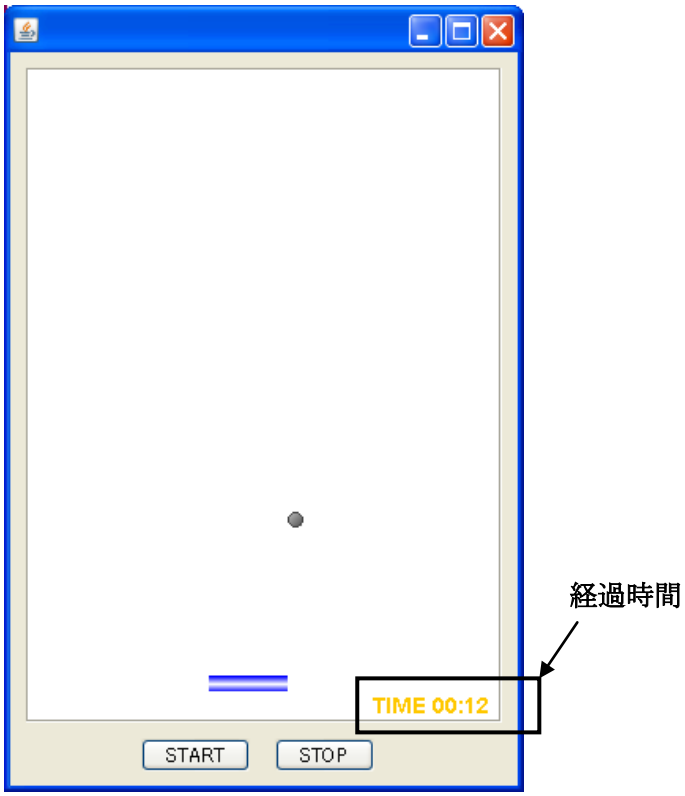
【ブロック崩し　Step2-2】

ゲームが開始してからの経過時間を表示する



＜Java プログラムでの時間取得方法＞

Java プログラムでは、簡易に実行時間を測りたいときに

```
long t = System.currentTimeMillis();
```

のように書くことが多い。変数 t には現在の時刻を表す long 型の数値がはいる。具体的にはこれは、ミリ秒単位で測定した現在時刻と UTC(協定世界時) 1970 年 1 月 1 日午前 0 時との差である。

