가천대학교
Gachon University

학생 여러분 반갑습니다.

다른 친구들이 입장할 때까지 조금 기다려 주십시오.

곧 모바일 프로그래밍 수업을 시작합니다.

음소거( )가 되었는지 확인 바랍니다.

모바일 프로그래밍
화목(1,2교시)/ 화목(3,4교시)
정윤현 (AI/소프트웨어학부)

Mobile Programming

Android Programming

Chap 6. Graphics & Animation

Prof. Younhyun Jung

Email) younhyun.jung@gachon.ac.kr

가천대학교 소프트웨어학과
Gachon University

Graphics in Android

- Android offers a custom 2D graphics library for drawing and animating geometries and images.
  - android.graphics.drawable package
    - the common classes used for drawing in two-dimensions.
  - android.view.animation package
    - the common classes used for animating in two-dimensions.

NOTE :
- 3D graphics (e.g., OpenGL) requires a lot of knowledge of the underlying graphics libraries.
- We will look at a simple form of 2D graphics

Canvas

- Sometimes your app has unique needs that aren't covered by the built-in views. This lecture shows you how to create your own views.
  - If you would like to perform specialized drawing and/or control the animation of graphics, you should do so by drawing through a Canvas.

- Reference:

http://developer.android.com/training/custom-views/index.html

Canvas

- To draw the screen, you need a valid **canvas**.
  - Typically, you get a canvas by extending View class
  - Implementing the onDraw() method

```
protected void onDraw(Canvas canvas) { … }
```

- Canvas holds the “draw” calls for a rectangle of space.

  Drawing Methods on a canvas
  - **Drawing** images, text, geometries and so on..
- Also, provide methods
  - **Obtaining/Updating** information on images, areas,…

  (getHeight(), getWidth())

Canvas

- Basic Draw Methods
  - void drawPoint (float x, float y, Paint paint)
  - void drawLine (float startX, float startY, float stopX, float stopY, Paint paint)
  - void drawCircle (float cx, float cy, float radius, Paint paint)
  - void drawRect (float left, float top, float right, float bottom, Paint paint)
  - void drawText (String text, float x, float y, Paint paint)
  - void drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)
  - …

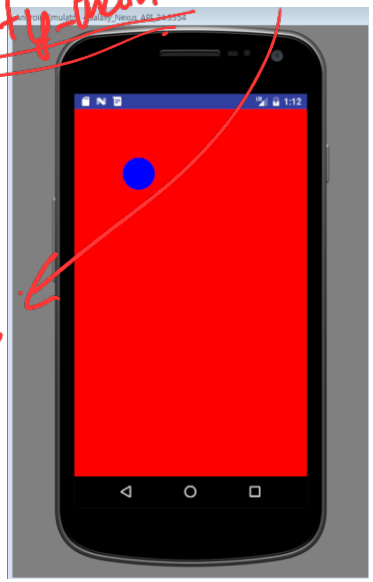
- Typically, a Paint object is placed in the last argument
- Reference:

http://developer.android.com/reference/android/graphics/Canvas.html

Example : Custom View

- Let's start with an example :

```
public class CustomView extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MyView vw = new MyView(this);
        setContentView(vw);
    }

    protected class MyView extends View {
        public MyView(Context context){
            super(context);
        }

        @Override
        protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);
            Paint pnt = new Paint();
            pnt.setColor(Color.BLUE);
            canvas.drawColor(Color.RED);
            canvas.drawCircle(200, 200, 50, pnt);
                                중심축좌표 반지름
        }
    }
}
```

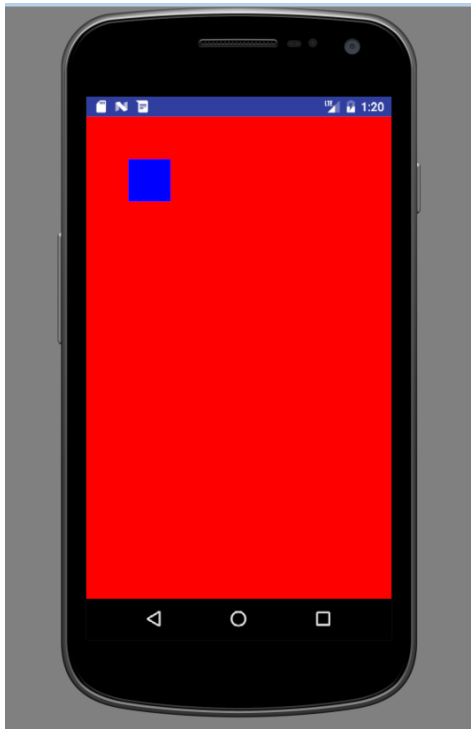


You can use this View like any other layout;
by using **SetContentView()** method

Example : Custom View

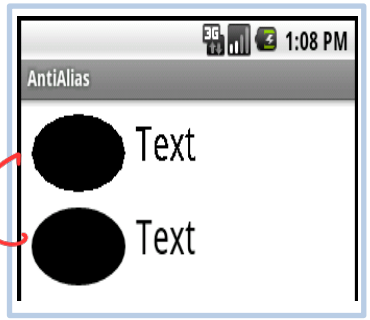
- Another example:

```
public class CustomView extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MyView vw = new MyView(this);
        setContentView(vw);
    }

    protected class MyView extends View {
        public MyView(Context context){
            super(context);
        }

        @Override
        protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);
            Paint pnt = new Paint();
            pnt.setColor(Color.BLUE);
            canvas.drawColor(Color.RED);
            canvas.drawRect(100, 100, 200, 200, pnt);

        }
    }
}
```



