Programming IV 最終課題 レポート

1 プログラム概要

簡易版テトリス
テトリスの基本ルールに従った、TCP/IP で対戦可能なパズルゲームを作成した。
参考:https://ja.wikipedia.org/wiki/%E3%83%86%E3%83%88%E3%83%AA%E3%82%B9

2 プログラム構成

2.1 クラス概要

Java 言語で、Swing を GUI ツールキットとして使用した。プログラムは以下の図に示すようにクラス化して作成した。

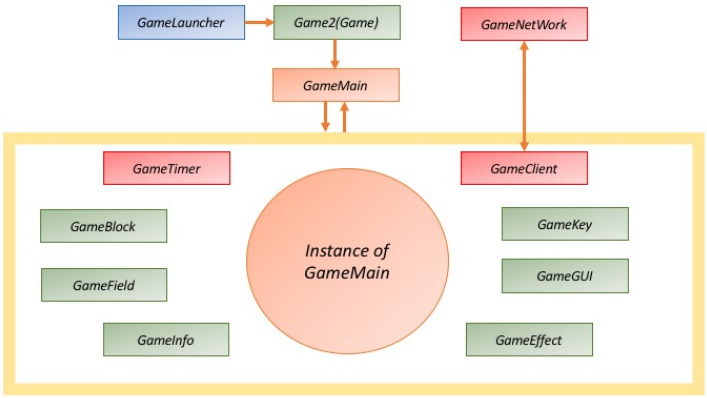

図 1: クラスの構成

2.1.1 GameLauncher クラス

ゲームを立ち上げるためのクラスとして構成した。処理内容は Game2(Game) オブジェクトを生成し、Game オブジェクトの start メソッドを実行する。ソースコード（GameLauncher.java）は後に記述する。

2.1.2 Game クラス

ゲーム全体を取りまとめるクラスとして構成した。処理内容は GameClient、GameMain オブジェクトを生成し、GameClient オブジェクトの setSocket メソッドにより”localhost”をアドレスと設定し、ソケットを利用して接続を行う。run メソッドのスレッド化に Runnable を implements する。ソースコード（Game.java）は後に記述する。

2.1.3 Game2 クラス

Game クラスの改良版。ゲーム全体を取りまとめるクラスとして構成した。GameClient、GameMain オブジェクトを生成する。また、接続先のアドレスをユーザーに指定をさせるための GUI を表示する。TextField にアドレスが入力され、Button が押下された場合、TextField が空である場合は、GameMain オブジェクトを引数なしで生成し、個人用の処理を行う。TextField が入力されている場合、GameClient オブジェクトの setSocket メソッドにより入力された文字列をアドレスと設定する。ソケットを利用して入力されたアドレスに接続ができた場合は、この GameClient オブジェクトを引数として GameMain オブジェクトを生成し、対戦用の処理を行う。ソケットを利用して入力されたアドレスに接続ができない場合は、これらを繰り返す。run メソッドのスレッド化に Runnable を implements する。ソースコード（Game2.java）は後に記述する。

2.1.4 GameMain クラス

ゲームの全てのデータの情報を管理、または処理するクラスとして構成した。ゲーム本体に使用する、GameTimer、GameKey、GameGUI、GameField（自分、相手、仮想処理用）、GameInfo（自分、相手）、GameBlock、GameEffect オブジェクトを生成する。Game2(Game) クラスで、GameClient オブジェクトを指定された際にはこれを用いて通信を行う。ソースコード（GameMain.java）は後に記述する。

2.1.5 GameGUI クラス

ゲーム画面の描画を行うクラスとして構成した。生成には GameKey オブジェクトを必要とし、Frame にキーリスナーを登録する。後に記述するような GUI コンポーネントを構成する。FieldDraw メソッドで GameField の Field の状況を描画し、inforDraw メソッドで GameInfo の描画を行う。ソースコード（GameGUI.java）は後に記述する。

2.1.6 GameKey クラス

KeyListener をインターフェースとして利用し、キーボードの入力の検知を行うクラスとして構成した。ソースコード（GameKey.java）は後に記述する。

2.1.7 GameBlock クラス

テトリスのミノに関する情報を処理するクラスとして構成した。インスタンス変数 x, y を座標とし、 blockType をブロックの種類、block をブロックの形として、処理を行う。newBlock メソッドで、ブロックを生成し、moveUP メソッドから turnLeft メソッドでブロックの移動や回転に関する処理を行う。また、動かせる状態に Permission を追加し、movePermissionUpdate メソッドでこれらを更新する。ソースコード（GameBlock.java）は後に記述する。

2.1.8 GameEffect クラス

Midi チャンネルを利用し、効果音を発生させるクラスとして構成した。ソースコード（GameEffect.java）は後に記述する。

2.1.9 GameField クラス

テトリスのフィールドに関する情報を処理するクラスとして構成した。フィールドの生成、ブロックの設置、消去、消去ラインのチェック、Hindrance 等の処理を行う。また、GameField オブジェクトを通信で使用する構成にしたため、Serializable をインターフェースとして利用し、通信時シリアライズを行なっている。ソースコード（GameField.java）は後に記述する。

2.1.10 GameInfo クラス

ゲームに関する情報をもつクラスとして構成した。Ren 数、DeleteLine 数、Hindrance 数、得点をインスタンス変数として持ち、GUI において Information パネルに表示する際に使用する。ソースコード（GameInfo.java）は後に記述する。

2.1.11 GameTimer クラス

Thread クラスを継承し、時間を管理するクラスとして構成した。システムの Unix Time を基準とし、GetTime メソッドは、TimerStart メソッド、もしくは TimerReset メソッドが実行されてからどれだけ時間が経過したかを値として返す。ソースコード（GameTimer.java）は後に記述する。

2.1.12 GameClient クラス

サーバへの接続、情報の送信を行うクラスとして構成した。setSoket メソッドで、指定した ipAddress で接続を行い、また、Thread クラスを継承し、サーバとの送受信はスレッド化させている。ソースコード（GameClient.java）は後に記述する。

2.1.13 GameNetwork クラス

対戦時、情報の共有等に関する処理をするクラスとして構成した。サーバーとしての役割を行う。ソースコード（GameNetwork.java）は後に記述する。

2.2 GUI 概要

以下の図のように GUI コンポーネントを構成した。

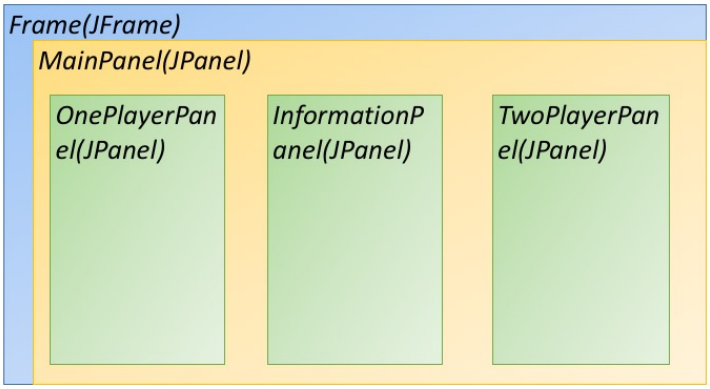

図 2: GUI コンポーネントの構成

- Frame

  メインフレームとして利用する。

- MainPanel

  メインパネルとして利用する。

- OnePlayPanel

  自分のフィールドを描画する。

- TwoPlayPanel

  相手のフィールドを描画する。

- InformationPanel

  Ren 数、Delete Line 数、おじゃまの段数、得点を表示する。

2.3 実行例

一人でおこう場合は、GameLancher クラスでゲームを立ち上げ、IP Address の入力を待つ GUI のテキストフィールドに空の文字列をいれて OK を押す。また、対戦時は、どちらかが GameNetwork クラスを立ち上げ、表示される IP Address を IP Address の入力を待つ GUI のテキストフィールドに入力し OK を押すことで、スタートまたはスタート待ちとなる。キー入力に関しては、z で左回転、x で右回転、上方向キーで HardDrop、下左右方向キーで移動、また、スペースキーで自殺となっている。以下の図は、実行画面のスクリーンショットである。

