電子計算機研究会
第7回オープン授業
ゲーム作成

ゲーム作成の手順

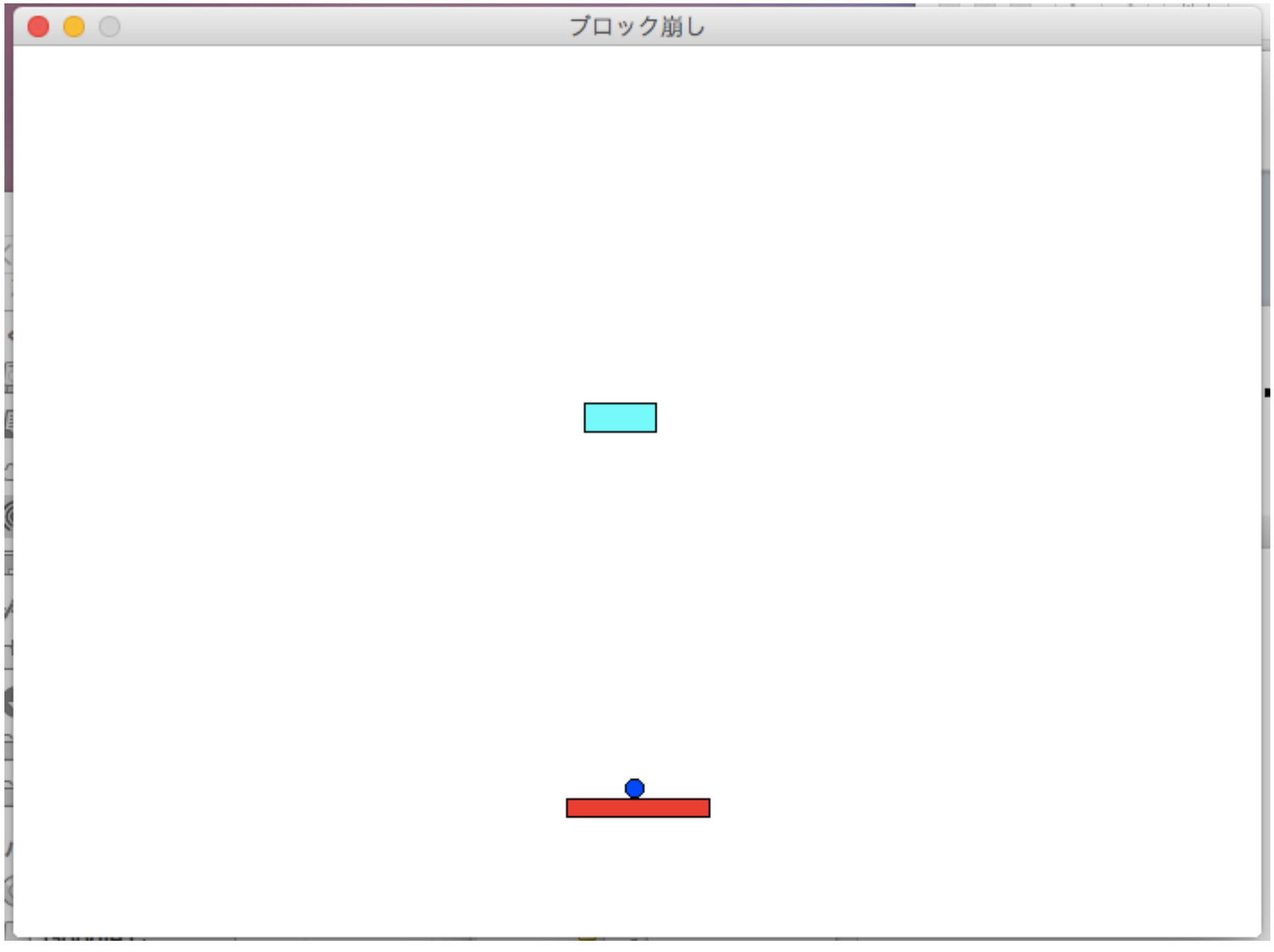
1. 電算Wikiのオープン授業から今回のオーブン授業のファイルをダウンロードする。

