全国高校生プログラミングコンテスト

CHaser2009

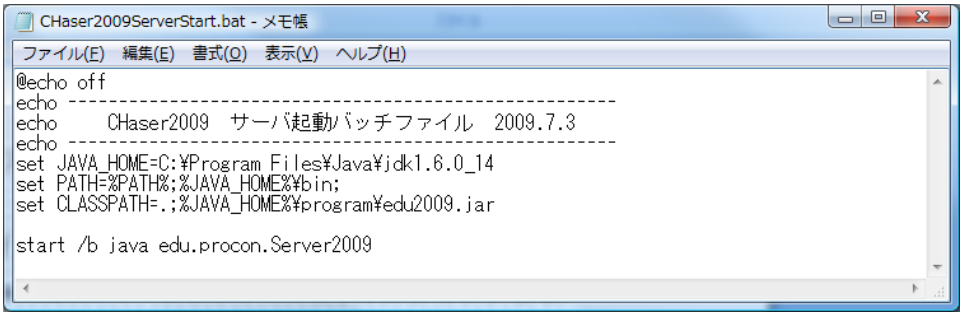
ステップアップヒント３

【競技サーバを起動するバッチファイル】

競技サーバを起動するには、バッチファイル（CHaser2009.bat）をダブルクリックして「java　edu.procon.Server2009」とコマンドを打たなければなりません。

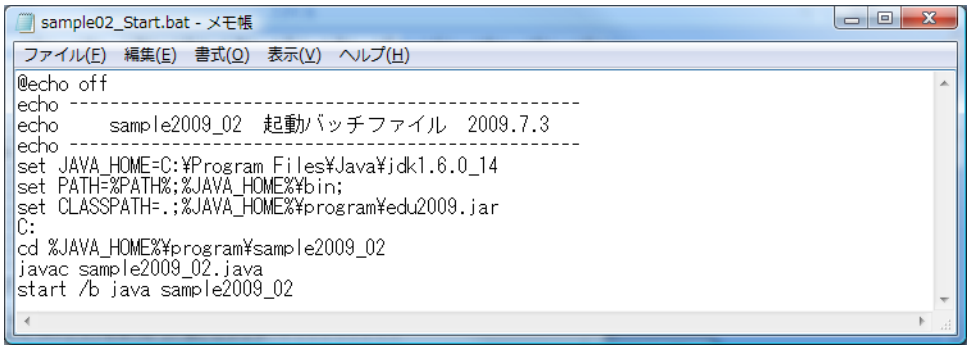
競技をするたびにコマンドを打つのは面倒なので、サーバを起動するバッチファイルを作りましょう。

program フォルダに、CHaser2009ServerStart.bat というファイルを作り、メモ帳などで以下のプログラムを入力します。



【画面 19】競技サーバ起動用のバッチファイル

こうすれば、バッチファイルをダブルクリックするだけで競技サーバが起動します。同様に、対戦プログラムを起動するバッチファイルも作ってみましょう。



【画面 20】サンプルプログラム起動用のバッチファイル

このバッチファイルは、プログラムが保存されているフォルダまで移動して、コンパイルをした後、プログラムを実行します。もし、プログラムにエラーがあれば、コンパイラがエラーメッセージを出力し、バッチファイルを終了します。

「sample2009_02」の部分を変更すれば別のプログラムでも使用できます。

【サンプルプログラム３】アイテム回収処理と相手への対応を追加したプログラム

program フォルダの下に「sample2009_03」という名前のフォルダを作り、次のプログラムを入力します。プログラム名は「sample2009_03.java」です。

【動作】

・相手が上下左右にいた場合、その方向にブロックを put します。
・アイテムが上下左右にあった場合、その方向に walk してアイテムを回収します。
・その他は、sample2009_02.java と同じです。
・old_mode という変数を使用して、１つ前のモード番号を格納します。