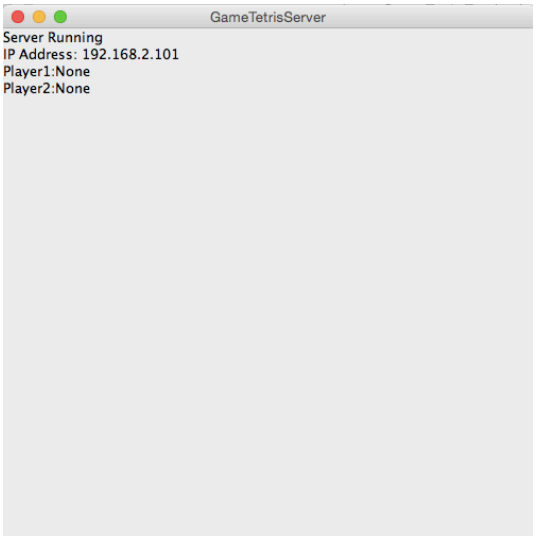

図 3: サーバー起動時

図 4: ゲーム起動時（IP Address 受付）

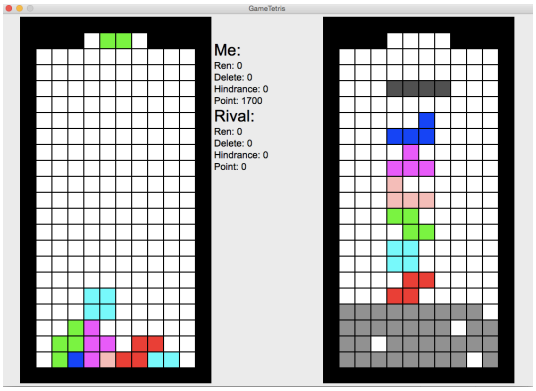

図 5: ゲーム画面

2.4 考察・結論

今回、TCP/IP によって対戦可能なテトリスを作成したが、課題として、同期の問題がある。GameField クラスの sendCheck メソッドを同期に用いているが、完全な同期を行うことができていないため、Unix タイム等を利用するべきであったと思われる。また、原因が特定できていないが、環境によって、MIDI の途中中断、Socket の Broken Pipe などの問題が発生してしまうことがある。これらを修正することが必要であると考えている。

さらに、現在では、壁蹴りや、T スピンというものが存在しているためこれらの処理も追加できれば良いと思っている。そして、本来実装予定である、全体のマジックナンバーを管理する GameSetting クラスを実装できなかったので、これの実装も行いたい。

参考文献

[1] ”ほむほむぺーじ，Java で Tetris，http://www.geocities.jp/h_o_m_2/java/，2016.2.5 参照

[2] ”M. Kom，Java Object の Socket での送受信，http://www.sys9.org/java32.php，2016.2.5 参照

[3] K. Sierra，B，Bates，”Head First Java 第 2 版，2006.3.27

ソースコード

ソースコード 1: GameLauncher.java

```
1  package gametetris;
2
3
4  public class GameLauncher {
5          public static void main(String[] args) {
6                  Game2 game = new Game2();
7                  game.start(); //ゲームの立ち上げ
8          }
9  }
```

ソースコード 2: Game.java

```
1  package gametetris;
2
3  import java.awt.BorderLayout;
4  import java.awt.Container;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.io.*;
8
9  import javax.swing.JButton;
10 import javax.swing.JFrame;
11 import javax.swing.JLabel;
12 import javax.swing.JTextField;
13
14 public class Game implements Runnable{ //
       run メソッドのスレッド化に Runnable を implements する
15         private GameClient client;
16         private GameMain game;
17
18         public Game(){
19                 client = new GameClient();
20                 client.setSocket("localhost"); //"localhost"でソケットに接続
21                 game = new GameMain(client); //ソケット接続が完了した
                       client を引数として GameMain を生成
22         }
```

```
23
24          public void start(){
25                  run(); //スタート
26          }
27
28          public void run() {
29                  try{ //エラーの監視
30                          while(game.run()){} //ゲーム実行
31                  }catch(Exception e){
32                          e.printStackTrace();
33                  }
34          }
35  }
```

ソースコード 3: Game2.java

```
1   package gametetris;
2
3   import java.awt.BorderLayout;
4   import java.awt.Container;
5   import java.awt.event.ActionEvent;
6   import java.awt.event.ActionListener;
7   import java.io.*;
8
9   import javax.swing.JButton;
10  import javax.swing.JFrame;
11  import javax.swing.JLabel;
12  import javax.swing.JTextField;
13
14  public class Game2 implements Runnable, ActionListener{
15          private GameClient client;
16          private GameMain game;
17          private String ip;
18          private JFrame frame;
19          private Container contentPane;
20          private JLabel label;
21          private JLabel label2;
22          private JTextField ip_text;
23          private JButton button;
24          private boolean startFlag;
25
26          public Game2(){
27                  client = new GameClient();
28                  frame = new JFrame("IP␣Address"); //以下、
                        IP Address取得のための GUIの生成
29                  frame.setBounds(30, 30, 500, 80);
30                  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31                  contentPane = frame.getContentPane();
32                  label = new JLabel("IP␣Address:␣");
33                  contentPane.add(label, BorderLayout.WEST);
34                  label2 = new JLabel("Input␣Server␣IP␣Address");
35                  contentPane.add(label2, BorderLayout.SOUTH);
36                  ip_text = new JTextField(10);
37                  contentPane.add(ip_text, BorderLayout.CENTER);
38                  button = new JButton("OK");
39                  button.addActionListener(this);
40                  contentPane.add(button, BorderLayout.EAST);
41                  frame.setVisible(true); //表示
42                  startFlag = false;
43          }
44
```

```
45          public void start(){
46                  while(!startFlag){ //Client の準備完了を待つ.
47                          try {
48                                  Thread.sleep(50);
49                          } catch (InterruptedException e) {
50                                  e.printStackTrace();
51                          }
52                  }
53                  label2.setText("OK");
54                  frame.setVisible(false);
55                  run(); //ゲームの開始
56          }
57
58
59          public void run() {
60                  try{
61                          while(game.run()){}
62                  }catch(Exception e){
63                          e.printStackTrace();
64                  }
65          }
66
67          public void actionPerformed(ActionEvent arg0) { //Button が押下された際
68                  ip = ip_text.getText();
69                  System.out.println(ip);
70                  if(ip.isEmpty()){ //TextField が空
71                          client = null;
72                          game = new GameMain();
73                          startFlag = true;
74                  }
75                  else if(client.setSocket(ip)){ //ソケット接続に成功
76                          game = new GameMain(client);
77                          startFlag = true;
78                  }else{ //ソケット接続に失敗
79                          label2.setText("Connection␣refused:␣Input␣Server␣IP␣
                                Address␣again");
80                  }
81          }
82  }
```

