CommandCenter Class Introduction

How to use

CommandCall (String command) is used to call a command. Its parameter has two types: 1) a series of inputs and 2) the name of an action, as shown in the following examples for calling a fire ball.

```
fakename.CommandCall (“2 3 6 _ A”);
or
fakename.CommandCall (“STAND_D_DF_FA”);
```

For the first type of parameter, each input must be split by space, and they will be executed from left to right. Numbers (2 3 6 in the above example) indicate movement directions as shown in the table below.

7(↖)	8(↑)	9(↗)
4(←)	5(・)	6(→)
1(↙)	2(↓)	3(↘)

(5 indicates the input for not pressing any key)

Moreover, the symbol "_" is used to connect simultaneously pressed inputs. Some skills like throw require pressing direction and attack keys at the same time. In the above example, the direction “6” and attack “A” will be pressed at the same time.

If you use CommandCenter, as in the following example, you can always assume that your character always faces to the right-hand side, so that you can refer to the above table. This is because even though your character's facing direction changes, the system will automatically reverse your command directions.

There is a point to be noticed. While a command is being input, the system will not accept the input of a subsequent command. For this you can use the getSkillFlag command. If a command is being input the flag will return 1, otherwise 0.

A sample AI is given below.

```
1  import aiinterface.CommandCenter;
2  import aiinterface.AIInterface;
3  import struct.FrameData;
4  import struct.GameData;
5  import struct.Key;
6
7  public class SampleAI implements AIInterface {
8
9      Key inputKey;
10     boolean playerNumber;
11     FrameData frameData;
12     CommandCenter cc;
```

```
13
14      @Override
15      public int initialize(GameData gameData, boolean playerNumber)
16      {
17          this.playerNumber = playerNumber;
18          this.inputKey = new Key();
19          cc = new CommandCenter();
20          frameData = new FrameData();
21          return 0;
22      }
23
24
25      @Override
26      public void getInformation(FrameData frameData)
27      {
28          this.frameData = frameData;
29          cc.setFrameData(this.frameData, playerNumber);
30      }
31
32      @Override
33      public void processing() {
34          if(!frameData.getEmptyFlag() &&
                    frameData.getRemainingFramesNumber()> 0) {
35              if(cc.getSkillFlag()) {
36                  inputKey = cc.getSkillKey();
37              } else {
38                  inputKey.empty();
39                  cc.skillCancel();
40                  if(frameData.getDistanceX() < 100) {
41                      cc.commandCall("CROUCH_FB");
42                  } else if (frameData.getDistanceX() > 300) {
43                      cc.commandCall("2 3 6 _ A");
44                  }
45              }
46          }
47      }
48
49      @Override
50      public Key input()
51      {
52          return inputKey;
53      }
54
55      @Override
56      public void close(){
57
58      }
```

```
59
60      @Override
61      public void roundEnd(int p1Hp, int p2Hp, int frames) {
62
63      }
64
65  }
```