2. 電算WikiのDJGL-更新履歴からDensanJavaGameLibrary01_01aをダウンロード

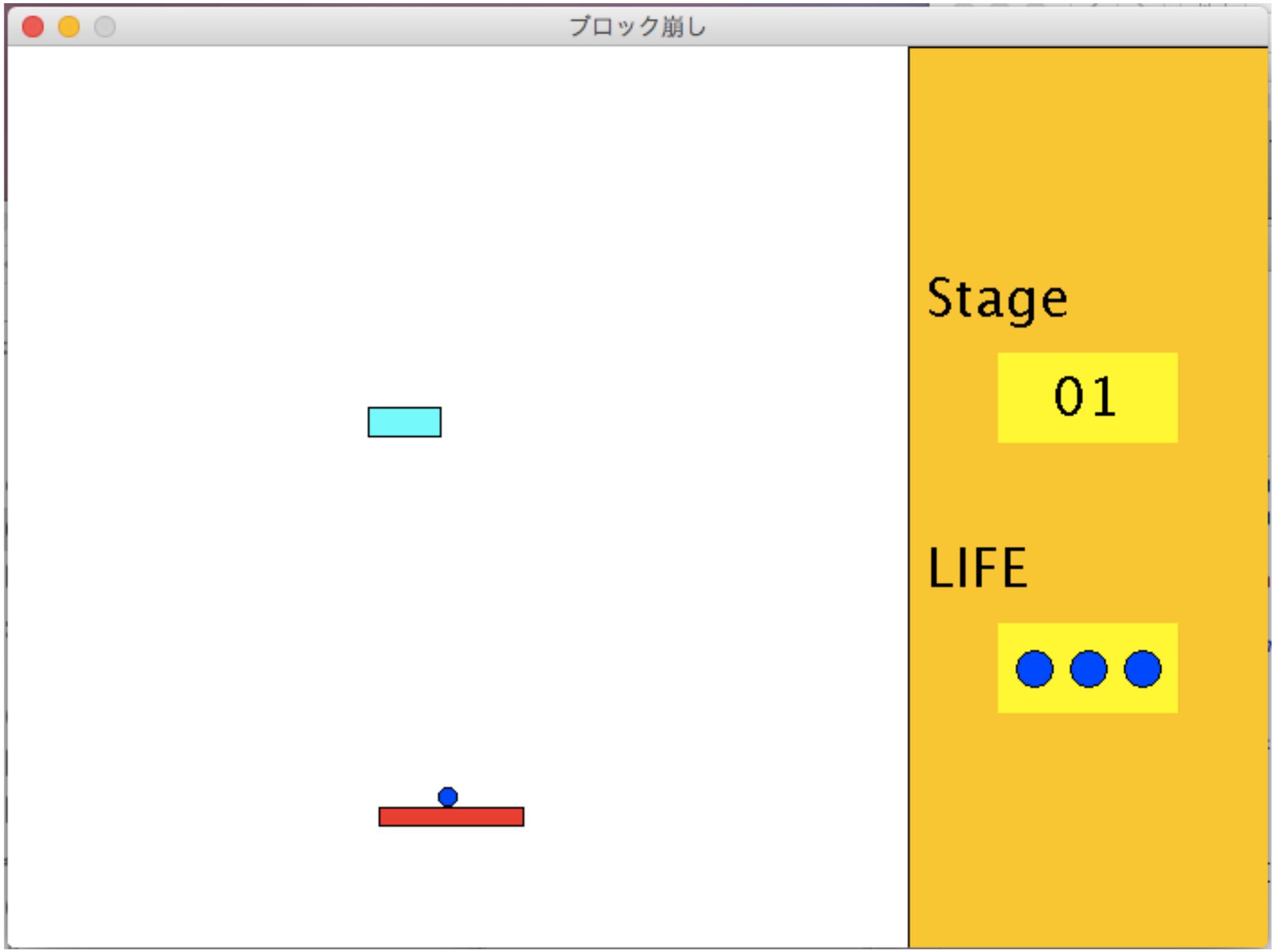
3. eclipseを起動し画面右上のファイルの項目からインポートを選択し、既存プロジェクトをワークスペースへを選択する。そしてダウンロードの項目からオープン授業のファイルを選択し、完了をクリック。

4. eclipse上のbreakBlock1の項目で左クリックし、ビルドパスの項目から、外部アーカイブの追加をクリックする。そして、ダウンロードの項目からDensanJavaGameLibrary01_01aを選択し、開くをクリックする。

ブロック崩し(制作途中)