ソースコード 4: GameMain.java

```
1   package gametetris;
2
3   public class GameMain {
4           private GameTimer timer;
5           private GameKey key;
6           private GameGUI game;
7           private GameField field1;
8           private GameField field2;
9           private GameInfo info1;
10          private GameInfo info2;
11          private GameField dummy;
12          private GameBlock block1;
13          private GameClient client;
14          private GameEffect effect;
15          private int waitingTime = 500; //落下タイム
16          private int sleepTime = 180; //キー押下時の WatingTime
17
18          public GameMain(GameClient c) { //ソケット接続時
19                  timer = new GameTimer();
```

```
20              key = new GameKey();
21              game = new GameGUI(key);
22              effect = new GameEffect();
23              field1 = new GameField();
24              field2 = new GameField();
25              info1 = new GameInfo();
26              info2 = new GameInfo();
27              block1 = new GameBlock();
28              client = c;
29
30              clientSet();
31
32              timer.start();
33              timer.timerStart(); //タイマーをスタート
34
35              block1.newBlock(); //新規ブロック生成
36              field1.set_BlockField(block1); //Field にブロックを設置
37
38              game.show(); //画面表示
39              game.fieldDraw(0, field1.get_AllField());
40              game.fieldDraw(1, field2.get_AllField());
41              info1.setInfo(field1);
42              info2.setInfo(field2);
43              game.inforDraw(0, info1);
44              game.inforDraw(1, info2);
45              game.display();
46      }
47
48      public GameMain() { //ソケット未接続時
49              timer = new GameTimer();
50              key = new GameKey();
51              game = new GameGUI(key);
52              effect = new GameEffect();
53              field1 = new GameField();
54              field2 = new GameField();
55              info1 = new GameInfo();
56              info2 = new GameInfo();
57              block1 = new GameBlock();
58              client = null;
59
60              timer.start();
61              timer.timerStart(); //タイマーをスタート
62
63              block1.newBlock(); //新規ブロック生成
64              field1.set_BlockField(block1); //Field にブロックを設置
65
66              game.show(); //画面表示
67              game.fieldDraw(0, field1.get_AllField());
68              game.fieldDraw(1, field2.get_AllField());
69              info1.setInfo(field1);
70              info2.setInfo(field2);
71              game.inforDraw(0, info1);
72              game.inforDraw(1, info2);
73              game.display();
74      }
75
76      public void clientSet(){
77              field1.sendCheck(); //送信チェック付加
78              client.sendGameField(field1); //自分の Field データ送信
79
80              System.out.println("Wainting␣Other␣Player");
```

```
81              while(true){ //受信チェック
82                      dummy = client.fetchGameField();
83                      if(dummy != null){
84                              field1.returnCheck(dummy);
85                              client.sendGameField(field1);
86                              if(dummy.check(field1)){
87                                      break;
88                              }
89                      }
90                      try {
91                              Thread.sleep(50);
92                      } catch (InterruptedException e) {
93                              e.printStackTrace();
94                      }
95              }
96              System.out.println("Start");
97              field2 = dummy;
98      }
99
100     public boolean run(){ //ゲームメイン処理
101             if(client != null){
102                     field2 = client.fetchGameField(); //相手の Field データを取得
103                     if(field2.getEnd() || field2 == null){ //相手が終了した際
104                             info2.setInfo(field2);
105                             game.inforDraw(1, info2);
106                             game.fieldDraw(1, field2.get_AllField());
107                             game.winDraw(true);
108                             game.display();
109                             effect.win();
110                             client.close(); //ソケットを閉じる
111                             effect.close(); //MIDI を閉じる
112                             return false;
113                     }
114                     if(!field1.check(field2)){ //受信チェック
115                             dummy = field2;
116                             field1.returnCheck(field2);
117                     }
118                     if(dummy.hindPermisson()){ //受信チェックで重複した場合は
                            false
119                             field1.addHindNum(dummy.getDleteNumber() + (
                                    dummy.getRen() - 1)); //Hindrance を追加
120                             dummy.setHindFrag(false);
121                     }
122                     client.sendGameField(field1); //自分の Field を送信
123             }
124
125             field1.unset_BlockField(block1); //Field から Current Block を消去
126
127             if(key.get_moveUP()){ //上を押下
128                     block1.movePermissionUpdate(field1);
129                     block1.moveUP(field1);
130                     effect.hardDrop();
131                     try {
132                             Thread.sleep(sleepTime);
133                     } catch (InterruptedException e) {
134                             e.printStackTrace();
135                     }
136                     effect.stop();
137             }
138
139             if(key.get_moveDown()){ //下を押下
```

```
140                     block1.movePermissionUpdate(field1);
141                     block1.moveDown();
142                     effect.moveDown();
143                     try {
144                             Thread.sleep(sleepTime);
145                     } catch (InterruptedException e) {
146                             e.printStackTrace();
147                     }
148                     effect.stop();
149             }
150
151             if(key.get_moveLeft()){ //左を押下
152                     block1.movePermissionUpdate(field1);
153                     block1.moveLeft();
154                     effect.moveLeft();
155                     try {
156                             Thread.sleep(sleepTime);
157                     } catch (InterruptedException e) {
158                             e.printStackTrace();
159                     }
160                     effect.stop();
161             }
162
163             if(key.get_moveRight()){ //右を押下
164                     block1.movePermissionUpdate(field1);
165                     block1.moveRight();
166                     effect.moveRight();
167                     try {
168                             Thread.sleep(sleepTime);
169                     } catch (InterruptedException e) {
170                             e.printStackTrace();
171                     }
172                     effect.stop();
173             }
174
175             if(key.get_turnLeft()){ //z を押下
176                     block1.movePermissionUpdate(field1);
177                     block1.turnLeft();
178                     effect.turnLeft();
179                     try {
180                             Thread.sleep(sleepTime);
181                     } catch (InterruptedException e) {
182                             e.printStackTrace();
183                     }
184                     effect.stop();
185             }
186
187             if(key.get_turnRight()){ //z を押下
188                     block1.movePermissionUpdate(field1);
189                     block1.turnRight();
190                     effect.turnRight();
191                     try {
192                             Thread.sleep(sleepTime);
193                     } catch (InterruptedException e) {
194                             e.printStackTrace();
195                     }
196                     effect.stop();
197             }
198
199             if(key.get_Start()){ //space を押下
```

```
200                     field1.setHindNum(20); //自殺宣告
201                     try {
202                             Thread.sleep(sleepTime);
203                     } catch (InterruptedException e) {
204                             e.printStackTrace();
205                     }
206             }
207
208
209             block1.movePermissionUpdate(field1);
210             if(block1.blockGround()){ //ブロックが接地
211                     field1.set_BlockField(block1); //Block を Field に追加
212                     field1.setDeleteNumber(0); //Delete Line 数を初期化
213                     if(timer.getTime() >= waitingTime || key.get_moveDown() ||
                            block1.get_HardDrop()){
214                             if(field1.checkLine()){ //消去ラインのチェック
215                                     field1.deleteLine(); //消去
216                                     field1.killLine();
217                                     field1.sendCheck(); //送信チェック
218                                     effect.delete(field1.getDleteNumber(), field1.
                                            getRen());
219                                     try {
220                                             Thread.sleep(100);
221                                     } catch (InterruptedException e) {
222                                             e.printStackTrace();
223                                     }
224                                     effect.stop();
225                             }else{
226                                     field1.setRenZero(); //Ren 数の初期化
227                             }
228                             field1.hindrance(); //おじゃま追加
229                             effect.hind(field1.getHind());
230                             try {
231                                     Thread.sleep(100);
232                             } catch (InterruptedException e) {
233                                     e.printStackTrace();
234                             }
235                             field1.setHindZero();
236                             effect.stop();
237                             block1.newBlock(); //新規ブロック生成
238                             if(!block1.newBlockPermission(field1)){ //ゲームオー
                                    バ判定
239                                     game.fieldDraw(0, field1.get_AllField());
240                                     game.fieldDraw(1, field2.get_AllField());
241                                     info1.setInfo(field1);
242                                     info2.setInfo(field2);
243                                     game.inforDraw(0, info1);
244                                     game.inforDraw(1, info2);
245                                     game.display();
246                                     field1.setEnd();
247                                     field1.sendCheck();
248                                     if(client != null){
249                                             client.sendGameField(field1);
250                                             game.winDraw(false);
251                                             game.display();
252                                             effect.lose();
253                                             while(client.fetchGameField() != null
                                                    ){}
254                                             client.close();
255                                             effect.close();
```

```
256                                         }else{
257                                                 effect.lose();
258                                         }
259                                         return false;
260                                 }
261                                 timer.timerReset(); //タイマーリセット
262                         }
263                 }else if(timer.getTime() >= waitingTime){ //自由落下
264                         block1.movePermissionUpdate(field1);
265                         block1.moveDown();
266                         timer.timerReset();
267                 }
268
269                 field1.set_BlockField(block1); //Block を Field に追加
270                 game.fieldDraw(0, field1.get_AllField()); //Field を描画
271                 game.fieldDraw(1, field2.get_AllField());
272                 info1.setInfo(field1);
273                 info2.setInfo(field2);
274                 game.inforDraw(0, info1);
275                 game.inforDraw(1, info2);
276                 game.display();
277                 return true;
278         }
279 }
```

