全国高校生プログラミングコンテスト

CHaser2011

ステップアップヒント2

9 競技部品のメソッドについて

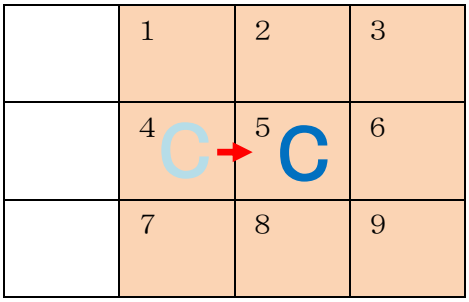
競技部品として用意されているメソッド（命令）について解説します。メソッドを用いてＣ又はＨ（以後プレイヤ）の制御を行います。メソッドには以下のものがあります。

ただし、メソッドの説明においてプレイヤは C とします。また、戻り値が入る変数は整数型の一次元配列 value[10]とします。なお、value[0]は制御情報で１の場合はゲーム続行ですが、０の場合はゲーム終了です。Ａ、Ｂの各メソッドを実行した直後に必ず value[0]の値をチェックし、０の場合はＣの終了メソッドの exit()を実行し、プログラムを終了するようにしてください。

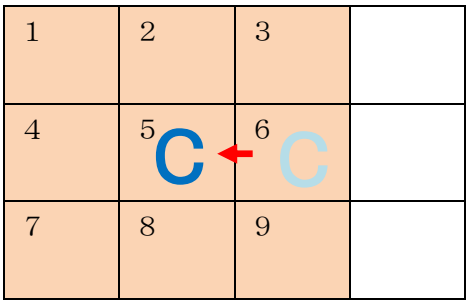
<table>
<tr><td colspan="3">Ａ　準備メソッド（サーバに接続し、プレイヤの周囲情報を得る）</td></tr>
<tr><td>メソッド名</td><td>機能</td><td>戻り値</td></tr>
<tr><td>①getReady()</td><td>サーバに接続し、プレイヤの周囲情報を得る。

<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5
C</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>
マス目にある数字は、value の添え字と一致する。</td><td>制御情報とプレイヤの周囲情報
value[0]=0 又は 1（制御情報）
value[1]=0～3（左図１の情報）
value[2]=0～3（左図２の情報）
value[3]=0～3（左図３の情報）
value[4]=0～3（左図４の情報）
value[5]=0～3（左図５の情報）
value[6]=0～3（左図６の情報）
value[7]=0～3（左図７の情報）
value[8]=0～3（左図８の情報）
value[9]=0～3（左図９の情報）
制御情報
サーバがゲームの状況を判断し
ゲーム続行の場合には 1、
ゲーム終了の場合は 0
を返します。
周囲情報
0:なし（床）
1:相手
2:ブロック
3:アイテム</td></tr>
</table>

<table>
<tr><td colspan="3">B　動作メソッド　walk 系（指定した方向へ１マス移動する）</td></tr>
<tr><td>メソッド名</td><td>機能</td><td>戻り値</td></tr>
<tr><td>①walkUp()
②walkDown()
③walkLeft()
④walkRight()</td><td>指定した方向へ１マス移動する。
walkUp()・・・上
walkDown()・・・下
walkLeft()・・・左
walkRight()・・・右

walkRight()の例
 

マス目にある数字は、value の添え字と一致する。
また、数字の並びは walkUp()、walkDown()、walkLeft()の場合も移動先の周囲 9 マスの左上が 1 となる。

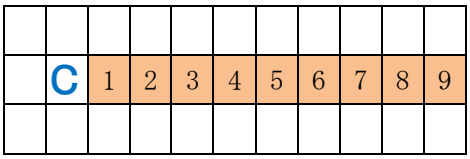
walkLeft()の例
 
</td><td>制御情報とプレイヤの周囲情報
value[0]=0 又は 1（制御情報）
value[1]=0～3（左図１の情報）
value[2]=0～3（左図２の情報）
value[3]=0～3（左図３の情報）
value[4]=0～3（左図４の情報）
value[5]=0～3（左図５の情報）
value[6]=0～3（左図６の情報）
value[7]=0～3（左図７の情報）
value[8]=0～3（左図８の情報）
value[9]=0～3（左図９の情報）
制御情報
サーバがゲームの状況を判断し
ゲーム続行の場合には 1、
ゲーム終了の場合は 0
を返します。
周囲情報
0:なし（床）
1:相手
2:ブロック
3:アイテム</td></tr>
</table>