ブロック崩し(完成版)



breakBlock1の構成

- main…ゲームの基本的な部分
- object…ゲーム上の物の動作の処理
- stage…ステージで行われる動作の処理

今回の課題

1. object内のクラス、Ball,Block,Racketの処理の一部が穴埋めとなっているので、その部分を作成する。

2. 自分自身でステージを作成する。

Racketクラスへの追加

- ラケットの描画(drawメソッド)
- ラケットの移動(updateメソッド)
- ラケットとボールの当たり判定
  (collideWithメソッド)

drawメソッド(Racket)

```
public void draw(Drawer d){
    //色塗り
    d.setColor(Color.RED);
    d.fillRect((int)getX(), (int)getY(), getWidth(), getHeight());

     // 枠線を描画
        d.setColor(Color.BLACK);
        d.drawRect((int)getX(), (int)getY(), getWidth(), getHeight());
    }
```

・長方形の黒の枠線を描画し、その中を赤色で塗りつぶしている

updateメソッド(Racket)

```
public void update(){
    //ラケットの移動

    if(KeyInput.isPressing(KeyEvent.VK_LEFT)&&getX()>0)
       addX(-5);
    else if(KeyInput.isPressing(KeyEvent.VK_RIGHT)&&getX()
+WIDTH<GameManager.getInstance().getFrameWidth())
          addX(5);
      }
```

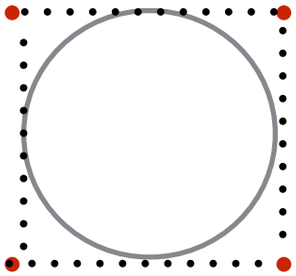
・キーボードで左矢印を押している間、x軸を5のスピードで左に移動。右矢印を押している間、x軸を5のスピードで右に移動する。

CollideWithメソッド(Racket)

