９　抽象クラス

9.1　クラスと比較したときの抽象クラスの特徴

①インスタンスを作れない。

②処理内容が定義されていないメソッドをもっています。

③ポリモーフィズムを活用できる。

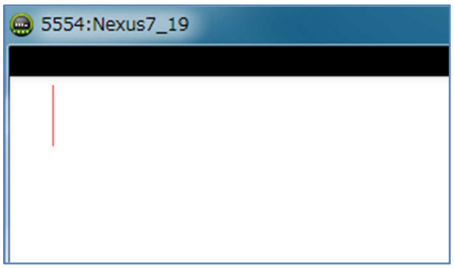
④大規模なプログラムを作成するときに役立つ。

9.2　線を描く

9.2.1　テンプレート(MainActivity.java)の①の箇所に　Main070View　を入力してください。

9.2.2　Main060View.java をコピーして Main070View.java　を新規作成してください。

ファイル名：　src/jp/edu/mie/ Main070View.java




```
package jp.edu.mie;
・・・
import android.view.View;
import android.graphics.Path;
import android.graphics.PointF;
import android.graphics.Rect;
import android.graphics.RectF;

public class Main070View extends View
{
    public Main070View(Context context)
    {
        ・・・
    }
    protected void onDraw(Canvas canvas)
    {
        Paint paint = new Paint();
        Shape[] shapes = new Shape[17];
        shapes[0] = new Line();

        for(int i = 0; i < shapes.length; i++)
        {
            if(shapes[i] == null){ break; }
            shapes[i].draw(canvas, paint);
        }
    }
}
abstract class Shape  //①
{
    abstract void draw(Canvas canvas, Paint paint); //②
}
//線
class Line extends Shape  //③
{
    void draw(Canvas canvas, Paint paint)
```

```
    {
        paint.setStrokeWidth(1);
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.argb(255, 255, 0, 0));
        canvas.drawLine(50, 10, 50, 1+80, paint);
    }
}
```

9.2.1 抽象クラスのソースリストの解説

①abstract class Shape

抽象クラス

abstract 修飾子を付けるとインスタンスを作れないクラスになります。

抽象クラスはポリモーフィズムの仕組みを活用できます。

②abstract void draw(Canvas canvas, Paint paint);

抽象メソッド

処理内容が定義されていないメソッドをもっています。

③class Line extends Shape

抽象クラスを継承したサブクラスは､抽象メソッドをオーバーライドしなければなりません。

9.2.2 色の指定

色を指定するには、Graphics クラスの setColor()メソッドを使います。指定した色はグラフィックスの描画関連メソッドが呼ばれるたびに使われます。

Paint クラスは描画のための色やスタイルの情報のクラスとなります。Paint クラス自体は色やスタイルの情報を持つだけで実際の描画は行いません。

文法

setColor(int color)

引数	説明
color	色情報を int 型で指定する。通常は int の数値をリテラルで指定するのは可読性に欠けるため、Color クラスを用いたりリソースを用いて指定する。

Color クラスでの色の指定は以下のパターンで行います。

引数	説明
Color#rgb(int red, int green, int blue)	red、green、blue を 0～255 の比率で指定する。
Color#argb(int alpha, int red, int green, int blue)	alpha、red、green、blue を 0～255 の比率で指定する。

Color#定数	定数には、BLACK（黒）、BLUE（青）、CYAN（空色）、DKGRAY（濃い灰色）、GRAY（灰色）、GREEN（緑）、LTGRAY（薄い灰色）、MAGENTA（ピンク）、RED（赤）、TRANSPARENT（透明）、WHITE（白）、YELLOW（黄色）等がある。

setColor(Color.argb(int 透明度, int 赤の値, int 緑の値, int 青の値));

1番目の引数に透明度（アルファ）を表す数値を指定します。数値は0から255までの数値で指定し、255を指定したときが不透明、0を指定した時が透明となります。

2番目から4番目の引数には「rgb」メソッドと同じく赤緑青を表す数値をそれぞれ0から255までの数値で指定します。結果として色を表すint型の値を返します。

例　①白色の場合

```
paint.setColor(Color.argb(255,255,255,255))
paint.setColor(Color.WHITE)
```

②黒色の場合

```
paint.setColor(Color.argb(255,0,0,0))
paint.setColor(Color.BLACK)
```