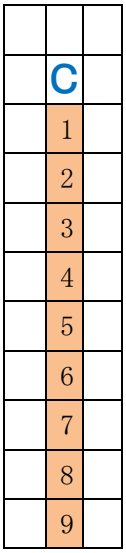
<table>
<tr><td colspan="3">B　動作メソッド　look 系（指定した方向の周囲 9 マスの情報を得る）</td></tr>
<tr><td>メソッド名</td><td>機能</td><td>戻り値</td></tr>
<tr><td>①lookUp()
②lookDown()
③lookLeft()
④lookRight()</td><td>指定した方向の周囲 9 マスの情報を得る。
lookUp()･･･上
lookDown()･･･下
lookLeft()･･･左
lookRight()･･･右

lookRight()の例
<table><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>C</td><td>4</td><td>5</td><td>6</td></tr><tr><td></td><td></td><td>7</td><td>8</td><td>9</td></tr></table>
マス目にある数字は、value の添え字と一致する。
また、数字の並びは lookUp()、lookDown()、lookLeft()の場合も探索範囲の左上が 1 となる。

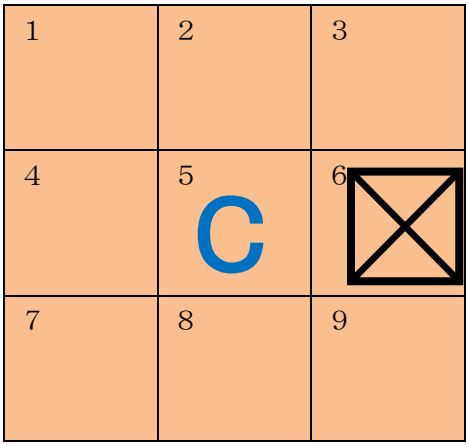
lookDown()の例
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td>C</td><td></td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table></td><td>制御情報とプレイヤの周囲情報
value[0]=0 又は 1（制御情報）
value[1]=0～3（左図 1 の情報）
value[2]=0～3（左図 2 の情報）
value[3]=0～3（左図 3 の情報）
value[4]=0～3（左図 4 の情報）
value[5]=0～3（左図 5 の情報）
value[6]=0～3（左図 6 の情報）
value[7]=0～3（左図 7 の情報）
value[8]=0～3（左図 8 の情報）
value[9]=0～3（左図 9 の情報）
制御情報
サーバがゲームの状況を判断し
ゲーム続行の場合には 1、
ゲーム終了の場合は 0
を返します。
周囲情報
0:なし（床）
1:相手
2:ブロック
3:アイテム</td></tr>
</table>

<table>
<tr><td colspan="3">B　動作メソッド　search 系（指定した方向に真っすぐ 9 マスの情報を得る）</td></tr>
<tr><td>メソッド名</td><td>機能</td><td>戻り値</td></tr>
<tr><td>①searchUp()
②searchDown()
③searchLeft()
④searchRight()</td><td>指定した方向に真っすぐ 9 マスの情報を得る。
searchUp()･･･上
searchDown()･･･下
searchLeft()･･･左
searchRight()･･･右

searchRight()の例
 

マス目にある数字は、value の添え字と一致する。
また、数字の並びはプレイヤに最も近いマス目が 1 となる。
searchDown()の例
 
</td><td>制御情報とプレイヤの周囲情報
value[0]=0 又は 1（制御情報）
value[1]=0～3（左図 1 の情報）
value[2]=0～3（左図 2 の情報）
value[3]=0～3（左図 3 の情報）
value[4]=0～3（左図 4 の情報）
value[5]=0～3（左図 5 の情報）
value[6]=0～3（左図 6 の情報）
value[7]=0～3（左図 7 の情報）
value[8]=0～3（左図 8 の情報）
value[9]=0～3（左図 9 の情報）
制御情報
サーバがゲームの状況を判断し
ゲーム続行の場合には 1、
ゲーム終了の場合は 0
を返します。
周囲情報
0:なし（床）
1:相手
2:ブロック
3:アイテム</td></tr>
</table>

<table>
<tr><td colspan="3">B　動作メソッド　put 系（指定した方向へブロックを置く）</td></tr>
<tr><td>メソッド名</td><td>機能</td><td>戻り値</td></tr>
<tr><td>①putUp()
②putDown()
③putLeft()
④putRight()</td><td>指定した方向へブロックを置く。
putUp()･･･上
putDown()･･･下
putLeft()･･･左
putRight()･･･右