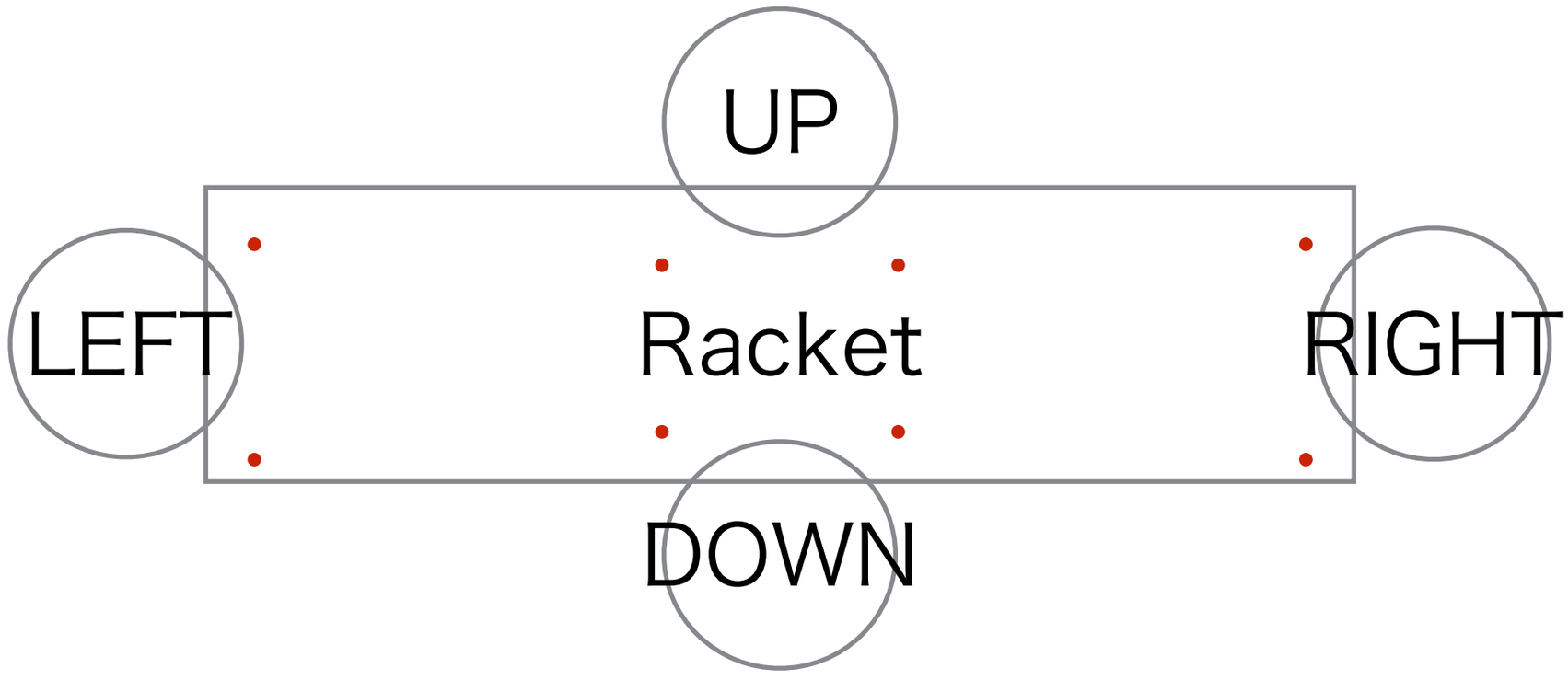
```
public int collideWith(Ball ball) {
        Rectangle racketRect = new Rectangle((int)getX(), (int)getY(), WIDTH,
HEIGHT);

        int ballX = (int)ball.getX();
        int ballY = (int)ball.getY();
        int ballSize = ball.getHeight();

        if (racketRect.contains(ballX, ballY)
            && racketRect.contains(ballX + ballSize, ballY)) {
            // ブロックの下から衝突＝ボールの左上・右上の点がブロック内
            return DOWN;
        } else if (racketRect.contains(ballX + ballSize, ballY)
            && racketRect.contains(ballX + ballSize, ballY + ballSize)) {
            // ブロックの左から衝突＝ボールの右上・右下の点がブロック内
            return LEFT;
        } else if (racketRect.contains(ballX, ballY)
            && racketRect.contains(ballX, ballY + ballSize)) {
            // ブロックの右から衝突＝ボールの左上・左下の点がブロック内
            return RIGHT;
        } else if (racketRect.contains(ballX, ballY + ballSize)
            && racketRect.contains(ballX + ballSize, ballY + ballSize)) {
            // ブロックの上から衝突＝ボールの左下・右下の点がブロック内
            return UP;
        } else if (racketRect.contains(ballX + ballSize, ballY)) {
            // ブロックの左下から衝突＝ボールの右上の点がブロック内
            return DOWN_LEFT;
        } else if (racketRect.contains(ballX, ballY)) {
            // ブロックの右下から衝突＝ボールの左上の点がブロック内
            return DOWN_RIGHT;
        } else if (racketRect.contains(ballX + ballSize, ballY + ballSize)) {
            // ブロックの左上から衝突＝ボールの右下の点がブロック内
            return UP_LEFT;
        } else if (racketRect.contains(ballX, ballY + ballSize)) {
            // ブロックの右上から衝突＝ボールの左下の点がブロック内
            return UP_RIGHT;
        }

        return NO_COLLISION;
    }
```

CollideWithメソッド解説

(ballX,ballY)　　(ballX+ballSize,ballY)



(ballX,ballY+ballSize)　　(ballX+ballSize,ballY+ballSize)



・ballとracketの衝突(上・下・右・左)を確認したい場合、
それぞれ上図の赤点のball座標2つがracket内に含まれているかを確認すればよい

Ballクラスへの追加

- ボールの描画(drawメソッド)
- 画面左右、上に当たった時のボールの方向の反転(updateメソッド)

drawメソッド(Ball)

```
//描画
    public void draw(Drawer d){
       //色を塗る
       d.setColor(Color.BLUE);
       d.fillCircle((int)getX(), (int)getY(), (getHeight() + getWidth())/2);
       // 枠線を描画
       d.setColor(Color.BLACK);
       d.drawCircle((int)getX(), (int)getY(), (getHeight() + getWidth())/2);
    }
```

・円形の黒の枠線を描画し、その中を青色で塗りつぶしている

updateメソッド(Ball)

```
    public void update(){
      addX(vx);
      addY(vy);

      // 左または右に当たったらx方向速度の符号を反転させる
        if (getX() < 0 || getX() >
GameManager.getInstance().getFrameWidth()- getWidth()) {
          vx = -vx;
        }

        // 上に当たったらy方向速度の符号を反転させる
        if (getY() < 0){
          vy = -vy;
        }
    }
```

Blockクラスへの追加

- ブロックの描画(drawメソッド)
- ブロックとボールの当たり判定
  (collideWithメソッド)

drawメソッド(Block)

```
public void draw(Drawer d) {
        //色塗り
            d.setColor(Color.CYAN);
            d.fillRect((int)getX(), (int)getY(), getWidth(), getHeight());
            // 枠線を描画
            d.setColor(Color.BLACK);
            d.drawRect((int)getX(), (int)getY(), getWidth(), getHeight());
    }
```