9.2.3　描画スタイル

直線、弧、矩形および塗りつぶされた矩形は setStyle() メソッドによって指定された 描画スタイルで描画されます。

文法

setStyle(Paint.Style style)

style の値

（1）Paint.style.FILL　：図形の内部を塗りつぶす。

（2）Paint.style.FILL_AND_STROKE　：図形の内部を塗りつぶし、輪郭線のみ描画する。

（3）Paint.style.STROKE　：図形の輪郭線のみ描画する。

9.2.4　線を描く

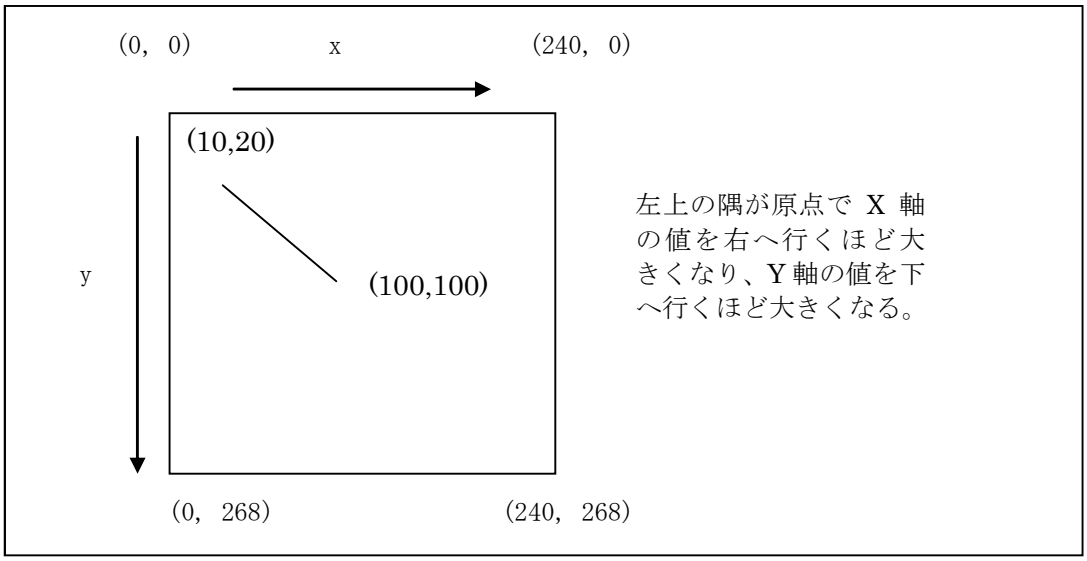
現在設定されている色と描画スタイルを使用して、座標（x1, y1）と座標（x2, y2）の間に直線を描画します。

文法

drawLine(int 始点のx座標(x1), int 始点のy座標(y1),
int 終点のx座標(x2), int 終点のy座標(y2), Paint paint);

例　(10,20)から(100,100)へ実線を引く

```
Paint.setStrokeWidth(1);//ライン幅
Paint.setStyle(Paint.Style.STROKE);
Paint.drawLine(10,20,100,100,paint);
```



9.3 パスの描画

Main070View.java に次のソースを追加してください。(網掛け部分)

ファイル名： src/jp/edu/mie/ Main070View.java

```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[0] = new Line();
        shapes[1] = new PathLine();
        ・・・
    }
}
・・・
//線
class Line extends Shape
{
    ・・・
}
//折れ線
class PathLine extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.argb(255, 255, 0, 0));
        Path path=new Path();
        path.moveTo(110+ 0, 10+ 0);
        path.lineTo(110+60, 20+10);
        path.lineTo(110+20, 20+30);
        path.lineTo(110+80, 20+50);
```

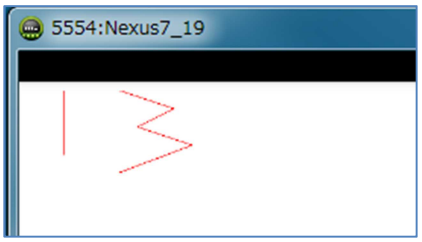
```
            path.lineTo(110+ 0, 20+80);
            canvas.drawPath(path,paint);
        }
}
```