Paint : Methods

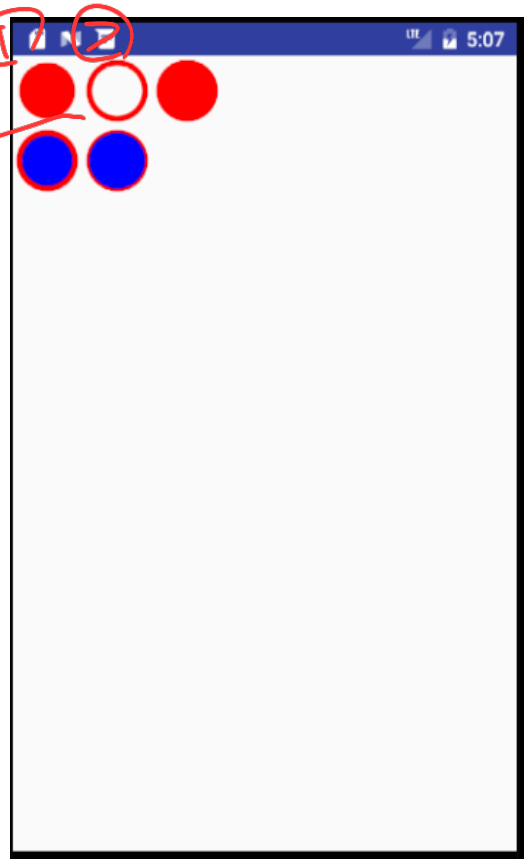
- Paint (android.graphics.Paint)
  - Store far more than a color !
  - Encapsulate the style, complex color etc..
- The Paint class
  - holds the style and color information about how to draw geometries, text and bitmaps.
- Methods.. (there are a large number of methods..)
  - setAntiAlias(boolean aa): Paint의 경계면을 부드럽게 처리할지 설정
  - setColor(int color) : Paint의 색상을 설정
  - setStrokeWidth(float width)
    - Paint의 굵기를 설정 합니다.
  - setStyle(Paint.Style style) : Paint 스타일을 설정 합니다.
    - FILL : 색상이 채워지고 테두리는 그려지지 않습니다.
    - FILL_AND_STROKE : 채우기와 테두리가 모두 그려집니다
    - STROKE : 채우기 없이 테두리만 그려집니다.



Exercise

- Do more! Modify source code as follows:

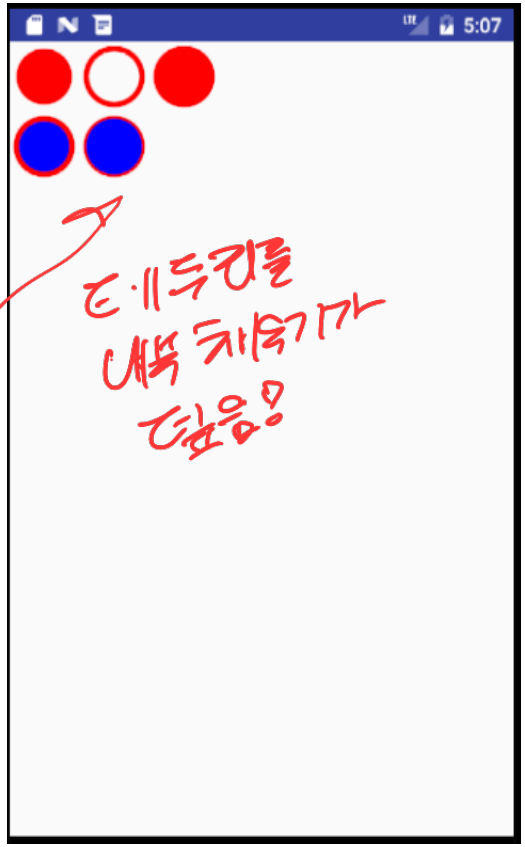
```
public void onDraw(Canvas canvas) {
    Paint Pnt = new Paint(Paint.ANTI_ALIAS_FLAG);

    Pnt.setStrokeWidth(8);
    Pnt.setColor(Color.RED);

    // 채우기
    Pnt.setStyle(Paint.Style.FILL);
    canvas.drawCircle(50, 50, 40, Pnt);

    // 외곽선그리기

    Pnt.setStyle(Paint.Style.STROKE);
    canvas.drawCircle(150, 50, 40, Pnt);

    // 외곽선및채우기

    Pnt.setStyle(Paint.Style.FILL_AND_STROKE);
    canvas.drawCircle(250, 50, 40, Pnt);
```



Exercise

- Do more! Modify source code as follows:

```
        // 파란색으로채우고빨간색으로외곽선그리기
        Pnt.setColor(Color.BLUE);
        Pnt.setStyle(Paint.Style.FILL);
        canvas.drawCircle(50, 150, 40, Pnt);
        Pnt.setColor(Color.RED);
        Pnt.setStyle(Paint.Style.STROKE);
        canvas.drawCircle(50, 150, 40, Pnt);

        // 빨간색으로외곽선그리고파란색으로채우기
        Pnt.setColor(Color.RED);
        Pnt.setStyle(Paint.Style.STROKE);
        canvas.drawCircle(150, 150, 40, Pnt);
        Pnt.setColor(Color.BLUE);
        Pnt.setStyle(Paint.Style.FILL);
        canvas.drawCircle(150, 150, 40, Pnt);
    }
  }
}
```



Paint : Methods

- Methods
  - setStrokeCap(Paint.Cap cap) : 선의 끝나는 지점의 장식을 설정
    - BUTT : 그 정해진 위치에서 끝납니다.
    - ROUND : 둥근 모양으로 끝이 장식됩니다.
    - SQUARE : 사각형 모양이며, 해당 좌표보다 조금 더 길게 그려 집니다.
  - setStrokeJoin(Paint.Join join) : 선의 끝 모양을 설정
    - MITER : 모서리를 각진 모양으로 만듭니다.
    - BEVEL : 모서리가 둥글게 살짝 깍인 모양으로 만듭니다.
    - ROUND : 모서리를 둥근 모양으로 만듭니다.