＜作成手順＞

1．BlockBreakerPanel.java の編集（次ページ以降を参照）

2．実行して以下を確認

・START ボタンでボールが動き、反射すること

・ゲーム開始からの経過時間が表示されること

・STOP ボタンでゲームを中断できること（その際、経過時間もストップする）

BlockBreakerPanel.java

```
1  package animation;
2
3  import java.awt.Color;
4  import java.awt.Font;
5  import java.awt.GradientPaint;
6  import java.awt.Graphics;
7  import java.awt.Graphics2D;
8  import java.awt.Point;
9  import java.awt.Rectangle;
10 import java.awt.event.ActionEvent;
11 import java.awt.event.ActionListener;
12 import java.awt.event.KeyEvent;
13 import java.awt.event.KeyListener;
14 import java.text.SimpleDateFormat;
15 import java.util.Date;
16
17 import javax.swing.JOptionPane;
18 import javax.swing.JPanel;
19 import javax.swing.Timer;
20
21 public class BlockBreakerPanel extends JPanel implements KeyListener, ActionListener{
22
23     private Rectangle paddle;     // パドル
24     private int pWidth = 50;      // パドルの幅
25     private int pHeight = 10;     // パドルの高さ
26     private Circle ball;          // ボール
27     private int radius = 5;       // ボールの半径
28     private int initd[] = {3, 3};         // ボールの移動量の初期値
29     private int dx=initd[0], dy=initd[1]; // ボールの移動量
30     private Point pStartPoint, bStartPoint;       // パドル、ボールの初期位置
31
32     private Timer timer;
33     private long startTime;       // ゲームがスタートした時の時刻
34     private long stopTime;        // ゲームが途中で中断した時の時刻
35     private long lossTime;        // 中断した時間
36     private boolean onGameFlg = false;        // ゲーム中はtrue
37
38     private Date date;                // ゲーム経過時間を格納する
39     private SimpleDateFormat sdf;     // 時間をフォーマットして表示するためのもの
40     private Font font;                // 時刻表示のフォント
41
42     /**
43      * コンストラクタ
44      */
45     public BlockBreakerPanel() {
46         // ボールを表す円を生成
47         ball = new Circle(radius);
48         // タイマー生成
49         timer = new Timer(15, this);
50         // Date:日付と時刻を表すクラス
51         date = new Date(0);
52         // SimpleDateFormat：日付や時刻をフォーマットするためのクラス
53         // このように初期化すると例えば5分30秒を 05:30 のように表示できる
54         sdf = new SimpleDateFormat("mm:ss");
55         // 論理フォント名Dialog(WindowsではMSゴシック)、太字でフォントサイズ14
56         font = new Font("Dialog", Font.BOLD, 14);
57         // キーイベントを受け取れるようにする
58         addKeyListener(this);
59     }
60
```

```
61
62      /**
63       * ゲームをスタートさせる
64       */
65      public void start() {
66          // System.currentTimeMillis()で現在の時刻がミリ秒で取得できる
67          if (onGameFlg)
68              // ゲーム中断時間を積算
69              lossTime += (System.currentTimeMillis() - stopTime);
70          else
71              // ゲームスタート時間
72              startTime = System.currentTimeMillis();
73          // タイマースタート
74          timer.start();
75          // ゲーム中フラグをtrue
76          onGameFlg = true;
77          // フォーカスを要求（Keyイベントを有効にするため）
78          if (!isFocusOwner())
79          requestFocus();
80      }
81      /**
82       * ゲームを中断させる
83       */
84      public void stop() {
85          // ゲーム中断時刻
86          stopTime = System.currentTimeMillis();
87          // タイマーストップ
88          timer.stop();
89          // フォーカスを要求（Keyイベントを有効にするため）
90          if (!isFocusOwner())
91          requestFocus();
92      }
93
94      /**
95       * 描画メソッド
96       */
97      @Override
98      public void paintComponent(Graphics g){
99          // ボタンなど配置していなければいらないが、忘れずに書く習慣にする
100         super.paintComponent(g);
101         // より高度な描画を可能にするようキャスト
102         Graphics2D g2d = (Graphics2D)g;
103         // 各描画対象の初期位置を計算して設定
104         /* このパネルがフレームに貼り付けられないとサイズが確定しないので
105          * コンストラクタの中ではgetWidth()などに正しい値がはいらない。
106          * そのため、ここでパドルの初期化と初期位置の決定を行う */
107         if (paddle == null) {
108             // パドルをRectangleオブジェクトとして使う
109             paddle = new Rectangle(pWidth, pHeight);
110             // パドルの初期位置はx方向は中心、y方向は下から20あけたところ
111             paddle.setLocation( x座標 , y座標 );
112             // ボールの初期位置
113             ball.setLocation(10, getHeight()/2);
114             // パドル、ボールの初期位置を覚えておく
115             pStartPoint = paddle.getLocation();
116             bStartPoint = ball.getLocation();
117         }
118
119         /* 白色背景 ----------------------------------------*/
120         // 白をセット
121         g2d.setColor(Color.WHITE);
```

```
122         // 四角で塗りつぶす　fillRect(左上のx座標, 左上のy座標, 幅, 高さ)
123         g2d.fillRect(0, 0, getWidth(), getHeight());
124 
125         /* パドル描画 ----------------------------------------*/
126         // グラデーションパターンを指定する
127         // new GradientPaint(点1のx座標, 点1のy座標, 色1, 点2のx座標, 点2のy座標, 色2, 循環するか)
128         GradientPaint gp = new GradientPaint(paddle.x, paddle.y, Color.BLUE,
