【実習２】　配列を使う　　ブロックとバーを表示する

１　ソースコード

（１）テンプレート　Jump の①の箇所に　Jump020View　を入力してください。

（２）次のアプリケーションを新規作成してください。

Jump010View をコピー＆ペーストして、ファイル名を「Jump020View」に変更してください。

プロジェクト名：JumpPro????(年組席)　　アプリケーション名：Jump020View

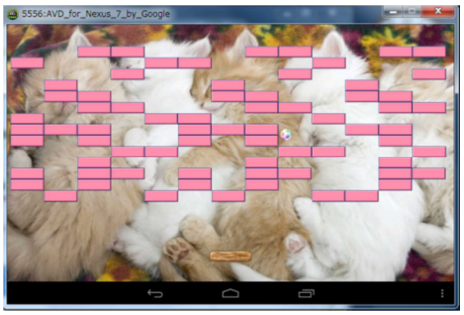
```
/*    年    組    席   名前
 * Jump020View
 *     ブロックとバーの表示（配列を使う）
 */
package jp.edu.mie;
・・・
public class Jump020View extends SurfaceView
                    implements SurfaceHolder.Callback
{
    ・・・
    private ScheduledExecutorService executor;

    //ブロック定数
    private final static int[] BLOCK={
        0,0,1,1,0,0,0,1,1,0,0,1,1,
        1,0,0,0,1,1,0,0,0,1,0,0,0,
        0,0,0,1,0,0,0,0,1,0,0,0,1,
        0,1,0,0,0,0,1,0,0,0,1,0,0,
        0,1,1,0,0,0,1,1,0,0,1,1,0,
        0,0,1,1,0,0,0,1,1,0,0,1,1,
        1,0,0,0,1,1,0,0,0,1,0,0,0,
        1,1,1,0,0,1,1,1,0,0,1,1,0,
        1,0,1,0,0,1,0,1,0,0,0,1,0,
        0,0,0,1,1,0,0,0,1,1,0,0,1,
        0,0,1,0,0,0,0,1,0,0,0,1,0,
        1,0,1,1,0,1,0,1,1,0,0,1,1,
        1,0,1,0,0,1,0,1,0,0,0,1,0,
        0,1,0,0,1,0,1,0,0,1,1,0,0,};

    private int[] block=new int[182];//ブロック
    private int barX=0;//バーX 座標
    private int barY;//バーY 座標
    private int blockWIDTH;//ブロックの幅
    private int blockHEIGHT;//ブロックの高さ
    private int barWIDTH;//バーの幅

    public Jump020View(Context context)
    {
        ・・・

        for(int i=0;i<4;i++){bmp[i]=readBitmap(context,"r"+i);}
```



```
        ballWIDTH=bmp[2].getWidth();
        ballHEIGHT=bmp[2].getHeight();
        //ブロックの読み込み
        for(int i=0;i<block.length;i++){block[i]=BLOCK[i];}
        blockWIDTH=bmp[1].getWidth();
        blockHEIGHT=bmp[1].getHeight();
        barWIDTH=bmp[3].getWidth();

        ・・・
    }
    //サーフェイス生成時に呼ばれる
    public void surfaceCreated(SurfaceHolder holder)
    {
        ・・・
        paint.setAntiAlias(true);//文字やラインを滑らかに見せる。
        barX=getWidth()/2-barWIDTH/2;//バーX 座標
        barY=getHeight()-100;//バーy 座標
        ballX=getWidth()/2-ballWIDTH/2;//ボール X 座標
        ballY=barY-ballHEIGHT/2;//ボール Y 座標

        ・・・
    }
    public void draw(Canvas canvas)
    {
        canvas=holder.lockCanvas();//Canvas オブジェクトをロックして取得
        canvas.drawBitmap(bmp[0],0,0,null);//背景の描画
        //ブロックの描画
        for(int i=0;i<BLOCK.length;i++)
        {
            if (block[i]!=0)
            {
                canvas.drawBitmap(bmp[1],10+(i%13)*blockWIDTH,
                                                    62+(i/13)*blockHEIGHT,null);
            }
        }
        canvas.drawBitmap(bmp[3],barX,barY,null);//バーの描画
        canvas.drawBitmap(bmp[2],ballX,ballY,null);//ボールの描画
        holder.unlockCanvasAndPost(canvas);//実画面に反映
    }

    ・・・

}
```

２　BLOCK配列

```
private final static int[] BLOCK={
        0,0,1,1,0,0,0,1,1,0,0,1,1,
        1,0,0,0,1,1,0,0,0,1,0,0,0,
        0,0,0,1,0,0,0,0,0,1,0,0,0,1,
        0,1,0,0,0,0,1,0,0,0,1,0,0,
        0,1,1,0,0,0,1,1,0,0,1,1,0,
        0,0,1,1,0,0,0,1,1,0,0,1,1,
        1,0,0,0,1,1,0,0,0,1,0,0,0,
        1,1,1,0,0,1,1,1,0,0,1,1,0,
        1,0,1,0,0,1,0,1,0,0,0,1,0,
        0,0,0,1,1,0,0,0,1,1,0,0,1,
        0,0,1,0,0,0,0,1,0,0,0,1,0,
        1,0,1,1,0,1,0,1,1,0,0,1,1,
        1,0,1,0,0,1,0,1,0,0,0,1,0,
        0,1,0,0,1,0,1,0,0,1,1,0,0,};
```