Exercise

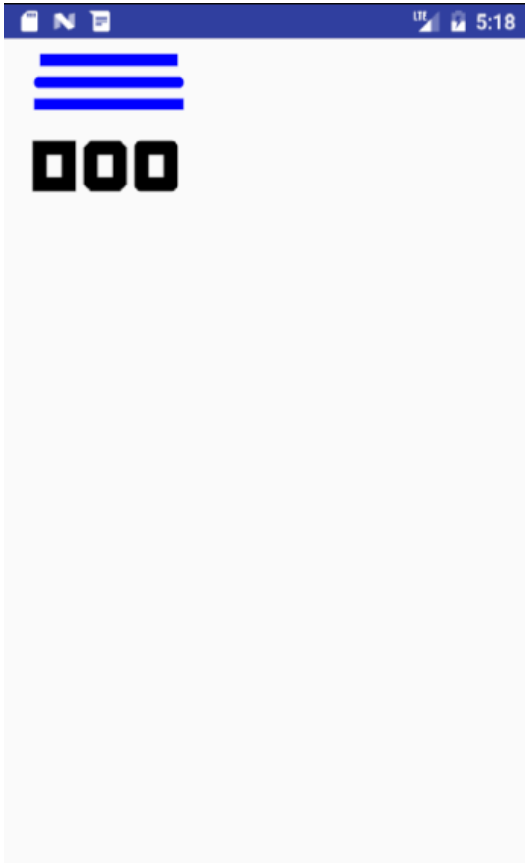
- Do more! Modify source code as follows:

```
public void onDraw(Canvas canvas) {
    Paint Pnt = new Paint(Paint.ANTI_ALIAS_FLAG);

    // 캡모양테스트

    Pnt.setColor(Color.BLUE);
    Pnt.setStrokeWidth(16);
    canvas.drawLine(50, 30, 240, 30, Pnt);
    Pnt.setStrokeCap(Paint.Cap.ROUND);
    canvas.drawLine(50, 60, 240, 60, Pnt);
    Pnt.setStrokeCap(Paint.Cap.SQUARE);
    canvas.drawLine(50, 90, 240, 90, Pnt);
```

Exercise

- Do more! Modify source code as follows:

```
    // 조인모양테스트

          Pnt.setColor(Color.BLACK);
           Pnt.setStrokeWidth(20);
           Pnt.setStyle(Paint.Style.STROKE);
           Pnt.setStrokeJoin(Paint.Join.MITER);
           canvas.drawRect(50, 150, 90, 200, Pnt);
           Pnt.setStrokeJoin(Paint.Join.BEVEL);
           canvas.drawRect(120, 150, 160, 200, Pnt);
           Pnt.setStrokeJoin(Paint.Join.ROUND);
           canvas.drawRect(190, 150, 230, 200, Pnt);

       }
   }
}
```



Path

- The Path class encapsulates (very useful tool)
  - compound (multiple contour) geometric paths consisting of straight line segments, quadratic curves, and cubic curves.
  - It can be drawn with **canvas.drawPath(path, paint)**, either filled or stroked (based on the paint's Style), or it can be used to draw text on a path.

```
moveTo(float x, float y)
lineTo(float x, float y)
addCircle(float x, float y, float radius, Path.Direction dir)
addRect(RectF rect, Path.Direction dir)
```

한번에 그리기 때문에 효과적.

Exercise

```
public void onDraw(Canvas canvas) {
    Path path = new Path();
    canvas.drawColor(Color.WHITE);

    Paint Pnt = new Paint();
    Pnt.setStrokeWidth(5);
    Pnt.setColor(Color.RED);
    Pnt.setStyle(Paint.Style.STROKE);

    // 원, 사각형을 패스로 정의한 후 출력
    path.addRect(100, 00, 150, 90, Path.Direction.CW);
    path.addCircle(50, 50, 40, Path.Direction.CW);
    canvas.drawPath(path, Pnt);

    // 패스로 정의한 후 출력
    path.reset();
    path.moveTo(10, 110);
    path.lineTo(50, 150);
    path.lineTo(400, 10);
    Pnt.setStrokeWidth(3);
    Pnt.setColor(Color.BLUE);
    canvas.drawPath(path, Pnt);
```

이전 정의 삭제.