129                 paddle.x, paddle.y+paddle.height/2, Color.WHITE, true);
130         g2d.setPaint(gp);
131         // パドルを描画
132         g2d.fill(paddle);
133 
134         /* ボール描画 ----------------------------------------*/
135         // グラデーションパターンを指定する（ライトグレー → ダークグレー）
136         gp = new GradientPaint(ball.x, ball.y, Color.LIGHT_GRAY,
137                 ball.x+ball.radius*2, ball.y+ball.radius*2, Color.DARK_GRAY, true);
138         g2d.setPaint(gp);
139         // 塗りつぶしで円描画
140         /* fillOval(左上の点のx座標, y座標, 幅, 高さ) */
141         g2d.fillOval(ball.x, ball.y, 2*radius, 2*radius);
142         // 円のエッジをダークグレイにする
143         g2d.setColor(Color.DARK_GRAY);
144         g2d.drawOval(ball.x, ball.y, 2*radius-1, 2*radius-1);
145 
146         /* 時間描画 ----------------------------------------*/
147         // ゲームがスタートしてからの経過時間を取得
148         if (onGameFlg)
149             date.setTime(System.currentTimeMillis() - startTime - lossTime);
150         else
151             date.setTime(0);
152         // フォントをセット
153         g2d.setFont(font);
154         // オレンジ色をセット
155         g2d.setColor(Color.ORANGE);
156         // 文字列（時間）を描画する
157         /* drawString(文字列, 左上のx座標, 左上のy座標)*/
158         g2d.drawString("TIME "+sdf.format(date), getWidth()-85, getHeight()-6);
159 
160     }
161 
162     /**
163      * キーを押すと呼ばれる<br>
164      * 矢印キーでパドルが左右に動く
165      */
166     @Override
167     public void keyPressed(KeyEvent e) {
168         int pdx = 10;    // １回のキー操作でずらす量
169         // 押されたキーのコードを取得
170         int keyCode = e.getKeyCode();
171         // キーコードで分岐
172         switch (keyCode) {
173         // 左矢印キーだったら
174         case KeyEvent.VK_LEFT:
175             // パドルが左に10ずれる（壁にめりこまない）
176             // Rectangleクラスの位置指定メソッド setLocation(左上の点のx座標, y座標)
177             if (paddle.x < pdx)
178                 paddle.setLocation(0, paddle.y);
179             else
180                 paddle.setLocation(paddle.x-pdx, paddle.y);
181             // 再描画
182             repaint();
```

```
183             break;
184         // 右矢印キーだったら
185         case KeyEvent.VK_RIGHT:
186             // パドルが右に10ずれる（壁にめりこまない
187             if (getWidth()-paddle.x-paddle.width < pdx)
188                 paddle.setLocation(getWidth()-paddle.width, paddle.y);
189             else
190                 paddle.setLocation(paddle.x+pdx, paddle.y);
191             // 再描画
192             repaint();
193         default:
194             break;
195         }
196     }
197 
198     /**
199      * タイマーによって一定間隔で呼ばれる
200      */
201     @Override
202     public void actionPerformed(ActionEvent e) {
203         // ボールの位置決定
204         ball.setLocation(ball.x+dx, ball.y+dy);
205         // 壁にぶつかったら反射
206         // 左右の壁ならx方向の速度をかえる
207         if (ball.getLeftPoint().x <= 0 ) {
208             ball.setLocation(0, ball.y); // 壁にめり込まない
209             dx = -dx;
210         } else if (ball.getRightPoint().x >= getWidth()) {
211             ball.setLocation(getWidth()-2*ball.radius, ball.y);    // 壁にめり込まない
212             dx = -dx;
213         }
214         // 上の壁ならy方向の速度をかえる
215         if (ball.getTopPoint().y <= 0) {
216             ball.setLocation(ball.x, 0); // 壁にめり込まない
217             dy = -dy;
218         }
219         // パドルにあたったら反射
220         if (paddle.contains(ball.getBottomPoint())) {
221             ball.setLocation(ball.x, paddle.y-2*ball.radius);
222             dy = -dy;
223         }
224         // 落ちたらゲーム終了、メッセージ表示
225         else if (ball.getTopPoint().y >= getHeight()) {
226             terminate("GAME OVER", "¥tゲーム終了です。");
227         }
228         // 再描画
229         repaint();
230     }
231 
232     /**
233      * ゲームを終了する
234      * @param title  メッセージボックスのタイトル
235      * @param message
236      */
237     private void terminate(String title, String message) {
238         // タイマーを止める
239         stop();
240         // メッセージ表示
241         JOptionPane.showMessageDialog(getParent(), message, title, JOptionPane.PLAIN_MESSAGE);
242         // 初期状態に戻す
243         paddle.setLocation(pStartPoint);
```

```
244             ball.setLocation(bStartPoint);
245             dx = initd[0];
246             dy = initd[1];
247             lossTime = 0;
248             // ゲーム中フラグをfalse
249             onGameFlg = false;
250         }
251 
252         /* 以下の2つのメソッドはKeyListenerを
253          * implementsしたため必要だが、空でよい */
254 
255         @Override
256         public void keyReleased(KeyEvent e) {
257         }
258 
259         @Override
260         public void keyTyped(KeyEvent e) {
261         }
262 
263     }
```