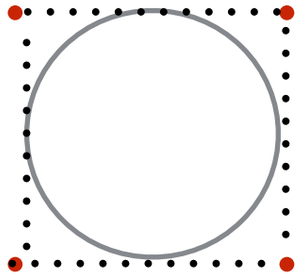
・長方形の黒の枠線を描画し、その中を薄青色で塗りつぶしている

CollideWIth(Block)メソッド

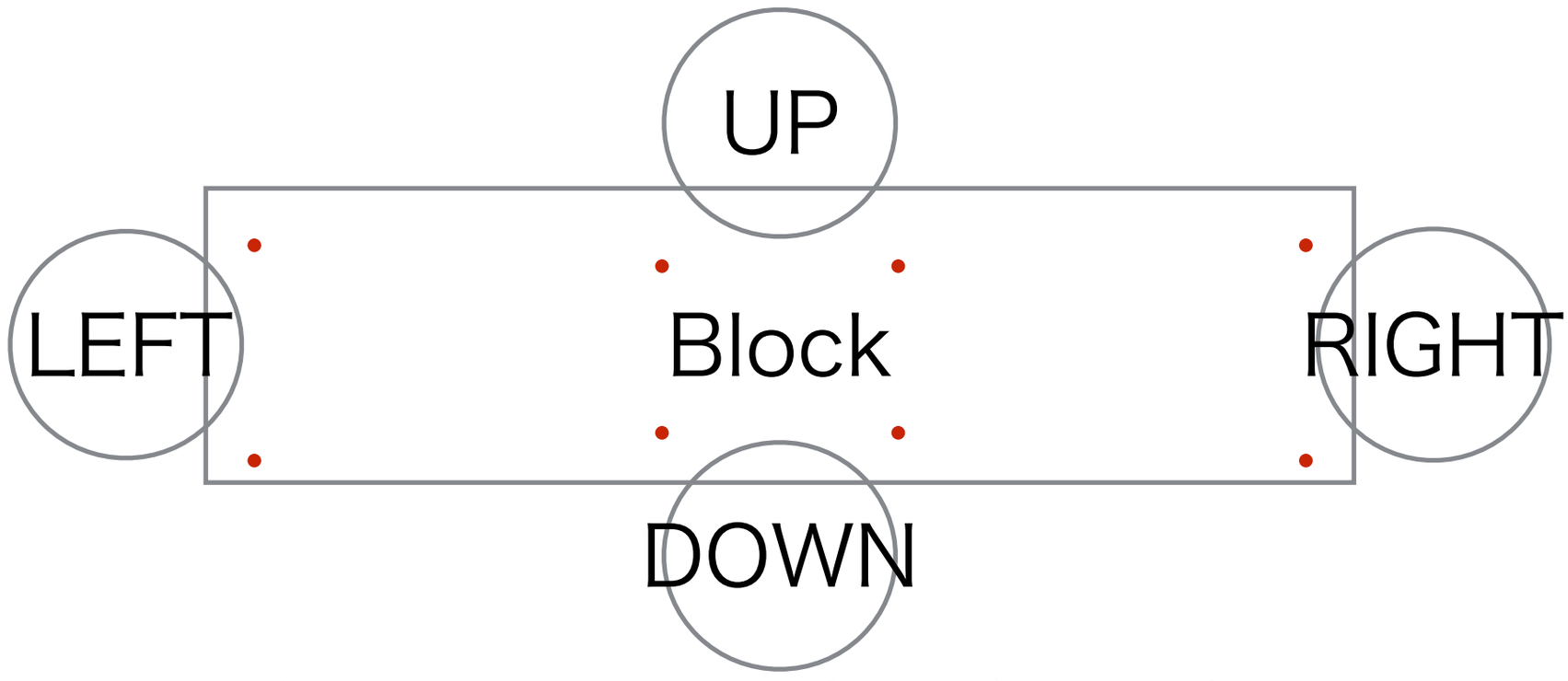
```
public int collideWith(Ball ball) {
        Rectangle blockRect = new Rectangle((int)getX(), (int)getY(), WIDTH, HEIGHT);

        int ballX = (int)ball.getX();
        int ballY = (int)ball.getY();
        int ballSize = ball.getHeight();
        if (blockRect.contains(ballX, ballY)
            && blockRect.contains(ballX + ballSize, ballY)) {
            // ブロックの下から衝突＝ボールの左上・右上の点がブロック内
            return DOWN;
        } else if (blockRect.contains(ballX + ballSize, ballY)
            && blockRect.contains(ballX + ballSize, ballY + ballSize)) {
            // ブロックの左から衝突＝ボールの右上・右下の点がブロック内
            return LEFT;
        } else if (blockRect.contains(ballX, ballY)
            && blockRect.contains(ballX, ballY + ballSize)) {
            // ブロックの右から衝突＝ボールの左上・左下の点がブロック内
            return RIGHT;
        } else if (blockRect.contains(ballX, ballY + ballSize)
            && blockRect.contains(ballX + ballSize, ballY + ballSize)) {
            // ブロックの上から衝突＝ボールの左下・右下の点がブロック内
            return UP;
        }

        return NO_COLLISION;
    }
```