9.3.1　パスの描画の解説

Path オブジェクトを使うと、複数の描画処理を Path オブジェクトに追加して、まとめて描画する事ができます。

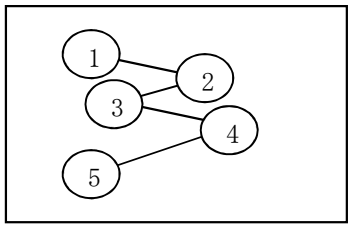
```
drawPath(Path path, Paint paint)
```

Path クラスは、複数の頂点座標（xy 座標）を保持するパス情報のクラスです。moveTo()メソッドでパスの開始座標を指定し、lineTo()メソッドで終点からのラインを追加します。




```
paint.setStyle(Paint.Style.STROKE);
paint.setColor(Color.argb(255, 255, 0, 0));
Path path=new Path();
①path.moveTo(110+ 0, 10+ 0);
②path.lineTo(110+60, 10+10);
③path.lineTo(110+20, 10+40);
④path.lineTo(110+80, 10+50);
⑤path.lineTo(110+ 0, 10+80);
canvas.drawPath(path,paint);
```

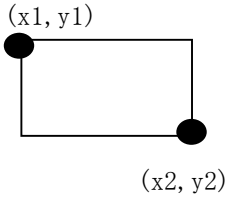
①から⑤の座標



9.4　四角形を描く・塗りつぶす

Main070View.java に次のソースを追加してください。(網掛け部分)

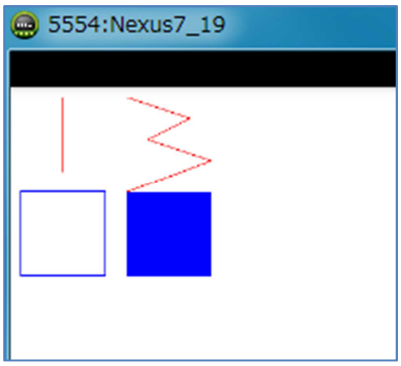
ファイル名：　src/jp/edu/mie/ Main070View.java




```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[1] = new PathLine();
        shapes[2] = new Rectangle();
        shapes[3] = new RectangleFill();
        ・・・
    }
}
・・・
//折れ線
class PathLine extends Shape
{
    ・・・
}
//四角形
class Rectangle extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.argb(255, 0, 0, 255));
        canvas.drawRect(new Rect(10+0, 100+0, 10+80, 100+80), paint);
    }
}
//四角形の塗りつぶし
class RectangleFill extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.FILL);
        paint.setColor(Color.argb(255, 0, 0, 255));
        canvas.drawRect(new Rect(110+0, 100+0, 110+80, 100+80), paint);
    }
}
```

9.4.1　四角形を描く・塗りつぶすの解説

現在の色および描画スタイルを使用して、指定された矩形のアウトラインや塗りつぶしを描画します。



文法

（１）四角の左上の座標を(x1,y1)右下の座標を(x2,y2)とすると、

```
drawRect( int x1, int y1, int x2, int y2, Paint)

drawRect( float x1, float y1, float x2, float y2, Paint)
```

※x1=bottom　x2=left　y1=right　y2=top で置き換わります。

（２）Rect インスタンスに設定して

```
drawRect( Rec, Paint )
```

（３）RectF インスタンスに設定して

```
drawRect( RectF, Paint )
```

座標位置の指定には、int 型と float 型の値および、int 型と float 型の配列が使われますが、座標を指定するためのクラスも用意されています。

Rect クラスと RectF クラスには、どちらも Public なインスタンス変数 bottom,left,right,top が定義されていて、それぞれ右上隅と左下隅の x 座標と y 座標を示しています。

Rect クラスと RectF クラスの違いは、Rect クラスがインスタンス変数 bottom,left,right,top の型が int 型であるのに対して、RectF クラスでは float 型である事です。

四角形を描く

```
paint.setStyle(Paint.Style.STROKE);

paint.setColor(Color.argb(255, 0, 0, 255));

canvas.drawRect(new Rect(10+0, 100+0, 10+80, 100+80), paint);
```

四角形の塗りつぶし

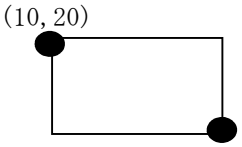
```
paint.setStyle(Paint.Style.FILL);

paint.setColor(Color.argb(255, 0, 0, 255));

canvas.drawRect(new Rect(110+0, 100+0, 110+80, 100+80), paint);
```

