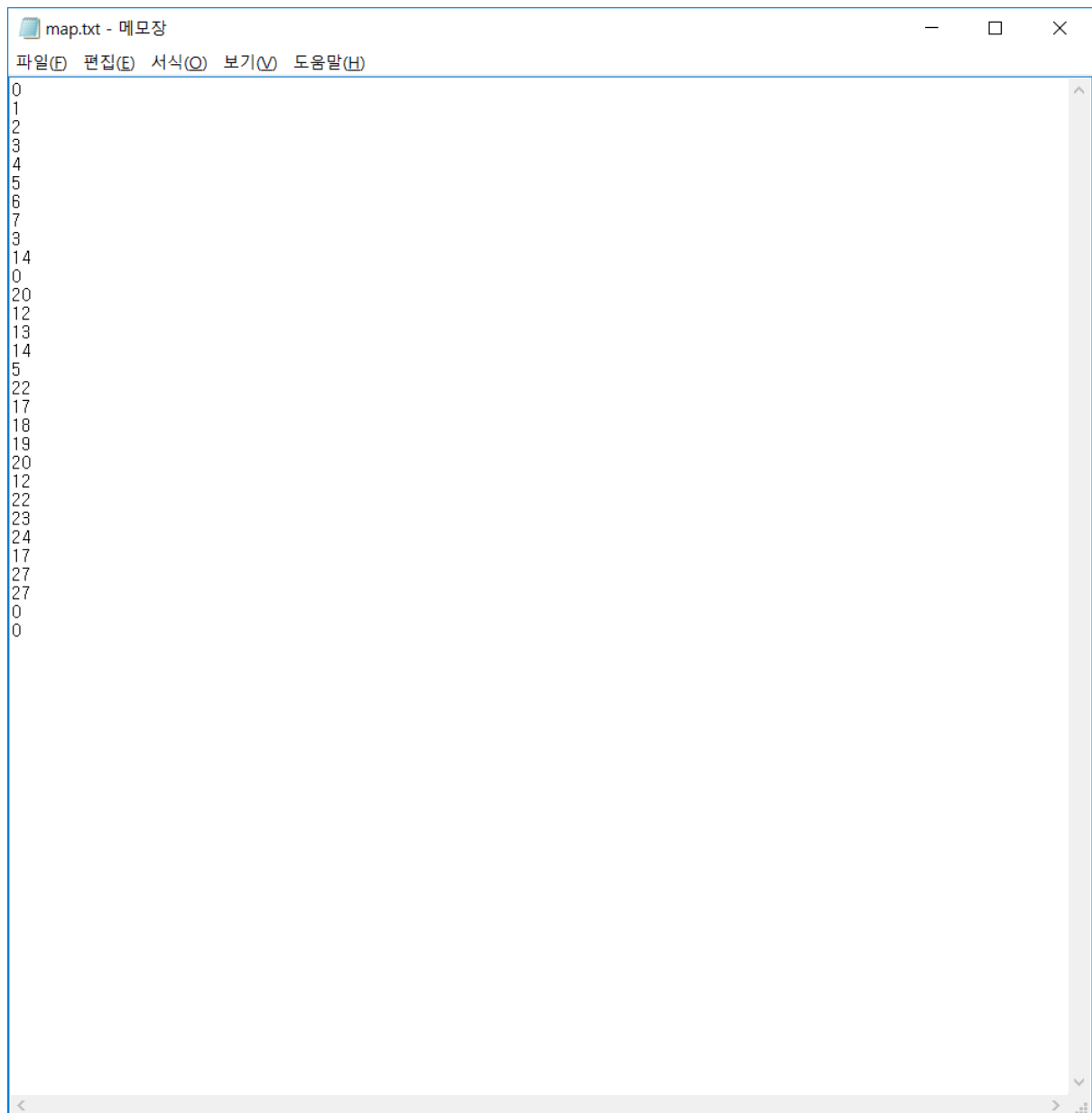
Dicegame startVlew 창입니다.



Configure창입니다.

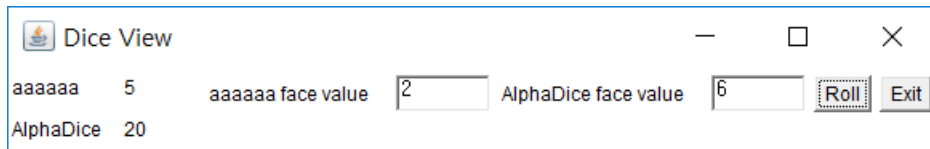
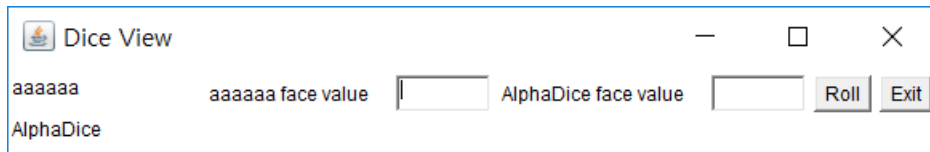


맵의 수정 결과입니다.(29번째 칸(맨 마지막 칸)을 0으로 바꿈)

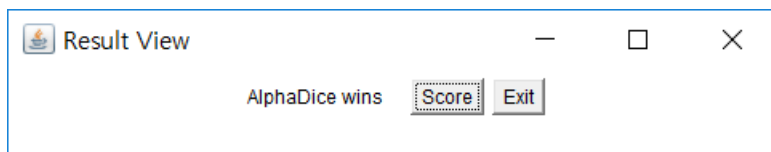


A screenshot of a Notepad window titled "map.txt - 메모장". The window has a menu bar with "파일(F)", "편집(E)", "서식(O)", "보기(V)", and "도움말(H)". The text area contains a list of numbers, mostly 0s, with some other values interspersed. The numbers are: 0, 1, 2, 3, 4, 5, 6, 7, 3, 14, 0, 20, 12, 13, 14, 5, 22, 17, 18, 19, 20, 12, 22, 23, 24, 17, 27, 27, 0, 0. The window has a scrollbar on the right side.

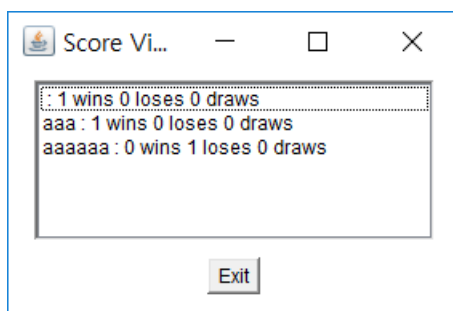
```
0
1
2
3
4
5
6
7
3
14
0
20
12
13
14
5
22
17
18
19
20
12
22
23
24
17
27
27
0
0
```



Dice View 창에서 저와 AlphaDice가 주사위게임을 하고 있습니다.



결과창에 AlphaDice wins라고 나옵니다.



Score View에서는 지금까지 한 아이디의 기록을 나타냅니다.