CollideWith(Block)メソッド解説

(ballX,ballY)　　(ballX+ballSize,ballY)



(ballX,ballY+ballSize)　　(ballX+ballSize,ballY+ballSize)



・ballとblockの衝突(上・下・右・左)を確認したい場合、
それぞれ上図の赤点のball座標2つがblock内に含まれているか
を確認すればよい

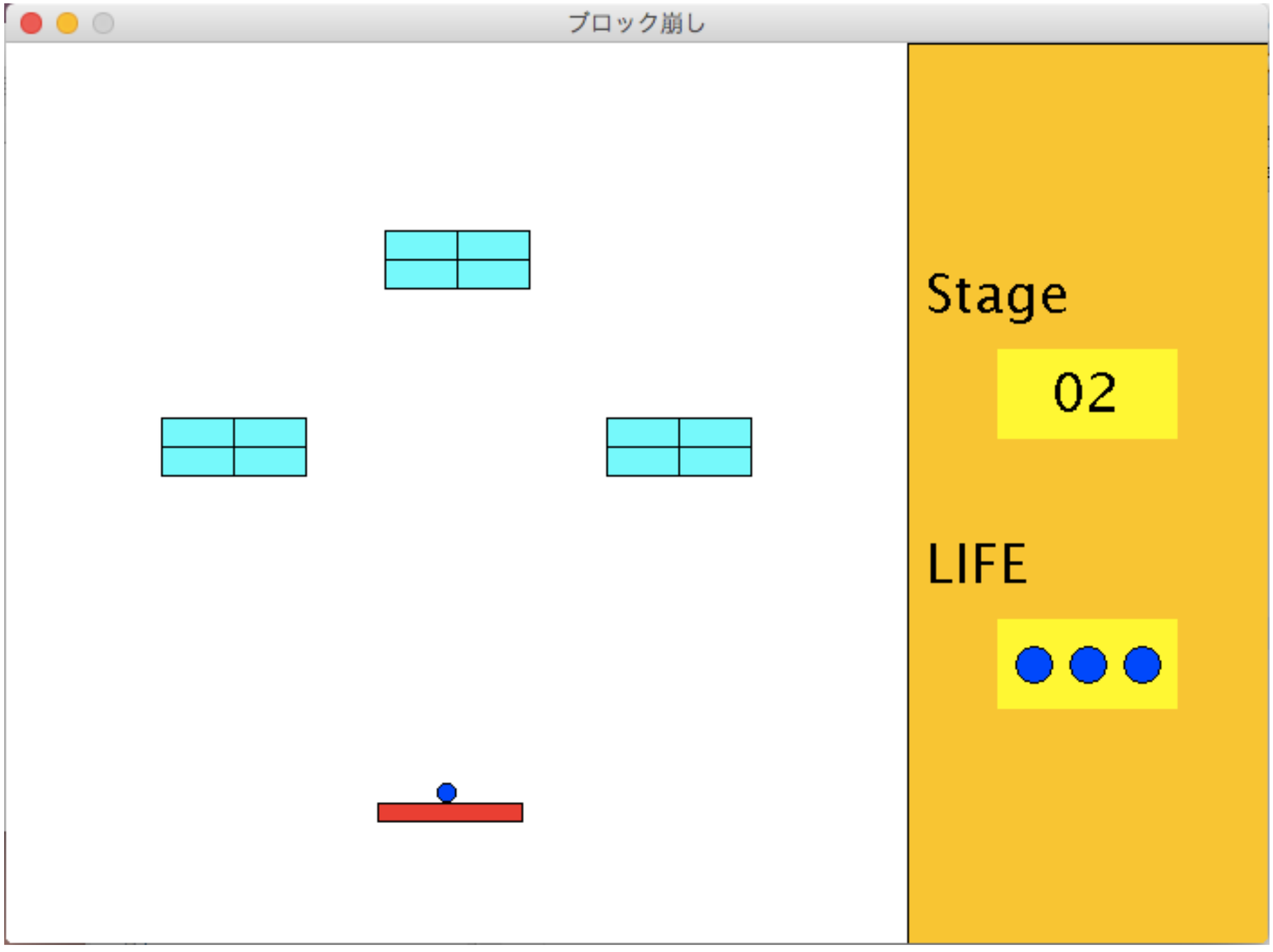
Stageの作成

```
public Stage1(){
      //super(ブロックの数,ステージ番号);
      super(1,1);

      //ブロックの作成 320はx座標, 200はy座標
      block[0] = new Block(320,200);
   }
```