Exercise

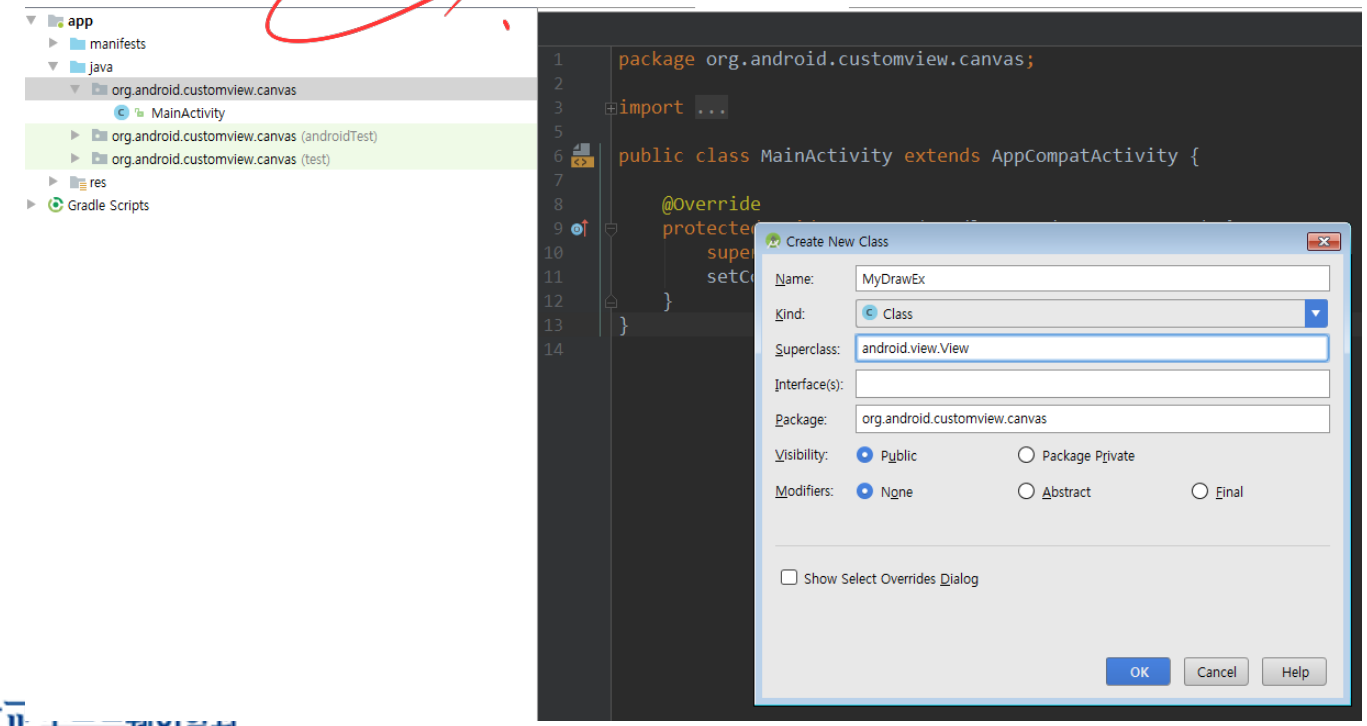

```
        // 패스 위에 텍스트 출력
        Pnt.setTextSize(20);
        Pnt.setStyle(Paint.Style.FILL);
        canvas.drawTextOnPath("Text on Path.", path, 0, 0, Pnt);
    }


    }
}
```

XML Layout으로 사용하기?

Making a custom view –Exercise

- 1. New Project
  - **Canvas**
- 2. Add a Class
- **New class** → MyDrawEx.java

개발 파일을 분리.



- **New class** → MyDrawEx.java

```
package org.android.customview.canvas;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

/**
 * Created by Administrator on 2018-04-23.
 */

public class MyDrawEx extends View {
    public MyDrawEx(Context c) {
        super(c);
    }
    public MyDrawEx(Context c, AttributeSet a) {
        super(c, a);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint = new Paint();
        paint.setColor(Color.RED);

        canvas.drawColor(Color.WHITE);
        for (int x=0; x<200; x+=5) {
            canvas.drawLine(x, 0, 200-x, 100, paint);
        }

    }
}
```

xml 에서 layout으로 사용하기 위해서는 생성자 두 개 필요

void drawLine (float startX, float startY, float stopX, float stopY, Paint paint)

Modify activity_main.xml layout as follows

- Add your MyView element to the XML, like so:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"    >

    <org.android.customview.canvas.MyDrawEx
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Drawable

- 1. using an image saved in your project resources; (then refer the resource in your code or layout XML)

- 2. using an XML file that defines the Drawable properties;
  (then refer the XML file in your code or layout XML)

http://developer.android.com/guide/topics/resources/drawable-resource.html

Drawable class (for Java code)

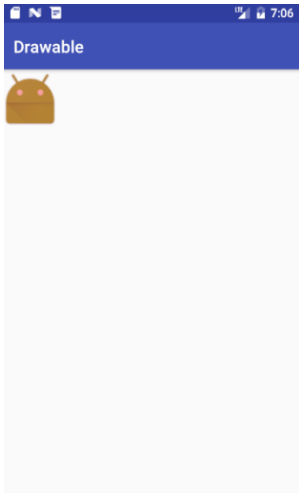
- you can handle image resources as a Drawable object in Java code

```
ImageView img = new ImageView(this);
// img.setImageResource(R.drawable.android);
Drawable myImage = getResources().getDrawable(R.drawable.my_image);
img.setImageDrawable(myImage);
```

- To add a resource Drawable to an Imageview in the XML layout
  - (we already learned this)

```
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:tint="#55ff0000"
android:src="@drawable/my_image"/>
```

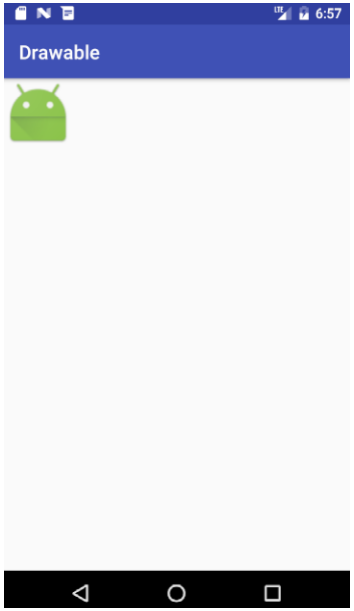
red tint



Example

```
package org.androidtown.listview.drawable;

import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

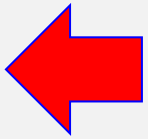
```
public class MainActivity extends AppCompatActivity {
    LinearLayout mLinearLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mLinearLayout = new LinearLayout(this);