例１　座標(10,20)から高さ 100 で幅 100 の正方形を描く

```
canvas.drawRect(new Rect(10, 20, 110, 120),paint);
```



例２　赤い輪郭線の矩形を描く

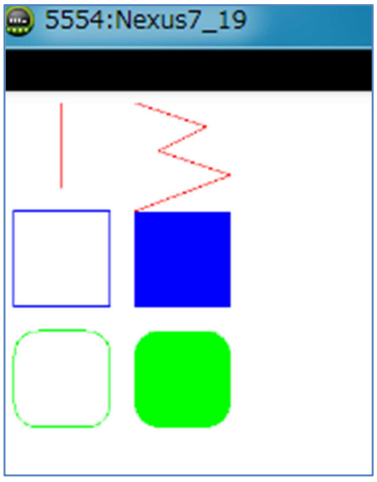
```
paint.setStyle(Paint.Style.STROKE);
paint.setColor(Color.argb(255, 255, 0, 0));
canvas.drawRect(10, 20, 110, 120, paint);
```

例３　赤く塗りつぶされた矩形を描く

```
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.argb(255, 255, 0, 0));
canvas.drawRect(10, 20, 110, 120, paint);
```

9.5　角丸矩形を描く・塗りつぶす

Main070View.java に次のソースを追加してください。(網掛け部分)



ファイル名：　src/jp/edu/mie/ Main070View.java

```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[2] = new Rectangle();
        shapes[3] = new RectangleFill();
        shapes[4] = new RoundedRectangle();
        shapes[5] = new RoundedRectangleFill();
        ・・・
    }
}
    ・・・
//四角形の塗りつぶし
class RectangleFill extends Shape
{
    ・・・
}
//角丸四角形
class RoundedRectangle extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.argb(255, 0, 255, 0));
        canvas.drawRoundRect(new RectF(10+0,
                                200+0, 10+80, 200+80), 20, 20, paint);
    }
}
//角丸四角形の塗りつぶし
class RoundedRectangleFill extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.FILL);
        paint.setColor(Color.argb(255, 0, 255, 0));
        canvas.drawRoundRect(new RectF(110+0,
                                200+0, 110+80, 200+80), 20, 20, paint);
    }
}
```

9.5.1　角丸矩形を描く・塗りつぶすの解説

現在の色およびストローク・スタイルを使用して、指定された角が丸い矩形のアウトラインを描画します。

```
drawRoundRect(RectF rectf, float コーナーの x 半径,
                                    float コーナーの y 半径,Paint paint) ;
```

角丸矩形の描画

```
paint.setStyle(Paint.Style.STROKE);
paint.setColor(Color.argb(255, 0, 255, 0));
canvas.drawRoundRect(new RectF(10+0, 200+0,
        10+80, 200+80), 20, 20, paint);
```

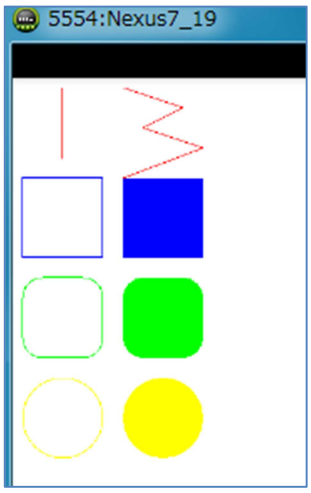
角丸矩形の塗りつぶし

```
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.argb(255, 0, 255, 0));
canvas.drawRoundRect(new RectF(110+0, 200+0,
        110+80, 200+80), 20, 20, paint);
```

9.6　円を描く・塗りつぶす

Main070View.java に次のソースを追加してください。(網掛け部分)

ファイル名：　src/jp/edu/mie/ Main070View.java




```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[4] = new RoundedRectangle();
        shapes[5] = new RoundedRectangleFill();
        shapes[6] = new Circle();
        shapes[7] = new CircleFill();
        ・・・
    }
}
    ・・・
//角丸四角形の塗りつぶし
class RoundedRectangleFill extends Shape
{
    ・・・
}
//円
class Circle extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
```

```
            paint.setStyle(Paint.Style.STROKE);
            paint.setColor(Color.argb(255, 255, 255, 0));
            canvas.drawCircle(50, 340, 40, paint);
        }
    }
    //円の塗りつぶし
    class CircleFill extends Shape
    {
        void draw(Canvas canvas, Paint paint)
        {
            paint.setStyle(Paint.Style.FILL);
            paint.setColor(Color.argb(255, 255, 255, 0));
            canvas.drawCircle(150, 340, 40, paint);
        }
    }
```