【sample2009_03.java】

```
1  public class sample2009_03 {
2      public static void main(String[] args) {
3          int[] value;
4          value = new int[10];
5
6          int mode = 1,      /* 現在のモード */
7              old_mode = 1; /* 前のモード */
8
9          /********** 競技サーバに接続する **********/
10         edu.procon.Connect2009 target;
11         target = new edu.procon.Connect2009("全プロ３");
12
13         while (true) {
14             value = target.getReady();
15             if (value[0] == 0) break;
16
17             /********** 周囲（上下左右）に相手がいるかチェック **********/
18             if(value[2]==1 || value[4]==1 || value[6]==1 || value[8]==1){
19                 mode = 90; /* mode を 90 に変更する */
20             }
21
22             /********** 周囲（上下左右）にアイテムがあるかチェック **********/
23             if(value[2]==3 || value[4]==3 || value[6]==3 || value[8]==3){
24                 old_mode = mode; /* 現在のモードを old_mode に保存する */
25                 mode = 20; /* mode を 20 に変更する */
26             }
27
28             /********** mode で分岐する **********/
29             switch (mode){
30                 case 1: /* 壁にぶつかるまで上に移動する */
31                     if(value[2] == 2){            /* 上にブロックがあったら */
32                         value = target.walkRight(); /* 右に移動し */
33                         mode = 2;                   /* mode を 2 に変更する */
34                     }
35                     else{                   /* そうでなければ、上に移動する */
36                         value = target.walkUp();
37                     }
38                     break;
39                 case 2: /* ブロックにぶつかるまで右に移動する */
40                     if(value[6] == 2){
```

```
41                          value = target.walkDown( );
42                          mode = 3;
43                      }
44                      else{
45                          value = target.walkRight( );
46                      }
47                      break;
48                  case 3: /* ブロックにぶつかるまで下に移動する */
49                      if(value[8] == 2){
50                          value = target.walkLeft( );
51                          mode = 4;
52                      }
53                      else{
54                          value = target.walkDown( );
55                      }
56                      break;
57                  case 4: /* ブロックにぶつかるまで左に移動する */
58                      if(value[4] == 2){
59                          value = target.walkUp( );
60                          mode = 1;
61                      }
62                      else{
63                          value = target.walkLeft( );
64                      }
65                      break;
66                  case 20: /* 周囲にアイテムがあったら、取りに行く */
67                      /* 上にアイテムがあったら、上に walk する */
68                      if(value[2] == 3)       value = target.walkUp();
69                      /* 左にアイテムがあったら、左に walk する */
70                      else if(value[4] == 3) value = target.walkLeft();
71                      /* 右にアイテムがあったら、右に walk する */
72                      else if(value[6] == 3) value = target.walkRight();
73                      /* 下にアイテムがあったら、下に walk する */
74                      else                    value = target.walkDown();
75                      mode = old_mode; /* モードを元に戻す */
76                      break;
77                  case 90: /* 周囲に相手がいたら、put する */
78                      /* 上に相手がいたら、上に put する */
79                      if(value[2] == 1)       value = target.putUp();
80                      /* 左に相手がいたら、左に put する */
81                      else if(value[4] == 1) value = target.putLeft();
82                      /* 右に相手がいたら、右に put する */
83                      else if(value[6] == 1) value = target.putRight();
84                      /* 下に相手がいたら、下に put する */
85                      else                    value = target.putDown();
86                      break;
87
88              }
89              /********** 制御情報が 0 だったら終了する **********/
90              if(value[0] == 0) break;
91          }
92          /********** 競技サーバから切断する **********/
93          target.exit( );
94      }
95  }
```

【プログラム説明】

①変数の宣言

```
6       int mode = 1,      /* 現在のモード */
7           old_mode = 1; /* 前のモード */
```

動作の種類を記憶させる mode を宣言します。old_mode はアイテム回収処理をした後に元のモードに戻るために必要で、1 つ前のモード番号を記憶させておきます。