// Instantiate an ImageView and define its properties
        ImageView img = new ImageView(this);
        Drawable myImage = getResources().getDrawable(R.drawable.ic_launcher);
        img.setImageDrawable(myImage);
        // to match the Drawable's dimensions
        img.setLayoutParams(new Gallery.LayoutParams(Gallery.LayoutParams.WRAP_CONTENT,
                    Gallery.LayoutParams.WRAP_CONTENT));

// Add the ImageView to the layout and set the layout as the content view
        mLinearLayout.addView(img);

        setContentView(mLinearLayout);
    }
}
```



Example : Layer List

- XML file saved at res/drawable/layers.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
<item>
    <bitmap android:src="@drawable/android_red"/>
</item>
<item android:top="10dp" android:left="10dp">
    <bitmap android:src="@drawable/android_blue"/>
</item>
<item android:top="20dp" android:left="20dp">
    <bitmap android:src="@drawable/android_green"/>
</item>
</layer-list>
```

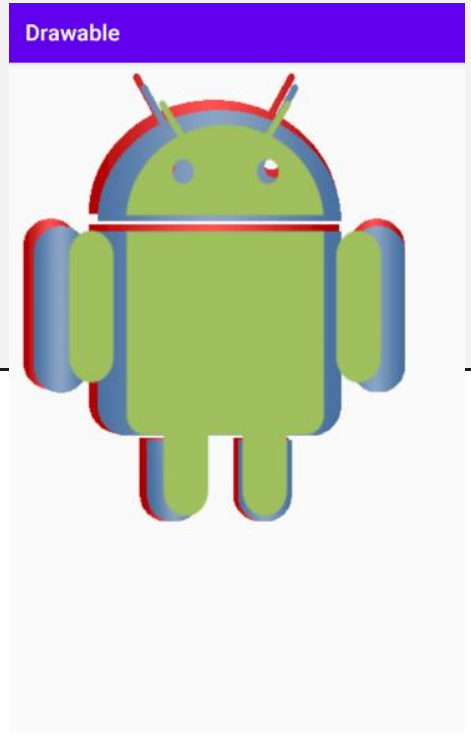


nested <bitmap> is used

Example : Layer List

- activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"    >


    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/layers"/>
</RelativeLayout>
```



Example : Transition Drawable

- cross-fade between the two drawable resources
- With a TransitionDrawable defined in a XML saved in res/drawable/my_transition.xml

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/android_red"/>
    <item android:drawable="@drawable/android_blue"/>
</transition>
```

- Applied to a View

```
<ImageButton
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/my_transition"/>
```

- You can instantiate an object by setting the above TransitionDrawable XML as the content of an ImageButton

```
ImageButton button = (ImageButton) findViewById(R.id.button);
TransitionDrawable drawable = (TransitionDrawable) button.getDrawable();
drawable.startTransition(5000);
```

Example : State List (useful!)

- XML file saved at res/drawable/button.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:drawable="@drawable/android_green"/>
    <item android:drawable="@drawable/android_red"/>
</selector>
```

- This layout XML applies the state list drawable to a Button:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"    >


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/button"/>
</RelativeLayout>
```

Drawing Bitmap on Canvas

- Working with Bitmap
  - You can draw bitmaps onto a canvas using drawBitmap() method
  - Class for bitmap : android.graphics.Bitmap
- To load a Bitmap resource

Example ID:
R.drawable.bluejay

```
Resources r = getResources();
BitmapDrawable bd=(BitmapDrawable) r.getDrawable(ID);
Bitmap pic = bd.getBitmap();
```

OR

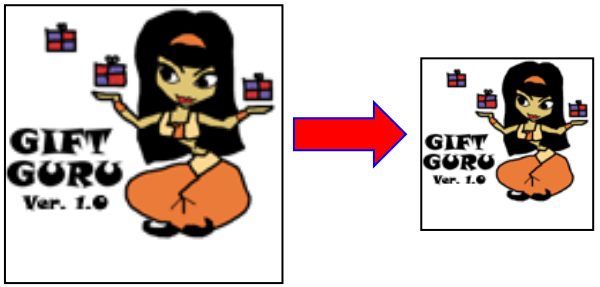
```
Import android.graphics.BitmapFactory;
…
Bitmap pic = BitmapFactory.decodeResource(getResources(), ID);
```

- Draw on canvas (there are many different drawing options)

```
canvas.drawBitmap(pic, 0, 0, null); //(Bitmap bitmap, float left, float top, Paint paint)
```

Drawing Bitmap on Canvas

- Scaling Bitmap Graphics
  - Use : **createScaledBitmap()** method
  - Example:




```
Bitmap sm = Bitmap.createScaledBitmap( pic, 50, 75, false);
```

- false: 현재 pixel 수 유지하며, 크기 조정
- true: 조정 된 크기에 맞게 pixel 수를 조정하여, 사이즈에 맞게 선명한 이미지 제공 (메모리 사용 증가)

Drawing Bitmap on Canvas

- Create Bitmap

```
static Bitmap createBitmap(Bitmap source, int x, int y,
                           int width, int height);
```