9.6.1 円を描く・塗りつぶすの解説

drawCircle(float 中心点のｘ座標, float 中心点のｙ座標, float　半径, Paint paint)

現在の色およびストローク・スタイルを使用して、指定された矩形をカバーする円形のアウトラインを描画します。

例１　Ｘ座標：５０　ｙ座標：３４０　半径：４０　の円を描く

```
//■円の描画
paint.setStyle(Paint.Style.STROKE);
paint.setColor(Color.argb(255, 255, 255, 0));
canvas.drawCircle(50, 340, 40, paint);
```

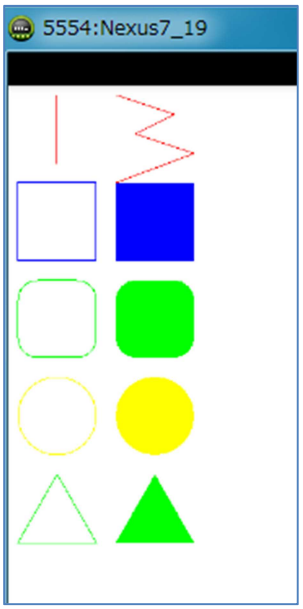
例２　Ｘ座標：１５０　ｙ座標：３４０　半径：４０　の円を塗り潰す

```
//■円の塗りつぶし
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.argb(255, 255, 255, 0));
canvas.drawCircle(150, 340, 40, paint);
```

9.7　三角形の描画

Main070View.java に次のソースを追加してください。(網掛け部分)

ファイル名：　src/jp/edu/mie/ Main070View.java

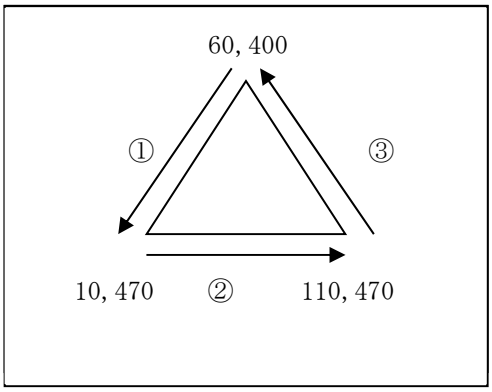



```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[6] = new Circle();
        shapes[7] = new CircleFill();
        shapes[8] = new Triangle();
        shapes[9] = new TriangleFill();
        ・・・
    }
}
    ・・・
//円の塗りつぶし
class CircleFill extends Shape
{
    ・・・
}
//三角形
class Triangle extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.GREEN);
        Path path = new Path();
        path.moveTo(50f, 400f);
        path.lineTo(10f, 470f);
        path.lineTo(90f, 470f);
        path.lineTo(50f, 400f);
        canvas.drawPath(path, paint);
    }
}
//三角形の塗りつぶし
class TriangleFill extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.FILL);
        paint.setColor(Color.GREEN);
        Path path = new Path();
        path.moveTo(150f, 400f);
        path.lineTo(110f, 470f);
```

```
            path.lineTo(190f, 470f);
            path.lineTo(150f, 400f);
            canvas.drawPath(path, paint);
        }
    }
```

9.7.1 三角形の描画の解説

```
paint.setColor(Color.GREEN);
paint.setStyle(Paint.Style.STROKE);
Path path1 = new Path();
path1.moveTo(60f, 400f);
①path1.lineTo(10f, 470f);
②path1.lineTo(110f, 470f);
③path1.lineTo(60f, 400f);
canvas.drawPath(path1, paint);
```

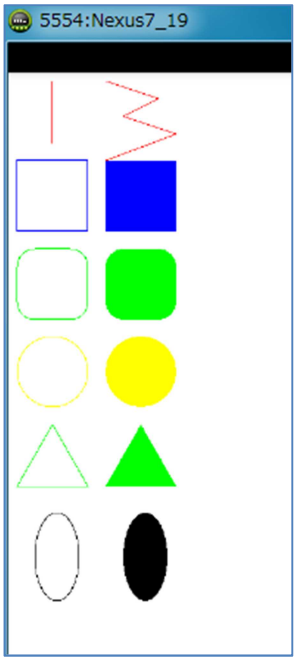


9.8 楕円の描画

Main070View.java に次のソースを追加してください。(網掛け部分)