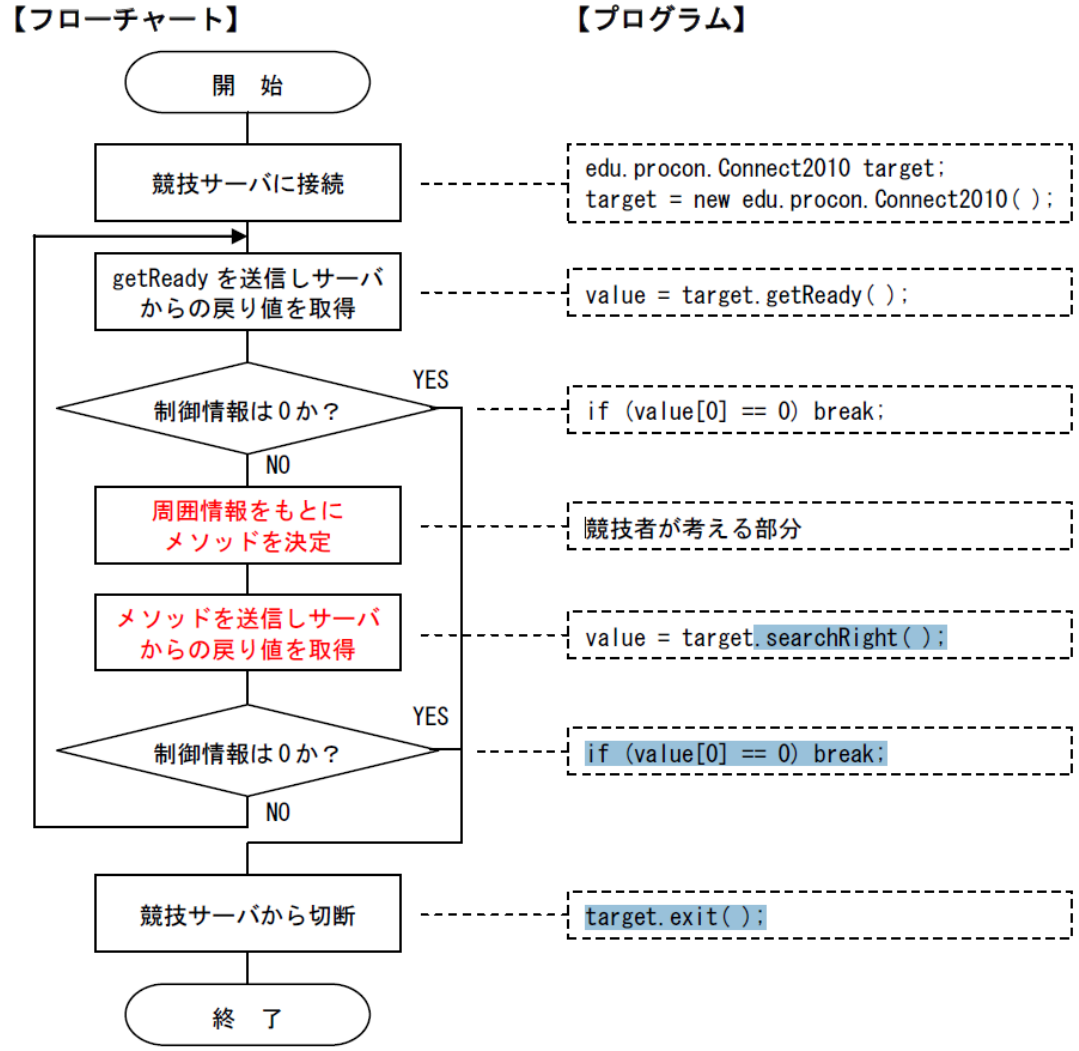
putRight()の例
 

※☒はブロック
マス目にある数字は、value の添え字と一致する。</td><td>制御情報とプレイヤの周囲情報
value[0]=0 又は 1（制御情報）
value[1]=0～3（左図 1 の情報）
value[2]=0～3（左図 2 の情報）
value[3]=0～3（左図 3 の情報）
value[4]=0～3（左図 4 の情報）
value[5]=0～3（左図 5 の情報）
value[6]=0～3（左図 6 の情報）
value[7]=0～3（左図 7 の情報）
value[8]=0～3（左図 8 の情報）
value[9]=0～3（左図 9 の情報）
制御情報
サーバがゲームの状況を判断し
ゲーム続行の場合には 1、
ゲーム終了の場合は 0
を返します。
周囲情報
0:なし（床）
1:相手
2:ブロック
3:アイテム</td></tr>
</table>