  - example

```
Bitmap sm = BitmapFactory.decodeResource(getResources(), ID);
```

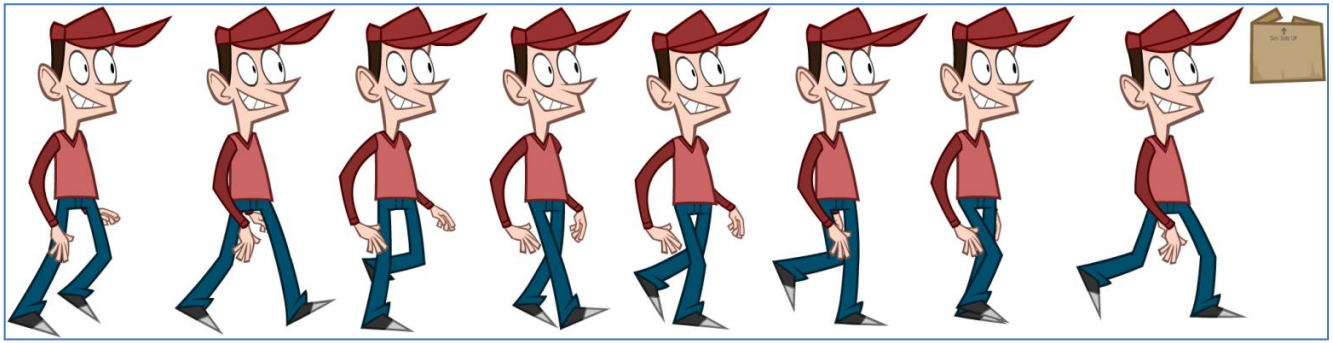
```
Bitmap bitm2 = Bitmap.createBitmap(sm, 0, 0, sm.getWidth(), sm.getHeight());
```

```
Bitmap bitm2 = Bitmap.createBitmap(sm, 10, 10, sm.getWidth()-20, sm.getHeight()-20);
```

Canvas method:

`void drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)`

: Draw the specified bitmap, scaling/translating automatically to fill the destination rectangle.



```
Bitmap pic = BitmapFactory.decodeResource(getResources(), ID);
```

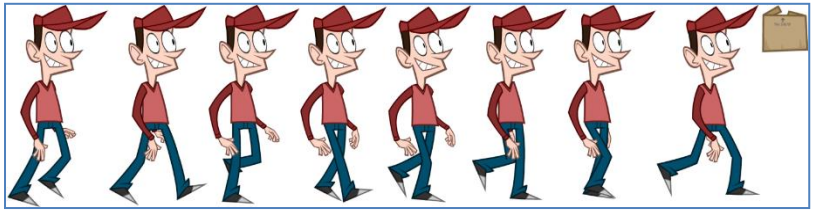
```
Paint Pnt = new Paint();
 canvas.drawColor(Color.WHITE);
canvas.drawBitmap(pic, new Rect(100, 0, 500, 500), new Rect(0, 0,400,500), Pnt);
```

exercise

- From **Making a custom view–Exercise**

```
public class MyDrawEx extends View {
    public MyDrawEx(Context c) {
        super(c);
    }
    public MyDrawEx(Context c, AttributeSet a) {
        super(c, a);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Bitmap sm = BitmapFactory.decodeResource(getResources(), R.drawable.image);

        Paint Pnt = new Paint();
        canvas.drawColor(Color.WHITE);

        Bitmap bitm2 = Bitmap.createBitmap(sm, 0, 0, sm.getWidth()-100, sm.getHeight()-100);
        canvas.drawBitmap(bitm2, 0,0, Pnt);
    }
}
```

Bitmap 자르기



exercise

- From **Making a custom view–Exercise**

```
public class MyDrawEx extends View {
    public MyDrawEx(Context c) {
        super(c);
    }
    public MyDrawEx(Context c, AttributeSet a) {
        super(c, a);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Bitmap sm = BitmapFactory.decodeResource(getResources(), R.drawable.image);

        Paint Pnt = new Paint();
        canvas.drawColor(Color.WHITE);
        canvas.drawBitmap(sm, new Rect(100, 0, 500, 500), new Rect(0, 0,400,500), Pnt);
    }
}
```

Exercise : Canvas with Bitmap

- There are two androids; green and red ones.
- Every time you touch the screen, the red android rotates by 10 degree