ソースコード 5: GameGUI.java

```
1  package gametetris;
2
3  import java.awt.*;
4
5  import javax.swing.*;
6
7  public class GameGUI {
8
9
10         private JFrame frame;
11         private JPanel mainPane;
12         private JPanel[] playerPane = new JPanel[2];
13         private JPanel infoPane;
14         private Image[] image = new Image[2];
15
16         private JLabel[] player = new JLabel[2];
17         private JLabel[] renNum = new JLabel[2];
18         private JLabel[] deleteNum = new JLabel[2];
19         private JLabel[] hindNum = new JLabel[2];
20         private JLabel[] point = new JLabel[2];
21
22         /* フィールドの設定用 */
23         private int MainW=1000;
24         private int MainH=700;
25         private int FieldW=360;
26         private int FieldH=690;
27         private int InfoW=200;
28         private int InfoH=600;
29         private int WindowW = 1050;
30         private int WindowH = 700;
31         private int WindowX = 0;
32         private int WindowY = 0;
33         private int FieldRows = 23;
34         private int FieldCols = 12;
```

```
35      private int BlockSize = 30;
36
37      private Graphics[] fg = new Graphics[2]; //Field
38      private Graphics[] bg = new Graphics[2]; //Block
39
40      public GameGUI(GameKey key){
41              frame = new JFrame("GameTetris");
42              frame.setBounds(WindowX, WindowY, WindowW, WindowH);
43              frame.setResizable(false);
44              frame.setFocusable(true);
45              frame.requestFocus();
46              frame.addKeyListener(key);
47              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48
49              Container container = frame.getContentPane();
50
51              mainPane = new JPanel();
52              mainPane.setLayout(new FlowLayout());
53              mainPane.setPreferredSize(new Dimension(MainW, MainH));
54
55              playerPane[0] = new JPanel();
56              playerPane[0].setPreferredSize(new Dimension(FieldW, FieldH));
57
58              playerPane[1] = new JPanel();
59              playerPane[1].setPreferredSize(new Dimension(FieldW, FieldH));
60
61              infoPane = new JPanel();
62              infoPane.setPreferredSize(new Dimension(InfoW, InfoH));
63              infoPane.setLayout(new BoxLayout(infoPane, BoxLayout.Y_AXIS));
64
65              player[0] = new JLabel("Me:");
66              renNum[0] = new JLabel("Ren:␣");
67              deleteNum[0] = new JLabel("Delete:␣");
68              hindNum[0] = new JLabel("Hindrance:␣");
69              point[0] = new JLabel("Point:␣");
70              player[1] = new JLabel("Rival:");
71              renNum[1] = new JLabel("Ren:␣");
72              deleteNum[1] = new JLabel("Delete:␣");
73              hindNum[1] = new JLabel("Hindrance:␣");
74              point[0] = new JLabel("Point:␣");
75              point[1] = new JLabel("Point:␣");
76              player[0].setFont(new Font("Arial", Font.PLAIN, 30));
77              player[1].setFont(new Font("Arial", Font.PLAIN, 30));
78              renNum[0].setFont(new Font("Arial", Font.PLAIN, 18));
79              renNum[1].setFont(new Font("Arial", Font.PLAIN, 18));
80              deleteNum[0].setFont(new Font("Arial", Font.PLAIN, 18));
81              deleteNum[1].setFont(new Font("Arial", Font.PLAIN, 18));
82              hindNum[0].setFont(new Font("Arial", Font.PLAIN, 18));
83              hindNum[1].setFont(new Font("Arial", Font.PLAIN, 18));
84              point[0].setFont(new Font("Arial", Font.PLAIN, 18));
85              point[1].setFont(new Font("Arial", Font.PLAIN, 18));
86
87              infoPane.add(player[0]);
88              infoPane.add(renNum[0]);
89              infoPane.add(deleteNum[0]);
90              infoPane.add(hindNum[0]);
91              infoPane.add(point[0]);
92              infoPane.add(player[1]);
93              infoPane.add(renNum[1]);
94              infoPane.add(deleteNum[1]);
95              infoPane.add(hindNum[1]);
```

```
96              infoPane.add(point[1]);
97
98              mainPane.add(playerPane[0]);
99              mainPane.add(infoPane);
100             mainPane.add(playerPane[1]);
101             container.add(mainPane);
102             frame.pack();
103
104             image[0] = playerPane[0].createImage(FieldW, FieldH-1);
105             image[1] = playerPane[1].createImage(FieldW, FieldH-1);
106
107             fg[0] = playerPane[0].getGraphics();
108             bg[0] = image[0].getGraphics();
109             bg[0].fillRect(0, 0, FieldW, FieldH-1);
110             fg[1] = playerPane[1].getGraphics();
111             bg[1] = image[1].getGraphics();
112             bg[1].fillRect(0, 0, FieldW, FieldH-1);
113     }
114
115     public void show(){
116             frame.setVisible(true);
117             frame.requestFocus();
118     }
119
120     public void fieldDraw(int p, int[][] field){
121             for(int row = 1; row <= FieldRows-1; row++){
122                     for(int col = 0; col <= FieldCols-1; col++){
123                             if(field[row][col] == 9){
124                                     bg[p].setColor(Color.black);
125                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
126                             }
127                             else if(field[row][col] == 0){
128                                     bg[p].setColor(Color.white);
129                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
130                             }
131                             else if(field[row][col] == 1){
132                                     bg[p].setColor(Color.magenta);
133                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
134                             }
135                             else if(field[row][col] == 2){
136                                     bg[p].setColor(Color.green);
137                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
138                             }
139                             else if(field[row][col] == 3){
140                                     bg[p].setColor(Color.red);
141                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
142                             }
143                             else if(field[row][col] == 4){
144                                     bg[p].setColor(Color.blue);
145                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
146                             }
147                             else if(field[row][col] == 5){
148                                     bg[p].setColor(Color.pink);
149                                     bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                            +1, BlockSize-2, BlockSize-2);
```

```
150                                 }
151                                 else if(field[row][col] == 6){
152                                         bg[p].setColor(Color.cyan);
153                                         bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                                +1, BlockSize-2, BlockSize-2);
154                                 }
155                                 else if(field[row][col] == 7){
156                                         bg[p].setColor(Color.darkGray);
157                                         bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                                +1, BlockSize-2, BlockSize-2);
158                                 }
159                                 else if(field[row][col] == 8){
160                                         bg[p].setColor(Color.orange);
161                                         bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                                +1, BlockSize-2, BlockSize-2);
162                                 }
163                                 else if(field[row][col] == 10){
164                                         bg[p].setColor(Color.gray);
165                                         bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                                +1, BlockSize-2, BlockSize-2);
166                                 }
167                                 else{
168                                         bg[p].setColor(Color.white);
169                                         bg[p].fillRect(BlockSize*col+1, BlockSize*row
                                                +1, BlockSize-2, BlockSize-2);
170                                 }
171                         }
172                 }
173         }
174
175         public void inforDraw(int p, GameInfo info){
176                 renNum[p].setText("Ren:␣"+info.getRen());
177                 deleteNum[p].setText("Delete:␣"+info.getDelete());
178                 hindNum[p].setText("Hindrance:␣"+info.getHind());
179                 point[p].setText("Point:␣"+info.getPoint());
180         }
181
182         public void winDraw( boolean win){
183                 if(win == true){
184                         player[0].setText("Me:␣Winner");
185                         player[1].setText("Rival:␣Loser");
186                 }else{
187                         player[0].setText("Me:␣Loser");
188                         player[1].setText("Rival:␣Winner");
189                 }
190         }
191
192         public void display(){
193                 for(int p = 0; p < 2; p++){
194                         fg[p].drawImage(image[p], 0, 0, playerPane[p]);
195                 }
196         }
197 }
```