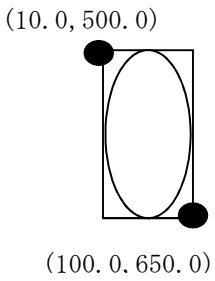
ファイル名： src/jp/edu/mie/ Main070View.java

```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[8] = new Triangle();
        shapes[9] = new TriangleFill();
        shapes[10] = new Oval();
        shapes[11] = new OvalFill();
        ・・・
    }
}
    ・・・
//三角形の塗りつぶし
class TriangleFill extends Shape
{
    ・・・
}
//楕円
class Oval extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.BLACK);
```



```
            canvas.drawOval(new RectF(30.0f, 500.0f, 80.0f, 600.0f), paint);
        }
    }
    //楕円の塗りつぶし
    class OvalFill extends Shape
    {
        void draw(Canvas canvas, Paint paint)
        {
            paint.setStyle(Paint.Style.FILL);
            paint.setColor(Color.BLACK);
            canvas.drawOval(new RectF(30.0f+100, 500.0f,
                                                    80.0f+100, 600.0f), paint);
        }
    }
```

9.8.1　楕円の描画の解説



外接矩形を描いて表現します。

文法

drawOval(RectF oval, Paint paint)

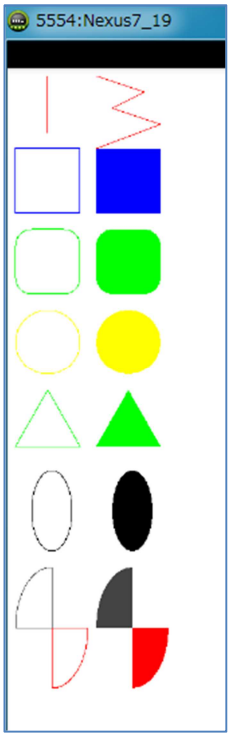
例

```
//■楕円の描画
paint.setColor(Color.BLACK);
paint.setStyle(Paint.Style.STROKE);
canvas.drawOval(new RectF(10.0f, 500.0f, 100.0f, 650.0f), paint);
```

9.9　円弧の描画

Main070View.java に次のソースを追加してください。(網掛け部分)

ファイル名：　src/jp/edu/mie/ Main070View.java




```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[10] = new Oval();
        shapes[11] = new OvalFill();
        shapes[12] = new Arc();
        shapes[13] = new ArcFill();
    }
}
    ・・・
//楕円の塗りつぶし
class OvalFill extends Shape
{
    ・・・
}
//円弧
class Arc extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.DKGRAY);
        canvas.drawArc(new RectF(10.0f, 620.0f,
                                 100.0f, 770.0f), 180, 90, true, paint);
        paint.setColor(Color.RED);
        canvas.drawArc(new RectF(10.0f, 620.0f,
                                 100.0f, 770.0f), 0, 90, true, paint);
    }
}
//円弧の塗りつぶし
class ArcFill extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.FILL);
        paint.setColor(Color.DKGRAY);
        canvas.drawArc(new RectF(110.0f, 620.0f,
                                 200.0f, 770.0f), 180, 90, true, paint);
        paint.setColor(Color.RED);
        canvas.drawArc(new RectF(110.0f, 620.0f,
                                 200.0f, 770.0f), 0, 90, true, paint);
```

```
    }
}
```

9.9.1 円弧の描画

RectF クラスで指定された矩形領域に、角度 startAngle を起点に時計回りに、角度 sweepAngle 幅の円弧を引きます。useCenter を true に指定すると、楕円の中心を含めて描画します。