・上の文に新たなブロックの作成をすることで、
ブロックが増やせる。

※新しくステージを作成する際にMainクラスのpublic Main() {}の中のnowScene = new Stage1();
のStage1の部分を自分が作成したステージ名に変えておくこと

ステージ例1



ステージ例1ソースコード

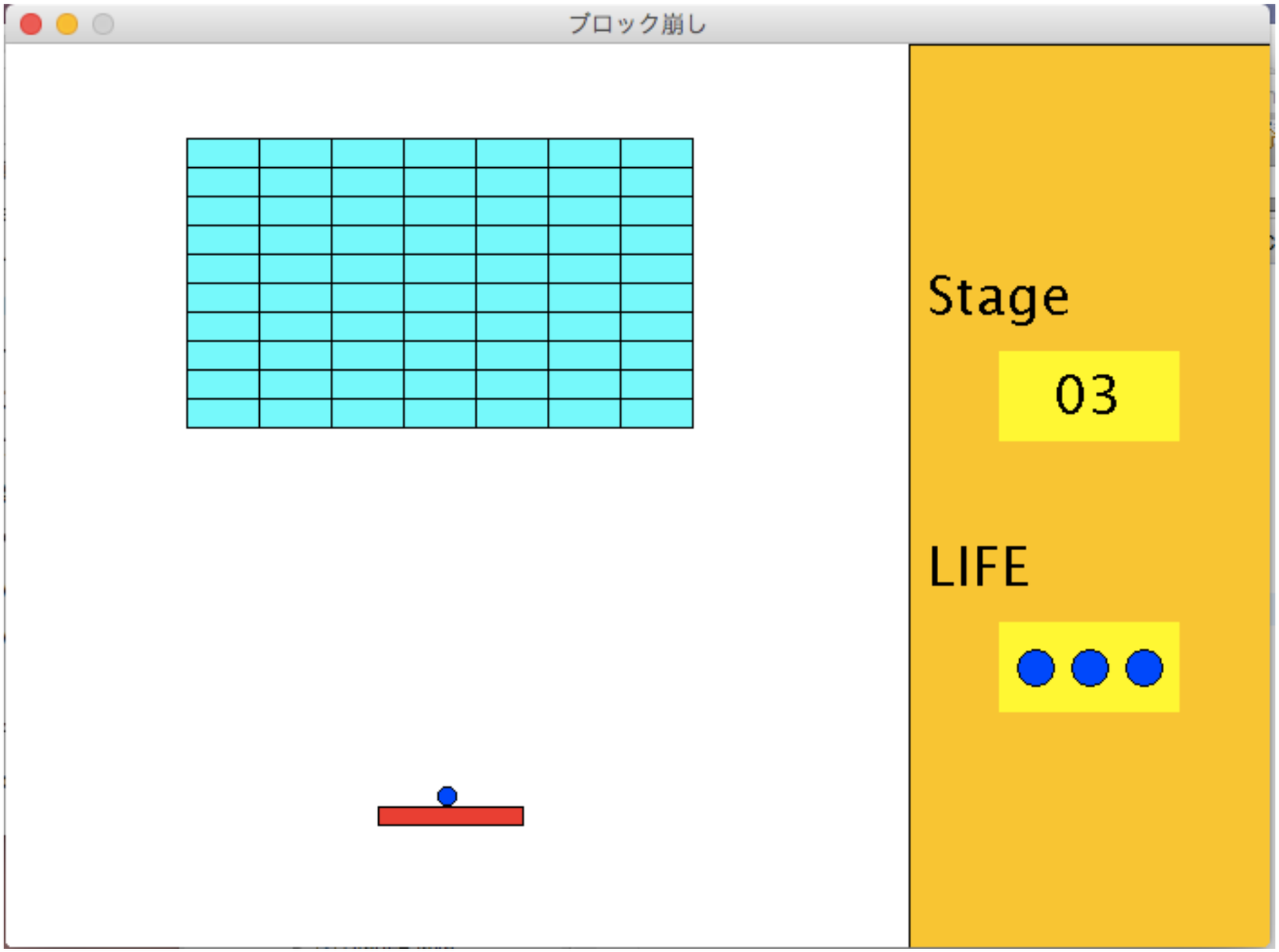
```
package stage;

import object.Block;
import densan.s.game.manager.GameManager;

public class Stage2 extends MainStage{

     // ブロックの行数
    private static final int NUM_BLOCK_ROW = 2;
    // ブロックの列数
    private static final int NUM_BLOCK_COL = 2;

     public Stage2() {
          super(NUM_BLOCK_ROW*NUM_BLOCK_COL*3, 2);

          int x=0;
          int y=0;
          for(int i=0;i<NUM_BLOCK_ROW;i++){
               for(int j=0;j<NUM_BLOCK_COL;j++){
                    x = (GameManager.getInstance().getFrameWidth()-BOARD_WIDTH)/3+(j-2)*Block.WIDTH;
                    y = GameManager.getInstance().getFrameHeight()*2/5+i*Block.HEIGHT;
                    block[i * NUM_BLOCK_COL + j] = new Block(x,y);
               }
          }
          for(int i=0;i<NUM_BLOCK_ROW;i++){
               for(int j=0;j<NUM_BLOCK_COL;j++){
                    x = (GameManager.getInstance().getFrameWidth()-BOARD_WIDTH)*2/3+j*Block.WIDTH;
                    y = GameManager.getInstance().getFrameHeight()*2/5+i*Block.HEIGHT;
                    block[(i+NUM_BLOCK_ROW) * NUM_BLOCK_COL + j] = new Block(x,y);
               }
          }
          for(int i=0;i<NUM_BLOCK_ROW;i++){
               for(int j=0;j<NUM_BLOCK_COL;j++){
                    x = (GameManager.getInstance().getFrameWidth()-BOARD_WIDTH)/2+(j-1)*Block.WIDTH;
                    y = GameManager.getInstance().getFrameHeight()/5+i*Block.HEIGHT;
                    block[(i+NUM_BLOCK_ROW*2) * NUM_BLOCK_COL + j] = new Block(x,y);
               }
          }
     }
```