ソースコード 6: GameKey.java

```
1  package gametetris;
2
3  import java.awt.event.*;
4
5  public class GameKey implements KeyListener {
```

```
6
7       public boolean UP;
8       public boolean DOWN;
9       public boolean LEFT;
10      public boolean RIGHT;
11      public boolean TURN_L;
12      public boolean TURN_R;
13      public boolean START;
14
15      private GameTimer timerDown;
16      private GameTimer timerLeft;
17      private GameTimer timerRight;
18
19      GameKey(){
20              timerDown = new GameTimer();
21              timerLeft = new GameTimer();
22              timerRight = new GameTimer();
23              timerDown.start();
24              timerLeft.start();
25              timerRight.start();
26      }
27
28      public boolean get_moveDown(){
29              return DOWN;
30      }
31
32      public boolean get_moveUP(){
33              return UP;
34      }
35
36      public boolean get_moveLeft(){
37              return LEFT;
38      }
39
40      public boolean get_moveRight(){
41              return RIGHT;
42      }
43
44      public boolean get_turnLeft(){
45              return TURN_L;
46      }
47
48      public boolean get_turnRight(){
49              return TURN_R;
50      }
51
52      public boolean get_Start(){
53              return START;
54      }
55
56
57      public void keyTyped(KeyEvent e){
58      }
59
60      public void keyPressed(KeyEvent e){
61              int key_code;
62              key_code = e.getKeyCode();
63
64              if(key_code == KeyEvent.VK_DOWN){
65                      if(!DOWN){
66                              DOWN = true;
```

```
67                              timerDown.timerStart();
68                      }
69                      return;
70              }
71              if(key_code == KeyEvent.VK_UP){
72                      UP = true;
73                      return;
74              }
75              if(key_code == KeyEvent.VK_LEFT){
76                      if(!LEFT){
77                              LEFT = true;
78                              timerLeft.timerStart();
79                      }
80                      return;
81              }
82              if(key_code == KeyEvent.VK_RIGHT){
83                      if(!RIGHT){
84                              RIGHT = true;
85                              timerRight.timerStart();
86                      }
87                      return;
88              }
89              if(key_code == KeyEvent.VK_Z){
90                      TURN_L = true;
91                      return;
92              }
93              if(key_code == KeyEvent.VK_X){
94                      TURN_R = true;
95                      return;
96              }
97              if(key_code == KeyEvent.VK_SPACE){
98                      START = true;
99                      return;
100             }
101     }
102
103     public void keyReleased(KeyEvent e){
104             int key_code;
105             key_code = e.getKeyCode();
106
107             if(key_code == KeyEvent.VK_DOWN){
108                     DOWN = false;
109                     timerDown.timerReset();
110                     timerDown.timerStop();
111                     return;
112             }
113             if(key_code == KeyEvent.VK_UP){
114                     UP = false;
115                     return;
116             }
117             if(key_code == KeyEvent.VK_LEFT){
118                     LEFT = false;
119                     timerLeft.timerReset();
120                     timerLeft.timerStop();
121                     return;
122             }
123             if(key_code == KeyEvent.VK_RIGHT){
124                     RIGHT = false;
125                     timerRight.timerReset();
126                     timerRight.timerStop();
127                     return;
```

```
128                     }
129                     if(key_code == KeyEvent.VK_Z){
130                             TURN_L = false;
131                             return;
132                     }
133                     if(key_code == KeyEvent.VK_X){
134                             TURN_R = false;
135                             return;
136                     }
137                     if(key_code == KeyEvent.VK_SPACE){
138                             START = false;
139                             return;
140                     }
141             }
142     }
```

ソースコード 7: GameBlock.java

```
1   package gametetris;
2
3   import java.util.*;
4
5   public class GameBlock implements Cloneable{
6
7           private int[][] block;
8           private int x;
9           private int y;
10          private int FieldRows= 23;
11          private int FieldCols = 12;
12          private int blockType;
13          private int blockNum = 8;
14          private int[] blockRand = new int[blockNum];
15          private boolean[] blockMovePermission;
16          private boolean hardDrop;
17
18          /**
19           * @param args
20           */
21          public static void main(String[] args) {
22                  GameBlock block = new GameBlock();
23                  for(int i = 0; i < 20; i++){
24                          block.newBlock();
25                  }
26          }
27
28          public GameBlock clone(){
29                  GameBlock object;
30                  try {
31                          int[][] tmp = new int[block.length][block.length];
32                          object = (GameBlock) super.clone();
33                          for(int i = 0; i < block.length; i++){
34                                  for(int j = 0; j < block.length; j++){
35                                          tmp[i][j] = block[i][j];
36                                  }
37                                  object.block = tmp;
38                          }
39                  } catch (CloneNotSupportedException e) {
40                          e.printStackTrace();
41                          return null;
42                  }
43                  return object;
```

```
44          }
45
46          public GameBlock(){
47                  blockMovePermission = new boolean[6];
48                  for(int i = 0; i < 6; i++){
49                          blockMovePermission[i] = false;
50                  }
51                  blockType = 0;
52          }
53
54          private int newBlockType(){
55                  if(blockType > blockNum-2){
56                          blockType = 0;
57                  }
58                  if(blockType == 0){
59                          Random rnd = new Random();
60                          for(int i = 0; i < blockNum-1; i++){
61                                  blockRand[i] = rnd.nextInt(blockNum-1)+1;
62                                  int a = blockRand[i];
63                                  for( i = 0; i < blockNum -1; i++){
64                                          if(blockRand[i] == a){
65                                                  break;
66                                          }
67                                  }
68                          }
69                  }
70                  return blockRand[blockType++];
71          }
72
73          public void newBlock(){
74                  int n = newBlockType();
75                  switch(n){
76                  case 1 :
77                          block = new int[][] { {0, n, 0},
78                                                {n, n, n},
79                                                {0, 0, 0} };
80                          x=FieldCols/2-2;
81                          y=0;
82                          break;
83
84                  case 2 :
85                          block = new int[][] { {n, n, 0},
86                                                {0, n, n},
87                                                {0, 0, 0} };
88                          x=FieldCols/2-2;
89                          y=0;
90                          break;
91
92                  case 3 :
93                          block = new int[][] { {0, n, n},
94                                                {n, n, 0},
95                                                {0, 0, 0} };
96                          x=FieldCols/2-2;
97                          y=0;
98                          break;
99
100                 case 4 :
101                         block = new int[][] { {0, 0, n},
102                                               {n, n, n},
103                                               {0, 0, 0} };
104                         x=FieldCols/2-2;
```

```
105                     y=0;
106                     break;
107
108             case 5 :
109                     block = new int[][] { {n, 0, 0},
110                                           {n, n, n},
111                                           {0, 0, 0} };
112                     x=FieldCols/2−2;
113                     y=0;
114                     break;
115
116             case 6 :
117                     block = new int[][] { {n, n},
118                                           {n, n} };
119                     x=FieldCols/2−2;
120                     y=0;
121                     break;
122
123             case 7 :
124                     block = new int[][] { {0, 0, 0, 0},
125                                           {n, n, n, n},
126                                           {0, 0, 0, 0},
127                                           {0, 0, 0, 0} };
128                     x=FieldCols/2−2;
129                     y=0;
130                     break;
131
132             }
133     }
134
135     public boolean setBlockPermission(GameField field){
136             movePermissionUpdate(field);
137             if(!blockMovePermission[5]){
138                     return false;
139             }
140             return true;
141     }
142
143     public boolean newBlockPermission(GameField field){
144             movePermissionUpdate(field);
145             if(y == 0 && !blockMovePermission[5]){
146                     return false;
147             }
148             return true;
149     }
150
151     public void moveUP(GameField field){
152             while(blockMovePermission[0]){
153                     y++;
154                     movePermissionUpdate(field);
155             }
156             hardDrop = true;
157     }
158
159     public boolean get_HardDrop(){
160             if(hardDrop){
161                     hardDrop = false;
162                     return true;
163             }
164             return false;
165     }
```

```
166
167         //下移動
168         public void moveDown(){
169                 if(blockMovePermission[0]){
170                         y++;
171                 }
172         }
173         //下移動（ダミーのみ使用可能）
174         private void moveDown(boolean permission){
175                 if(permission){
176                         y++;
177                 }
178         }
179
180         //右移動
181         public void moveRight(){
182                 if(blockMovePermission[1]){
183                         x++;
184                 }
185         }
186
187         //右移動（ダミーのみ使用可能）
188         private void moveRight(boolean permisson){
189                 if(permisson){
190                         x++;
191                 }
192         }
193
194         //左移動
195         public void moveLeft(){
196                 if(blockMovePermission[2]){
197                         x--;
198                 }
199         }
200
201         //左移動（ダミーのみ使用可能）
202         private void moveLeft(boolean permisson){
203                 if(permisson){
204                         x--;
205                 }
206         }
207
208         //右回転
209         public void turnRight(){
210                 if(blockMovePermission[3]){
211                         int[][] tmp = new int[block.length][block.length];
212
213                         for(int row = 0; row < block.length; row++){
214                                 for(int col = 0; col < block.length; col++){
215                                         tmp[row][col] = block[row][col];
216                                 }
217                         }
218
219                         for(int row = 0; row < block.length; row++){
220                                 for(int col = 0; col < block.length; col++){
221                                         block[col][block.length - 1 - row] = tmp[row][
                                                col];
222                                 }
223                         }
224                 }
225         }
```

```
226
227         //右回転（ダミーのみ使用可能）
228         private void turnRight(boolean permission){
229                 if(permission){
230                         int[][] tmp = new int[block.length][block.length];
231
232                         for(int row = 0; row < block.length; row++){
233                                 for(int col = 0; col < block.length; col++){
234                                         tmp[row][col] = block[row][col];
235                                 }
236                         }
237
238                         for(int row = 0; row < block.length; row++){
239                                 for(int col = 0; col < block.length; col++){
240                                         block[col][block.length - 1 - row] = tmp[row][
                                                col];
241                                 }
242                         }
243                 }
244         }
245
246         //左回転
247         public void turnLeft(){
248                 if(blockMovePermission[4]){
249                         int[][] tmp = new int[block.length][block.length];
250
251                         for(int row = 0; row < block.length; row++){
252                                 for(int col = 0; col < block.length; col++){
253                                         tmp[row][col] = block[row][col];
254                                 }
255                         }
256
257                         for(int row = 0; row < block.length; row++){
258                                 for(int col = 0; col < block.length; col++){
259                                         block[block.length - 1 - col][row] = tmp[row][
                                                col];
260                                 }
261                         }
262                 }
263         }
264
265         //左回転（ダミーのみ使用可能）
266         private void turnLeft(boolean permission){
267                 if(permission){
268                         int[][] tmp = new int[block.length][block.length];
269
270                         for(int row = 0; row < block.length; row++){
271                                 for(int col = 0; col < block.length; col++){
272                                         tmp[row][col] = block[row][col];
273                                 }
274                         }
275
276                         for(int row = 0; row < block.length; row++){
277                                 for(int col = 0; col < block.length; col++){
278                                         block[block.length - 1 - col][row] = tmp[row][
                                                col];
279                                 }
280                         }
281                 }
282         }
283
```

```
284         public void movePermissionUpdate(GameField field){
285                 GameBlock[] dummy = new GameBlock[6];
286                 for(int i = 0; i < 6; i++){
287                         dummy[i] = clone();
288                 }
289                 dummy[0].moveDown(true);
290                 dummy[1].moveRight(true);
291                 dummy[2].moveLeft(true);
292                 dummy[3].turnRight(true);
293                 dummy[4].turnLeft(true);
294
295                 for(int i = 0; i < 6; i++){
296                         blockMovePermission[i] = true;
297                         for(int row=0; row < dummy[i].get_BlockSize(); row++){
298                                 if(dummy[i].get_Y() + row < 0)continue;
299                                 if(dummy[i].get_Y() + row > FieldRows-1)break;
300                                 for(int col=0; col < dummy[i].get_BlockSize(); col
                                        ++){
301                                     if(dummy[i].get_X() + col < 0)continue;
302                                     if(dummy[i].get_X() + col > FieldCols-1)
                                            break;
303                                     if(dummy[i].get_Block(row, col) != 0 && field.
                                            get_Field(dummy[i].get_Y() + row, dummy
                                            [i].get_X() + col) != 0){
304                                         blockMovePermission[i] = false;
305                                         break;
306                                     }
307                                 }
308                                 if(!blockMovePermission[i])break;
309                         }
310                 }
311         }
312
313         public boolean blockGround(){
314                 return !blockMovePermission[0];
315         }
316
317         //blockをクリア
318         public void clear(){
319                 block = new int[0][0];
320         }
321
322         public int[][] get_AllBlock(){
323                 return block;
324         }
325
326         public int get_Block(int row, int col){
327                 return block[row][col];
328         }
329
330         public int get_BlockSize(){
331                 return block.length;
332         }
333
334         public int get_X(){
335                 return x;
336         }
337
338         public int get_Y(){
339                 return y;
340         }
```

```
341
342         public void debug(){
343                 System.out.println("y="+y+",x="+x);
344                 for(int i = 0; i < block.length; i++){
345                         for(int j = 0; j < block.length; j++){
346                                 System.out.print(block[i][j]);
347                         }
348                         System.out.println();
349                 }
350         }
351 }
```