弧の中心は、矩形領域の中心となります。

startAngle が 0 で時計針の短針が 3 時の位置に、90 で 6 時の位置、180 で 9 時の位置が開始位置になります。

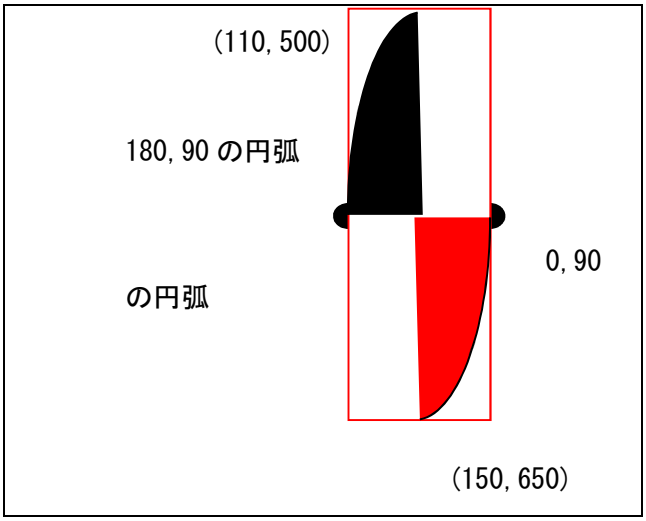
sweepAngle の値に負の値を指定すると、反時計回りの角度になります

文法

```
drawArc(RectF oval, float startAngle, float sweepAngle,
                boolean useCenter, Paint paint)
```

例

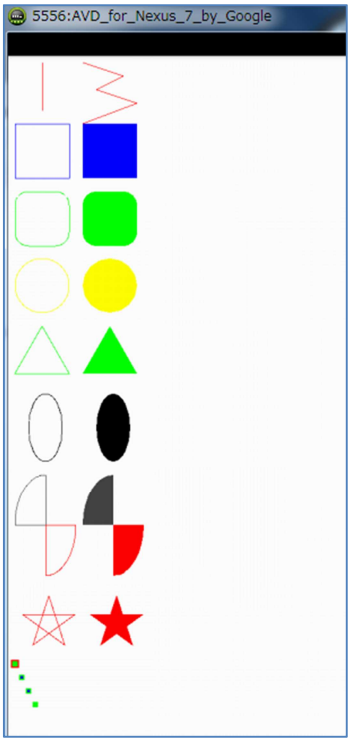
```
//■円弧の描画
        paint.setColor(Color.DKGRAY);
        paint.setStyle(Paint.Style.FILL);
        canvas.drawArc(new RectF(110.0f, 500.0f, 150.0f, 650.0f),
                                                    180, 90, true, paint);
        paint.setColor(Color.RED);
        canvas.drawArc(new RectF(110.0f, 500.0f, 150.0f, 650.0f),
                                                    0, 90, true, paint);
```



9.10　点の描画

Main070View.java に次のソースを追加してください。(網掛け部分)

ファイル名：　src/jp/edu/mie/ Main070View.java




```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[12] = new Arc();
        shapes[13] = new ArcFill();
        shapes[14] = new Point();
    }
}
    ・・・
//円弧の塗りつぶし
class ArcFill extends Shape
{
    ・・・
}
//点
class Point extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        float[] pts = { 10, 800, 20, 820, 30, 840, 40, 860 };
        paint.setStrokeWidth(12);
        canvas.drawPoint(pts[0], pts[1], paint);
        paint.setColor(Color.GREEN);
        paint.setStrokeWidth(8);
        canvas.drawPoints(pts, paint);
        paint.setColor(Color.BLUE);
        paint.setStrokeWidth(4);
        canvas.drawPoints(pts, 2, 4, paint);
    }
}
```

9.10.1　点の描画の解説

点を描画する Canvas クラスのメソッドには、次のものがあります。

```
void drawPoint(float x, float y, Paint paint)
```

ｘ軸とｙ軸の座標を指定して、点を描画します 。

```
void drawPoints(float[] pts, Paint paint)
```

ｘ軸とｙ軸の座標のペアの配列を指定して、点を描画します。

```
void drawPoints(float[] pts, int offset, int count, Paint paint)
```