続く

ステージ例1ソースコード続き

```
    @Override
    public MainStage nowStage() {
        return this;
    }

    @Override
    public MainStage newStage() {
        return new Stage2();
    }

    @Override
    public MainStage nextStage() {
        return null;
    }

}
```

ステージ例2



ステージ例2ソースコード

```
package stage;

import object.Block;
import densan.s.game.manager.GameManager;


public class Stage3 extends MainStage {

    // ブロックの行数
    private static final int NUM_BLOCK_ROW = 10;
    // ブロックの列数
    private static final int NUM_BLOCK_COL = 7;

    public Stage3(){
        super(NUM_BLOCK_ROW * NUM_BLOCK_COL,3);

        //ブロックを並べる
        for(int i=0;i<NUM_BLOCK_ROW;i++){
            for(int j=0;j<NUM_BLOCK_COL;j++){
                int x = j*Block.WIDTH + GameManager.getInstance().getFrameWidth()/7;
                int y = i * Block.HEIGHT + GameManager.getInstance().getFrameHeight()/10;
                block[i * NUM_BLOCK_COL + j] = new Block(x, y);
            }
        }
    }
```

続く

ステージ例2ソースコード

続き

```
@Override
public MainStage nowStage() {
    return this;
}

    @Override
    public MainStage newStage() {
        return new Stage3();
    }

    @Override
    public MainStage nextStage() {
        return null;
    }

}
```

来週の課題

- タイトルの表示
- ゲームオーバー画面、クリア画面の表示
- 複数のステージの作成