BLOCK 配列は、ゲームスタート時にブロックが存在するかどうかを示す情報です。１４行１３列で、０は存在しない、１は存在することを示します。

３　ブロックの表示

```
for (int i=0; i<block.length; i++) { block[i]=BLOCK[i]; }//①
    ・
    ・
for(int i=0;i<BLOCK.length;i++)//②
{
    if (block[i]!=0)
    {
        canvas.drawBitmap(bmp[1],10+(i%13)*blockWIDTH,
                                        62+(i/13)*blockHEIGHT,null);
    }
}
```

①BLOCK 配列の中身を block 配列にコピーします。

②block[i]が０でなかったら canvas.drawBitmap メソッドで次のとおり表示します。

※ボールがブロックに衝突すると、block 配列の要素に 0 が指定され、ブロックが消滅します。

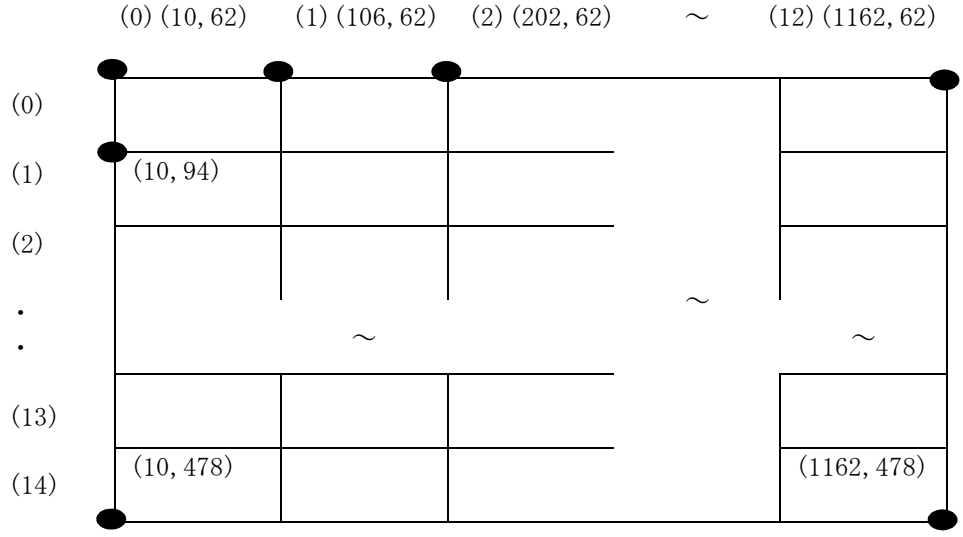
BLOCK 配列（i は 0～181 に変化する）

```
private final static int[] BLOCK={
        0,0,1,1,0,0,0,1,1,0,0,1,1,
        1,0,0,0,1,1,0,0,0,1,0,0,0,
        0,0,0,1,0,0,0,0,1,0,0,0,1,
        0,1,0,0,0,0,1,0,0,0,1,0,0,
        0,1,1,0,0,0,1,1,0,0,1,1,0,
        0,0,1,1,0,0,0,1,1,0,0,1,1,
        1,0,0,0,1,1,0,0,0,1,0,0,0,
        1,1,1,0,0,1,1,1,0,0,1,1,0,
        1,0,1,0,0,1,0,1,0,0,0,1,0,
        0,0,0,1,1,0,0,0,1,1,0,0,1,
        0,0,1,0,0,0,0,1,0,0,0,1,0,
        1,0,1,1,0,1,0,1,1,0,0,1,1,
        1,0,1,0,0,1,0,1,0,0,0,1,0,
        0,1,0,0,1,0,1,0,0,1,1,0,0,};
```

次のような一次元配列で保存されます。

(0)	(1)	(2)	(3)			～						(12)
0	0	1	1	0	0	0	1	1	0	0	1	1
1	0	0	0	1	1	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	1
0	1	0	0	0	0	1	0	0	0	1	0	0
0	1	1	0	0	0	1	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	0	0	1	1
1	0	0	0	1	1	0	0	0	1	0	0	0
1	1	1	0	0	1	1	1	0	0	1	1	0
1	0	1	0	0	1	0	1	0	0	0	1	0
0	0	0	1	1	0	0	0	1	1	0	0	1
0	0	1	0	0	0	0	1	0	0	0	1	0
1	0	1	1	0	1	0	1	1	0	0	1	1
1	0	1	0	0	1	0	1	0	0	0	1	0
0	1	0	0	1	0	1	0	0	1	1	0	0

さらに一部分を詳細に見てみると次のようになります。



※　canvas.drawBitmap(bmp[1],10+(i%13)*blockWIDTH,
62+(i/13)*blockHEIGHT,null);

10　…　ブロックの X 軸の開始点（任意の数値）
62　…　ブロックの Y 軸の開始点（任意の数値）
i%13　…　i÷13=商…あまり　13 は列数
i/13　…　13 は列数

４　バーとボールの表示

```
barX=getWidth()/2-barWIDTH/2;//バーX 座標
barY=getHeight()-100;//バーy 座標
ballX=getWidth()/2-ballWIDTH/2;//ボール X 座標
ballY=barY-ballHEIGHT/2;//ボール Y 座標

・・・

canvas.drawBitmap(bmp[3],barX,barY,null);//バーの描画
canvas.drawBitmap(bmp[2],ballX,ballY,null);//ボールの描画
```

drawBitmap メソッドで指定した XY 座標は左上の座標となります。