ソースコード 8: GameEffect.java

```
1  package gametetris;
2
3  import javax.sound.midi.Instrument;
4  import javax.sound.midi.MidiChannel;
5  import javax.sound.midi.MidiDevice;
6  import javax.sound.midi.MidiSystem;
7  import javax.sound.midi.MidiUnavailableException;
8  import javax.sound.midi.Soundbank;
9  import javax.sound.midi.Synthesizer;
10
11 public class GameEffect {
12         Synthesizer synth;
13         MidiChannel channel;
14         private MidiDevice synthesizer;
15         public GameEffect() {
16                 try {
17                         synth= MidiSystem.getSynthesizer();
18
19                         synth.open();
20                         Instrument[] instruments = synth.getDefaultSoundbank().
                               getInstruments();
21                         synth.loadInstrument(instruments[0]);
22
23                         channel = synth.getChannels()[10];
24                 } catch(Exception e){
25                         e.printStackTrace();
26                         if(channel != null) channel.allNotesOff();
27                 }
28         }
29
30         public void close(){
31                 synth.close();
32         }
33
34         public void stop(){
35                 channel.allNotesOff();
36         }
37
38         public void hardDrop(){
39                 channel.noteOn(100 , 100);
40         }
41
42         public void moveDown(){
43                 channel.noteOn(60 , 100);
44         }
45
46         public void moveRight(){
```

```
47                  channel.noteOn(60 , 100);
48          }
49
50          public void moveLeft(){
51                  channel.noteOn(60 , 100);
52          }
53
54          public void turnRight(){
55                  channel.noteOn(60 , 100);
56          }
57
58          public void turnLeft(){
59                  channel.noteOn(60 , 100);
60          }
61
62          public void delete(int deleteNum, int ren){
63                  channel.noteOn(127 , 100);
64          }
65
66          public void hind(int num){
67                  if(num == 0){
68                          channel.noteOn(30 , 0);
69                  }else if(num < 3){
70                          channel.noteOn(30 , 100);
71                  }else if(num < 10){
72                          channel.noteOn(25 , 100);
73                  }else{
74                          channel.noteOn(20 , 100);
75                  }
76          }
77
78          public void win(){
79                  channel.noteOn(30 , 100);
80                  try {
81                          Thread.sleep(1000);
82                  } catch (InterruptedException e) {
83                          e.printStackTrace();
84                  }
85                  stop();
86                  channel.noteOn(60 , 100);
87                  try {
88                          Thread.sleep(1000);
89                  } catch (InterruptedException e) {
90                          e.printStackTrace();
91                  }
92                  stop();
93
94          }
95
96          public void lose(){
97                  channel.noteOn(60 , 100);
98                  try {
99                          Thread.sleep(1000);
100                 } catch (InterruptedException e) {
101                         e.printStackTrace();
102                 }
103                 stop();
104                 channel.noteOn(30 , 100);
105                 try {
106                         Thread.sleep(1000);
107                 } catch (InterruptedException e) {
```

```
108                         e.printStackTrace();
109                 }
110                 stop();
111
112         }
113
114         /**
115          * @param args
116          */
117         public static void main(String[] args) {
118                 // TODO Auto-generated method stub
119                 GameEffect effect = new GameEffect();
120         }
121
122 }
```