C　終了メソッド　（サーバへの接続を切断する）		
メソッド名	機能	戻り値
①exit()	サーバへの接続を切断する。	なし

10 プログラムの流れについて

みなさんが作るクライアントプログラムと競技サーバのやりとりは、準備メソッドの getReady をサーバへ送信して周囲情報を取得した後、動作メソッド(walk 系、look 系、search 系、put 系）を選んで送信し最後に周囲情報を取得という流れを繰り返します。
従って、クライアントプログラムは図 1 のフローチャートのようになります。
みなさんは、サーバから得た周囲情報をもとにメソッドを選んで送信する部分を考えます。



【図 1】プログラムの大まかな流れ

11 サンプルプログラム２（sample2011_02.java：モードを使ったプログラム）

このプログラムは左上からスタートするクライアント専用です。sample01.map では H が左上からスタートしますので、sample2011_02.java は H の制御をすることになります。

①プログラムの保存先

program フォルダの下に「sample2011_02」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「sample2011_02.java」です。

②プログラムの動作

・左上からスタートするクライアント専用のプログラムです。
・まずブロックの直前まで下に移動します。
・次にブロックの直前まで右に移動します。以降はブロックの内側を反時計回り（上、左、下、右）に制限ターンになるまで移動を繰り返します。
・mode（モード）という変数を作り、動作の種類を記憶させています。

③対戦（1 台のパソコンでサーバと 2 つのクライアントを起動する場合）

・CHaser2011.bat を 3 つ起動させる。
・3 つの画面でそれぞれのコマンドを入力し、対戦させる。

サーバ → java edu.procon.Server2010 -Fsample01.map
Cool →java sample2011_01 2009
Hot →java sample2011_02 localhost 2010

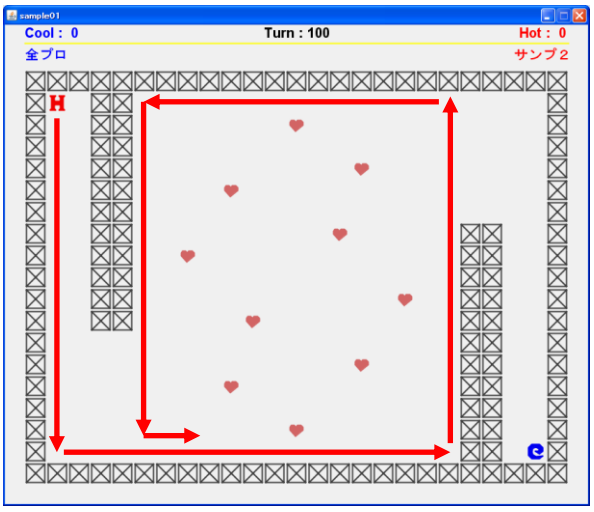
} コマンドライン引数が違うことに注意

java sample2011_02 では、ネットワークを介してサーバのパソコンに接続する場合は以下のように実行します。

java sample2011_02 サーバの IP アドレス ポート番号

（例）java sample2011_02 192.168.1.1 2010（サーバの IP アドレスが 192.168.1.1 の場合）

ただし、引数の記述に間違えがあるとエラーとなりますので注意してください。



【図 2】対戦の様子

④サンプルプログラム２（sample2011_02.java）

```
/*****************************************************
sample2011_02.java
*****************************************************/

public class sample2011_02 {
   public static void main(String[] args) {
      int[] value;
      value = new int[10];
      int mode = 1;

      /********** 競技サーバに接続する **********/
      edu.procon.Connect2010 target;
      target = new edu.procon.Connect2010("サンプ２", args[0] ,Integer.parseInt(args[1]));
      while (true) {
         value = target.getReady( );
         if (value[0] == 0) break;

         /********** mode の値で分岐する **********/
         switch (mode) {
            case 1:                              // ブロック（壁）にぶつかるまで下に移動する
            if(value[8] != 2){                   // 下が壁でなければ、
               value = target.walkDown( );       // 下に移動する
            } else {
               value = target.walkRight( );      // 下が壁ならば、右に移動し、
               mode = 2;                         // mode を 2 に変更する
            }
            break;
            case 2:                              // ブロック（壁）にぶつかるまで右に移動する
            if(value[6] != 2){
               value = target.walkRight( );
            } else {
               value = target.walkUp( );
               mode = 3;
            }
            break;
            case 3:                              // ブロック（壁）にぶつかるまで上に移動する
            if(value[2] != 2){
               value = target.walkUp( );
            } else {
               value = target.walkLeft( );
               mode = 4;
            }
            break;
            case 4:                              // ブロック（壁）にぶつかるまで左に移動する
            if(value[4] != 2){
               value = target.walkLeft( );
            } else {
               value = target.walkDown( );
               mode = 1;
            }
            break;
         }

         /********** 制御情報が 0 だったら終了する **********/
         if(value[0] == 0) break;
      }

      /********** 競技サーバから切断する **********/
```

```
59          target.exit();
60      }
61  }
```