②チーム名

```
11      target = new edu.procon.Connect2009("全プロ３" );
```

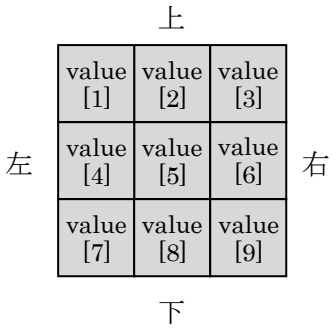
edu.procon.Connect2009 クラスを使って競技サーバに接続します。「サンプルプログラム３」なので、チーム名を「全プロ３」にしました（変更しても構いません）。

③周囲（上下左右）に相手がいるかチェック

```
17      /********** 周囲（上下左右）に相手がいるかチェック **********/
18      if(value[2]==1 || value[4]==1 || value[6]==1 || value[8]==1){
19          mode = 90; /* mode を 90 に変更する */
```

CHaser2009 は、相手にブロックを put して勝利することが一番の目的です。サーバから戻り値で相手は「1」なので、if 文を使って判断します。「value[2]==1」は上に相手がいるか？という意味です。同様に「value[4]==1」は左に相手がいるか？という意味になります。「||」は論理和（OR）をする演算子で、上（value[2]）または左（value[4]）または右（value[6]）または下（value[8]）に相手がいるか？という意味になります。

もし、相手がいたら mode を 90 にします。



【図 6】サーバの戻り値の格納先

④mode=90 の処理

```
77      case 90: /* 周囲に相手がいたら、put する */
78          /* 上に相手がいたら、上に put する */
79          if(value[2] == 1)       value = target.putUp();
80          /* 左に相手がいたら、左に put する */
81          else if(value[4] == 1) value = target.putLeft();
82          /* 右に相手がいたら、右に put する */
83          else if(value[6] == 1) value = target.putRight();
84          /* 下に相手がいたら、下に put する */
85          else                    value = target.putDown();
86          break;
```

switch 文を使って mode の値で分岐処理を行います。mode=90 の場合、上下左右に相手がいるので、相手のいる方向に put し勝利となります。

⑤周囲（上下左右）にアイテムがあるかチェック

```
22      /********** 周囲（上下左右）にアイテムがあるかチェック **********/
23      if(value[2]==3 || value[4]==3 || value[6]==3 || value[8]==3){
24          old_mode = mode; /* 現在のモードを old_mode に保存する */
25          mode = 20; /* mode を 20 に変更する */
26      }
```

相手への処理と同様で if 文を使用して周囲にアイテムがあるか判断します。アイテムは「3」です。アイテムを回収したあと元のモードに戻らなければならないので old_mode に現在のモード番号を格納してから、mode に 20 を格納します。

⑥mode=20 の処理

```
66      case 20: /* 周囲にアイテムがあったら、取りに行く */
67          /* 上にアイテムがあったら、上に walk する */
68          if(value[2] == 3)       value = target.walkUp();
69          /* 左にアイテムがあったら、左に walk する */
70          else if(value[4] == 3) value = target.walkLeft();
71          /* 右にアイテムがあったら、右に walk する */
72          else if(value[6] == 3) value = target.walkRight();
73          /* 下にアイテムがあったら、下に walk する */
74          else                    value = target.walkDown();
75          mode = old_mode; /* モードを元に戻す */
76          break;
```

mode=20 の場合、上下左右にアイテムがあるので、その方向に walk します。アイテムを回収したあと、元のモードに戻らなければならないので old_mode の値を mode に代入します。

現在、使用しているモードは表 4 の通りです。

【表 4】モードの種類と動作

mode	動　作
1	ブロックにぶつかるまで上に移動する
2	ブロックにぶつかるまで右に移動する
3	ブロックにぶつかるまで下に移動する
4	ブロックにぶつかるまで左に移動する
20（新規）	周囲（上下左右）のアイテムを回収する
90（新規）	周囲（上下左右）の相手に put する

【発展】

今回追加したプログラムは上下左右しか対応していないので、斜め方向（右上、右下、左上、左下）に対応できるモードを新たに作らないといけません。