ソースコード 9: GameField.java

```
1  package gametetris;
2
3  import java.awt.event.ActionEvent;
4  import java.io.FileNotFoundException;
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7  import java.io.ObjectOutputStream;
8  import java.io.Serializable;
9
10 public class GameField implements Serializable {
11
12         private int FieldRows = 23;
13         private int FieldCols = 12;
14         private int [][] field = new int[FieldRows][FieldCols];
15         private int delNumber;
16         private int[] delRows = new int[FieldRows];
17         private int hindNum;
18         private boolean hindFlag;
19         private int sendCheckNum;
20         private int getCheckNum;
21         private boolean endFlag;
22         private int renNumber;
23         private int point;
24
25         public GameField(){
26                 hindNum = 0;
27                 delNumber = 0;
28                 sendCheckNum = 0;
29                 getCheckNum = -1;
30                 renNumber = 0;
31                 point = 0;
32                 hindFlag = false;
33                 endFlag = false;
34                 newField();
35         }
36
37         public void newField(){
38                 for(int col=0; col < FieldCols; col++){
39                         if(col < FieldCols/2-2 || FieldCols/2+1 < col){
40                                 field[0][col] = 100;
41                         }else{
42                                 field[0][col] = 0;
43                         }
44                 }
```

```
45              for(int col=0; col < FieldCols; col++){
46                      if(col < FieldCols/2−2 || FieldCols/2+1 < col){
47                              field[1][col] = 9;
48                      }
49              }
50              for(int row=2; row < FieldRows; row++){
51                      //左の壁
52                      field[row][0] = 9;
53                      //空間
54                      for(int col=1; col < FieldCols; col++){
55                              field[row][col] = 0;
56                      }
57                      //右の壁
58                      field[row][FieldCols−1] = 9;
59              }
60              //フィールドの底
61              for(int col=0; col < FieldCols; col++){
62                      field[FieldRows−1][col] = 9;
63              }
64      }
65
66      public int get_Field(int row, int col){
67              return field[row][col];
68      }
69
70      public int[][] get_AllField(){
71              return field;
72      }
73
74      public void set_Field(int row, int col, int blockNum){
75              field[row][col] = blockNum;
76      }
77
78      public void set_BlockField(GameBlock block){
79              for(int i = 0; i < block.get_BlockSize(); i++){
80                      for(int j = 0; j < block.get_BlockSize(); j++){
81                              if(i+block.get_Y() < 0 || j+block.get_X() < 0)
                                    continue;
82                              if(block.get_Block(i, j) != 0 && get_Field(i+block.
                                    get_Y(), j+block.get_X()) != 9){
83                                      set_Field(i+block.get_Y(), j+block.get_X(),
                                            block.get_Block(i, j));
84                              }
85                      }
86              }
87      }
88
89      public void unset_BlockField(GameBlock block){
90              for(int i = 0; i < block.get_BlockSize(); i++){
91                      for(int j = 0; j < block.get_BlockSize(); j++){
92                              if(i+block.get_Y() < 0 || j+block.get_X() < 0)
                                    continue;
93                              if(block.get_Block(i, j) != 0 && get_Field(i+block.
                                    get_Y(), j+block.get_X()) != 9){
94                                      set_Field(i+block.get_Y(), j+block.get_X(), 0);
95                              }
96                      }
97              }
98      }
99
100     public boolean checkLine(){
```

```
101             int del = 0;
102             int cells;
103             for(int row=2; row < FieldRows−1; row++){
104                     cells = 0;
105                     for(int col=1; col < FieldCols−1; col++){
106                             if(field[row][col]!=0){
107                                     cells++;
108                             }else{
109                                     break;
110                             }
111                     }
112                     if(cells == FieldCols−2){
113                             delRows[del++] = row;
114                     }
115             }
116             if(del != 0){
117                     delNumber = del;
118                     return true;
119             }else{
120                     return false;
121             }
122     }
123
124     public void deleteLine(){
125             for(int n = 0; n < delNumber; n++){
126                     for(int col=1; col < FieldCols−1; col++){
127                             field[delRows[n]][col] = 0;
128                     }
129             }
130             renNumber++;
131             point += (delNumber * delNumber + renNumber) * 100;
132     }
133
134     public void killLine(){
135
136             for(int n=0; n < delNumber; n++){
137
138                     for(int row=delRows[n]; row > 2; row−−){
139                             for(int col=1; col < FieldCols−1; col++){
140                                     field[row][col] = field[row−1][col];
141                             }
142                     }
143
144                     for(int col=1; col < FieldCols−1; col++){
145                             field[3][col] = 0;
146                     }
147
148             }
149     }
150
151     public void hindrance(){
152             for(int row = 0; row < 2; row++){
153                     for(int col=FieldCols/2−2; col < FieldCols/2+2; col++){
154                             if(row + hindNum < FieldRows − 1){
155                                     field[row][col]=field[row+hindNum][col];
156                             }
157                     }
158             }
159             for(int row = 2; row < FieldRows−1; row++){
160                     for(int col=1; col < FieldCols−1; col++){
161                             if(row + hindNum < FieldRows − 1){
```

```
162                                     field[row][col]=field[row+hindNum][col];
163                             }
164                     }
165             }
166             for(int row = FieldRows - hindNum - 1; row < FieldRows-1; row
                    ++){
167                     int x = ((int)(Math.random()*10))+1;
168                     if(row < 0)continue;
169                     if(row < 2){
170                             for(int col=FieldCols/2-2; col < FieldCols/2+2; col
                                    ++){
171                                     if(col == x){
172                                             field[row][col] = 0;
173                                     }else{
174                                             field[row][col] = 10;
175                                     }
176                             }
177                     }else{
178                             for(int col=1; col < FieldCols-1; col++){
179                                     if(col == x){
180                                             field[row][col] = 0;
181                                     }else{
182                                             field[row][col] = 10;
183                                     }
184                             }
185                     }
186             }
187             hindFlag = true;
188     }
189
190     public int getHind(){
191             return hindNum;
192     }
193
194     public void setHindZero(){
195             hindNum = 0;
196     }
197
198     public void setDeleteNumber(int n){
199             delNumber = n;
200     }
201
202     public int getDleteNumber(){
203             int number = delNumber;
204             return number;
205     }
206
207     public void setRenNumber(int n){
208             renNumber = n;
209     }
210
211     public void setHindNum(int n){
212             hindNum = n;
213     }
214
215     public void addHindNum(int n){
216             hindNum = hindNum + n;
217     }
218
219     public void addPoint(int n){
220             point += n;
```

```
221         }
222
223         public int getPoint(){
224                 return point;
225         }
226
227         public void debug(){
228                 for(int i = 0; i < FieldRows; i++){
229                         for(int j = 0; j < FieldCols; j++){
230                                 System.out.print(field[i][j]);
231                         }
232                         System.out.println();
233                 }
234                 System.out.println();
235         }
236
237         public void sendCheck(){
238                 int n = (int)(Math.random()*10000000);
239                 if(sendCheckNum == n){
240                         sendCheckNum = n + (int)(Math.random()*10000000) + 1;
241                 }else{
242                         sendCheckNum = n;
243                 }
244         }
245
246         public int getsendCheck(){
247                 return sendCheckNum;
248         }
249
250         public void returnCheck(GameField field){
251                 getCheckNum = field.getsendCheck();
252         }
253
254         public int getCheck(){
255                 return getCheckNum;
256         }
257
258         public boolean check(GameField field){
259                 if(getCheckNum == field.getsendCheck()){
260                         return true;
261                 }
262                 return false;
263         }
264
265         public void setCheck(int n){
266                 getCheckNum = n;
267         }
268
269
270         public GameField serialize() {
271                 return this;
272         }
273
274         public void setEnd(){
275                 endFlag = true;
276         }
277
278         public boolean getEnd(){
279                 return endFlag;
280         }
281
```

```
282         public void setRenZero(){
283                 renNumber = 0;
284         }
285
286         public int getRen(){
287                 return renNumber;
288         }
289
290         public void setHindFrag(boolean b){
291                 hindFlag = b;
292         }
293
294         public boolean hindPermisson(){
295                 return hindFlag;
296         }
297 }
```