ｘ軸とｙ軸の座標のペアの配列と、描画対象となる配列のオフセット位置と、オフセット位置からの描画する点の数を指定して、点を描画します。

引数 count には、描画する点の数ｘ２の値を指定します。

次のプログラムでは、

（１）３行目～５行目で、座標（10,800）の位置に点の大きさ１２の赤色の点

（２）７行目～９行目で、座標（10,800），(20,820)，(30,840)，(40,860）の位置に大きさ８の緑色の４つの点

（３）１１行目～１３行目で、座標（20,690)，(30,710）の位置に大きさ４の青色の２つの点（配列の オフセット位置２からの４つの値（２個の点））

を描画しています。

```
01  float[] pts = { 10, 800, 20, 820, 30, 840, 40, 860 };
02  //■ drawPoint(float x, float y, Paint paint)の例
03  paint.setColor(Color.RED);
04  paint.setStrokeWidth(12);
05  canvas.drawPoint(pts[0], pts[1], paint);
06  //■ drawPoint(float x, float y, Paint paint)の例
07  paint.setColor(Color.GREEN);
08  paint.setStrokeWidth(8);
09  canvas.drawPoints(pts, paint);
10  //■ drawPoints(float[] pts, int offset, int count, Paint paint)の例
11  paint.setColor(Color.BLUE);
12  paint.setStrokeWidth(4);
13  canvas.drawPoints(pts, 2, 4, paint);
```

9.11 星の描画

Main070View.java に次のソースを追加してください。(網掛け部分)

ファイル名： src/jp/edu/mie/ Main070View.java

```
package jp.edu.mie;
・・・
public class Main070View extends View
{
    ・・・
    protected void onDraw(Canvas canvas)
    {
        ・・・
        shapes[14] = new Point();
        shapes[15] = new Star();
        shapes[16] = new StarFill();
    }
}
    ・・・
//点
class Point extends Shape
{
    ・・・
}
//星
class Star extends Shape
{
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.RED);
        Path path = new Path();
        float theta = (float)(Math.PI * 72 / 180);
        float r = 40f;
        PointF center = new PointF(60f, 940f);
        float dx1 = (float)(r*Math.sin(theta));
        float dx2 = (float)(r*Math.sin(2*theta));
        float dy1 = (float)(r*Math.cos(theta));
        float dy2 = (float)(r*Math.cos(2*theta));
        path.moveTo(center.x, center.y-r);
        path.lineTo(center.x-dx2, center.y-dy2);
        path.lineTo(center.x+dx1, center.y-dy1);
        path.lineTo(center.x-dx1, center.y-dy1);
        path.lineTo(center.x+dx2, center.y-dy2);
        path.lineTo(center.x, center.y-r);
        canvas.drawPath(path, paint);
    }
}
//星の塗りつぶし
class StarFill extends Shape
{
```

```
    void draw(Canvas canvas, Paint paint)
    {
        paint.setStyle(Paint.Style.FILL_AND_STROKE);
        Path path = new Path();
        paint.setColor(Color.RED);
        float theta = (float)(Math.PI * 72 / 180);
        float r = 40f;
        PointF center = new PointF(160f, 940f);
        float dx1 = (float)(r*Math.sin(theta));
        float dx2 = (float)(r*Math.sin(2*theta));
        float dy1 = (float)(r*Math.cos(theta));
        float dy2 = (float)(r*Math.cos(2*theta));
        path.moveTo(center.x, center.y-r);
        path.lineTo(center.x-dx2, center.y-dy2);
        path.lineTo(center.x+dx1, center.y-dy1);
        path.lineTo(center.x-dx1, center.y-dy1);
        path.lineTo(center.x+dx2, center.y-dy2);
        path.lineTo(center.x, center.y-r);
        canvas.drawPath(path, paint);
    }
}
```

9.12　画面の幅と高さを取得する

画面の幅と高さを取得するには、Canvas クラスの getWidth()メソッドと getHeight()メソッドを使います。

canvas.drawRect(new Rect(0, 0, getWidth(),getHeight()), paint);は
canvas.drawRect(new Rect(0, 0, 480, 854), paint)と同じ意味です。

9.13　前に表示された文字や絵を消す

画面いっぱいに白い長方形を描くと、画面の表示をクリアしたことになります。次の例は、画面をクリアして車を表示したソースです。

```
//画面いっぱいに白い長方形を描く
        canvas.drawColor(Color.WHITE);//■背景を塗りつぶす
        canvas.drawBitmap(carImage,x,y,paint)
```

【演習３】テーマを決めて、デザイン画を作成してください。

・プロジェクト名：kadai3Pro????(年組席)

・アプリケーション名：kadai3View????（????：年組席）

・画面にテーマとなる文字を表示して、そのテーマに合ったデザインを作成してください。

・ソースリストを提出してください。

・相互評価項目（各 10 点）

①全体のデザイン性（色調等）、②図形のこだわり、③背景のこだわり

④組立てのこだわり(テーマ性)、⑤技術レベル

・ソースの中に課題番号と組席名前を明記してください。

■生徒の作品例