Exercise : Canvas with Bitmap

- Let's make another example

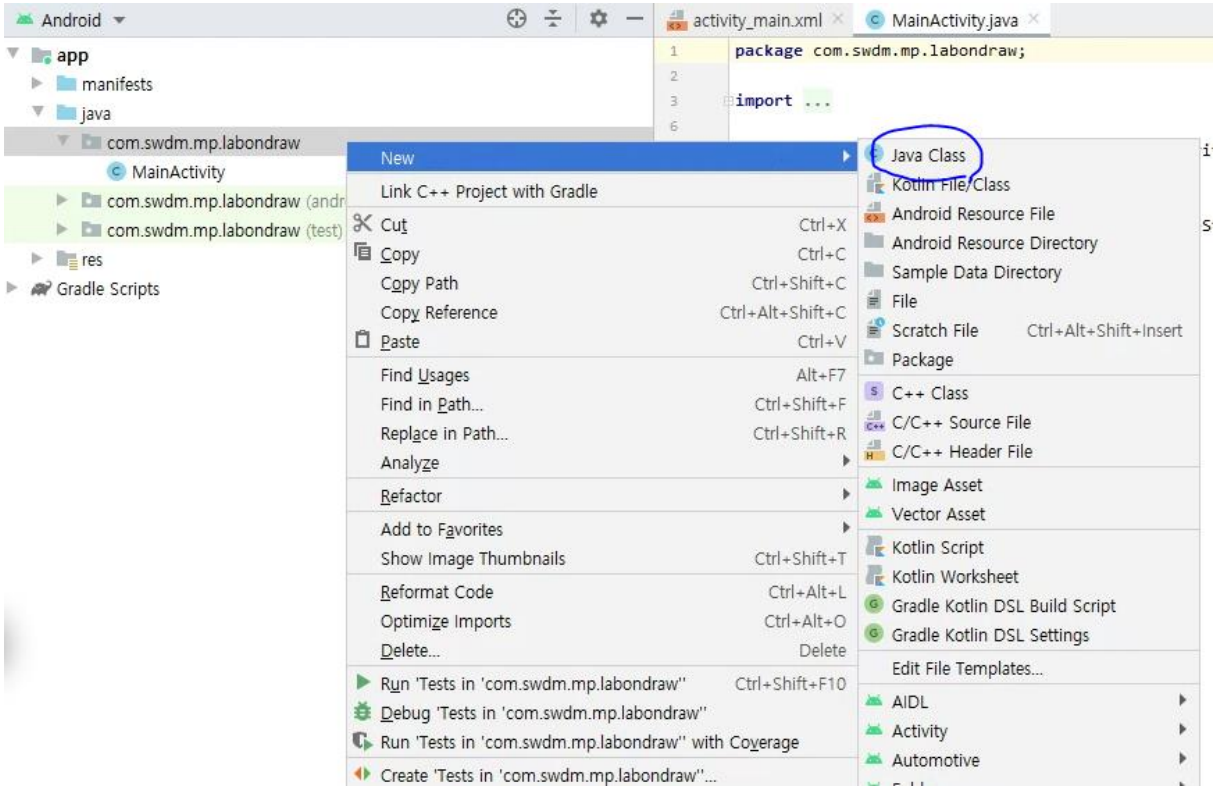
1. Create a new Java class that extends View
   within your package (e.g., com.mp)

New Project

- **LabOnDraw**

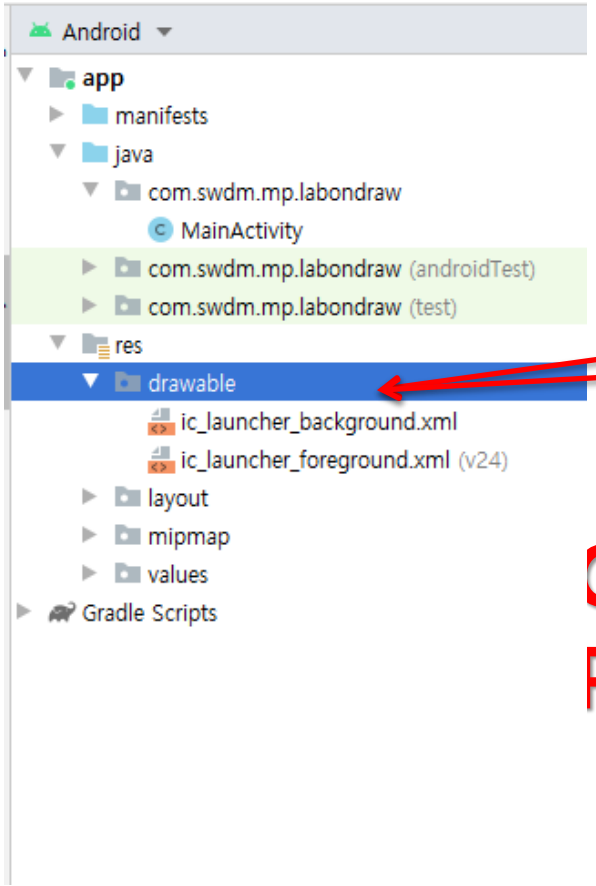
Add a new Class
extends "View"