ソースコード 10: GameInfo.java

```
1  package gametetris;
2
3  public class GameInfo {
4          private int renNum;
5          private int deleteNum;
6          private int hindNum;
7          private int point;
8
9          public GameInfo() {
10                 renNum = 0;
11                 deleteNum = 0;
12                 hindNum = 0;
13                 point = 0;
14         }
15
16         public void setInfo(GameField field){
17                 renNum = field.getRen();
18                 deleteNum = field.getDleteNumber();
19                 hindNum = field.getHind();
20                 point = field.getPoint();
21         }
22
23         public int getRen(){
24                 return renNum;
25         }
26
27         public int getDelete(){
28                 return deleteNum;
29         }
30
31         public int getHind(){
32                 return hindNum;
33         }
34
35         public int getPoint(){
36                 return point;
37         }
38
39         public void debug(){
40                 System.out.println("renNum:␣"+ renNum);
41                 System.out.println("deleteNum:␣"+deleteNum);
42                 System.out.println("hindNum:␣"+hindNum);
```

```
43                  System.out.println("point:␣"+point);
44          }
45
46
47          /**
48           * @param args
49           */
50          public static void main(String[] args) {
51                  // TODO Auto-generated method stub
52
53          }
54
55  }
```

ソースコード 11: GameTimer.java

```
1   package gametetris;
2
3   public class GameTimer extends Thread {
4           private long time0;
5           private long time1;
6           private long pastTime;
7           private boolean timer_on;
8
9           public Thread TetrisTimerThread;
10
11          GameTimer(){
12                  time0  = 0;
13                  time1  = 0;
14                  pastTime = 0;
15                  timer_on = false; //初期状態では計測未開始
16          }
17
18
19          public void timerStart(){
20                  timer_on = true;
21          }
22
23
24          public void timerStop(){
25                  timer_on = false;
26          }
27
28          public void timerReset(){
29                  time0 = 0;
30                  pastTime = 0;
31          }
32
33          public long getTime(){
34                  return pastTime;
35          }
36
37          public void run(){
38                  try {
39                          while(true){
40                                  if(timer_on){
41                                          if(time0 == 0){
42                                                  time0 = System.currentTimeMillis();
43                                                  pastTime = 0;
44                                          }else{
45                                                  time1  = System.currentTimeMillis();
```

```
46                                                  pastTime = pastTime + (time1 −
                                                        time0);
47                                                  time0   = time1;
48                                          }
49                                  }else{
50                                          time0 = 0;
51                                  }
52                                  this.sleep(10);
53                          }
54                  }catch(Exception ex){
55                          ex.printStackTrace();
56                  }
57          }
58
59          public static void main(String[] args){
60                  GameTimer timer = new GameTimer();
61                  timer.start();
62                  timer.timerStart();
63                  while(true){
64                          long a = timer.getTime();
65                          System.out.println(a);
66                  }
67          }
68  }
```

ソースコード 12: GameClient.java

```
1   package gametetris;
2
3   import java.net.ServerSocket;
4   import java.net.Socket;
5   import java.net.UnknownHostException;
6   import java.io.IOException;
7   import java.io.ObjectOutputStream;
8   import java.io.ObjectInputStream;
9   import java.util.ArrayList;
10
11  public class GameClient extends Thread{
12          Socket socket;
13          GameField field1;
14          GameField field2;
15
16          public GameClient(){
17
18          }
19
20          public boolean setSocket(String ipAddress){
21                  try {
22                          socket = new Socket(ipAddress, 26000);
23                          System.out.println(socket.getInetAddress());
24                  } catch (UnknownHostException e1) {
25                          e1.printStackTrace();
26                          return false;
27                  } catch (IOException e1) {
28                          e1.printStackTrace();
29                          return false;
30                  }
31                  field1 = null;
32                  field2 = null;
33                  start();
34                  return true;
```

```
35          }
36
37          public void run(){
38                  while(true){
39                          try{
40                                  // オブジェクト出力ストリーム
41                                  ObjectOutputStream oos = new ObjectOutputStream
                                        ( socket.getOutputStream() );
42                                  // 書き出し
43                                  oos.writeObject(field1);
44
45                                  // オブジェクト入力ストリーム
46                                  ObjectInputStream ois = new ObjectInputStream(
                                        socket.getInputStream() );
47                                  field2 = (GameField)(ois.readObject());
48                          } catch(Exception e){
49                                  //e.printStackTrace();
50                          }
51                  }
52          }
53
54          public void sendGameField(GameField field){
55                  field1 = field;
56          }
57
58          public GameField fetchGameField(){
59                  return field2;
60          }
61
62          public void close(){
63                  try {
64                          socket.close();
65                  } catch (IOException e) {
66                          // TODO Auto-generated catch block
67                          e.printStackTrace();
68                  }
69          }
70  }
```

ソースコード 13: GameNetwork.java

```
1   package gametetris;
2   import java.net.InetAddress;
3   import java.net.ServerSocket;
4   import java.net.Socket;
5   import java.net.UnknownHostException;
6   import java.awt.BorderLayout;
7   import java.awt.Container;
8   import java.io.IOException;
9   import java.io.ObjectOutputStream;
10  import java.io.ObjectInputStream;
11  import java.util.ArrayList;
12
13  import javax.swing.BoxLayout;
14  import javax.swing.JButton;
15  import javax.swing.JFrame;
16  import javax.swing.JLabel;
17  import javax.swing.JTextField;
18
19  public class GameNetwork extends Thread{
20
```

```
21      JFrame frame;
22      Container contentPane;
23      JLabel label;
24      JLabel label2;
25      JLabel label3;
26      JLabel label4;
27
28      ServerSocket serverSocket1;
29      ServerSocket serverSocket2;
30
31      public static void main(String[] args){
32              GameNetwork gmn = new GameNetwork();
33              gmn.start();
34      }
35
36      public GameNetwork(){
37              try {
38                      serverSocket1 = new ServerSocket(26000);
39              } catch (IOException e1) {
40                      // TODO Auto-generated catch block
41                      e1.printStackTrace();
42              }
43              InetAddress addr = null;
44              try {
45                      addr = InetAddress.getLocalHost();
46              } catch (UnknownHostException e) {
47                      // TODO Auto-generated catch block
48                      e.printStackTrace();
49              }
50              frame = new JFrame("GameTetrisServer");
51              frame.setBounds(30, 30, 500, 500);
52              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
53              contentPane = frame.getContentPane();
54              contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.
                    Y_AXIS));
55              label = new JLabel("Server␣Running");
56              contentPane.add(label);
57              label2 = new JLabel("IP␣Address:␣"+addr.getHostAddress());
58              contentPane.add(label2);
59              label3 = new JLabel("Player1:None");
60              contentPane.add(label3);
61              label4 = new JLabel("Player2:None");
62              contentPane.add(label4);
63              frame.setVisible(true);
64              // try {
65              //  serverSocket2 = new ServerSocket(27000);
66              // } catch (IOException e) {
67              //  // TODO Auto-generated catch block
68              //  e.printStackTrace();
69              // }
70      }
71
72      public void run() {
73              try {
74
75                      while(true){
76                              System.out.println("クライアントからの接続を待ちま
                                    す");
77                              Socket s1 = serverSocket1.accept();
78                              System.out.println("プレイヤー 1 OK!");
79                              label3.setText("Player1:OK");
```

```
80                              Socket s2 = serverSocket1.accept();
81                              System.out.println("プレイヤー２ OK!");
82                              label4.setText("Player2:OK");
83
84                              while(true) {
85                                      try {
86                                              // 受信
87                                              ObjectInputStream ois1 = new
                                                    ObjectInputStream(s1.
                                                    getInputStream());
88                                              GameField data1 = (GameField)(ois1.
                                                    readObject());
89                                              ObjectInputStream ois2 = new
                                                    ObjectInputStream(s2.
                                                    getInputStream());
90                                              GameField data2 = (GameField)(ois2.
                                                    readObject());
91
92                                              // 返信
93                                              ObjectOutputStream oos1 = new
                                                    ObjectOutputStream(s1.
                                                    getOutputStream());
94                                              oos1.writeObject(data2);
95                                              ObjectOutputStream oos2 = new
                                                    ObjectOutputStream(s2.
                                                    getOutputStream());
96                                              oos2.writeObject(data1);
97                                              try {
98                                                      Thread.sleep(100);
99                                              } catch (InterruptedException e) {
100                                                     e.printStackTrace();
101                                             }
102                                     } catch (Exception ex) {
103                                             System.out.println("切断されました");
104                                             label3.setText("Player1:None");
105                                             label4.setText("Player2:None");
106                                             ex.printStackTrace();
107                                             break;
108                                     }
109                             }
110                     }
111             } catch (Exception e) {
112                     e.printStackTrace();
113             }
114     }
115 }
```