⑤プログラム説明

・変数の宣言

```
7       int[] value;
8       value = new int[10];
9       int mode = 1;
```

value は、サーバからの戻り値（制御情報 1＋周囲情報 9）を格納する配列です。
mode は、動作の種類を記憶させる変数です。mode を使うことで、どの動作をしているのかを管理することができます。サンプルプログラム 2 では表 3 のように 4 つのモードを用意しています。この方式ならば、今後プログラムを発展させるときモードを追加するだけで様々な動作ができるようになります。

【表 3】モードの種類と動作

mode	動作
1	ブロック（壁）にぶつかるまで下に移動する
2	ブロック（壁）にぶつかるまで右に移動する
3	ブロック（壁）にぶつかるまで上に移動する
4	ブロック（壁）にぶつかるまで左に移動する

・競技サーバへの接続

```
11      /********** 競技サーバに接続する **********/
12      edu.procon.Connect2010 target;
13      target = new edu.procon.Connect2010("サンプ２", args[0] ,Integer.parseInt(args[1]));
```

edu.procon.Connect2010 クラスを使って競技サーバに接続します。Connect2010 の第 1 引数の文字列”サンプ２”はサーバに送信するチーム名で競技画面に表示されます。文字列は全角と半角の区別はなく 4 文字以内です。半角は全角に変換されます。また、5 文字以上の場合、先頭の 4 文字が有効です。
第 2 引数 args[0]はサーバの IP アドレスを指定します。第 3 引数は Integer.parseInt(args[0])はプログラムの起動時に指定するポート番号です。

・getReady でサーバからの戻り値を得る

```
15          value = target.getReady( );
16          if (value[0] == 0) break;
```

getReady メソッドを送信するとサーバから戻り値が送信されてくるので、配列 value で受け取ります。value[0]に制御情報が格納されているので、0 だったら break し、while ループを抜けてプログラムを終了します。

・mode で分岐する

```
18          /********** mode の値で分岐する  **********/
19          switch (mode) {
20             case 1:                          // ブロック（壁）にぶつかるまで下に移動する
21             if(value[8] != 2){               // 下が壁でなければ、
22             value = target.walkDown( );      // 下に移動する
23          } else {
24             value = target.walkRight( );     // 下が壁ならば、右に移動し、
25             mode = 2;                        // mode を 2 に変更する
26          }
27          break;
28          case 2:                             // ブロック（壁）にぶつかるまで右に移動する
29              ・
30              ・
31              ・
```

switch 文を使って mode の値で分岐を行います。mode = 1 の場合、ブロック（壁）にぶつかるまで下に移動するので、自分のすぐ下にブロックがあるかを if(value[8] == 2)で判断します。
もし、下がブロックだった場合は次に右に移動するので walkRight メソッドを送信し、mode を 2 にします。ブロックでなければ、walkDown メソッドを送信し下に移動します。同様に mode2 から 4 を作ります。

・制御情報が 0 だったら終了する

```
54          /**********  制御情報が  0  だったら終了する  **********/
55          if(value[0] == 0) break;
```

メソッドを送信するとサーバから戻り値が返ってくるので、終了の判定をします。

・競技サーバから切断する

```
58          /**********  競技サーバから切断する  **********/
59          target.exit( );
```

exit メソッドで競技サーバから切断します。
⑥注意
このプログラムは、左上からスタートするクライアント専用です。左下からスタートするとすぐに右のブロックにぶつかってしまいゲームオーバーとなります。また、モードが切り替わる際、移動する方向にブロックがあるかのチェックをしていないので、ブロックにぶつかってしまう場合があります。

これを回避するためにプログラム中で判断を追加しなければなりません。どこにどのような判断を追加したらよいか考えてみてください。