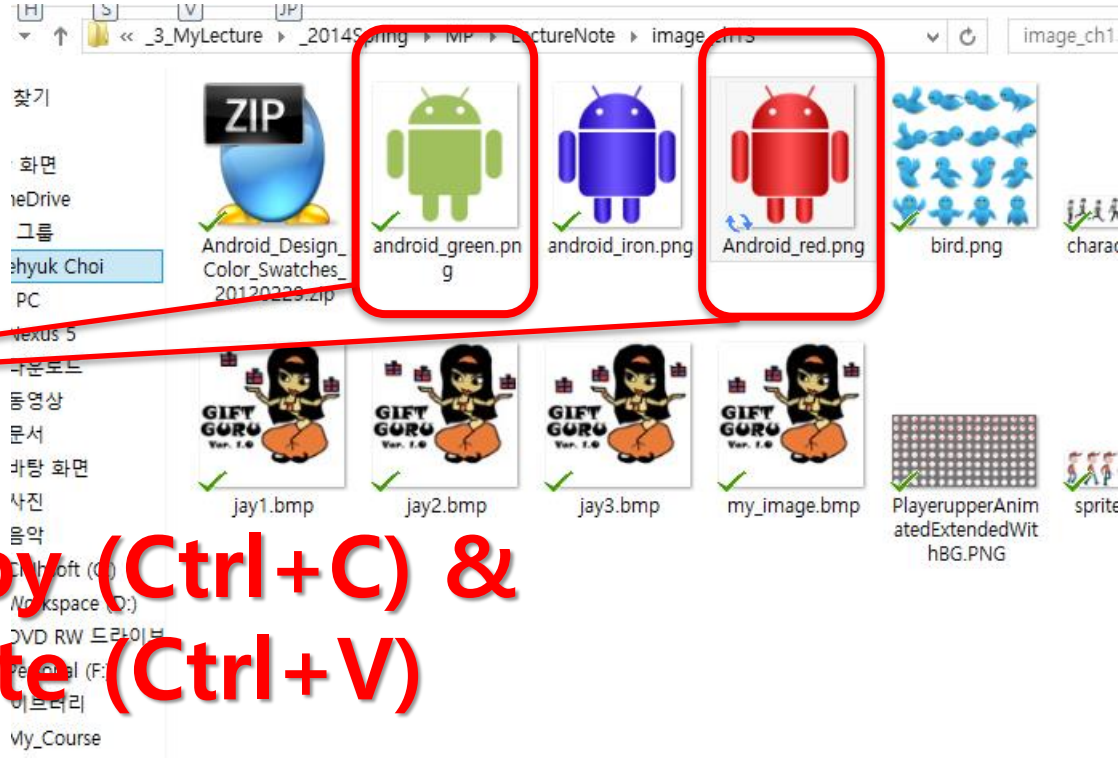
- **New class** → MyDrawEx.java



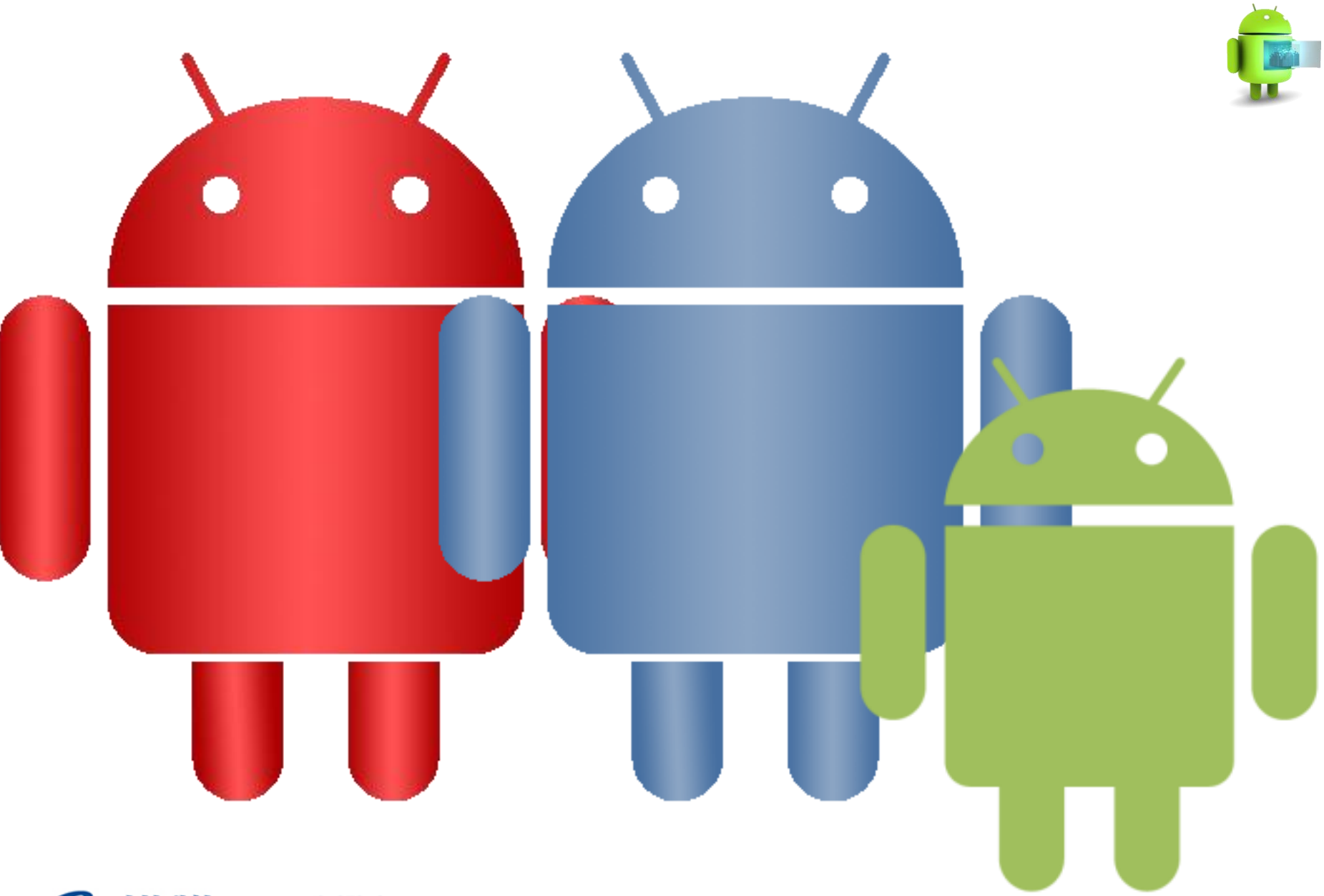
Exercise : Canvas with Bitmap

- Prepare Two Images : android_red.png, android_green.png





Copy (Ctrl+C) &
Paste (Ctrl+V)



Exercise : Canvas with Bitmap

- MyDrawEx.java

```
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
```

```
public class MyDrawEx extends View {
 private Paint mPaint;

    private Bitmap mAndroidGreen;
    private Bitmap mAndroidRed;
    private int nAngle = 0;
    public void init()
    {
        mPaint = new Paint();

        Resources res = getResources();
        mAndroidGreen = BitmapFactory.decodeResource(res, R.drawable.android_green);
        mAndroidRed = BitmapFactory.decodeResource(res, R.drawable.android_red);

    }
            public MyDrawEx(Context c) {
                        super(c);
                        init();
            }
            public MyDrawEx(Context c, AttributeSet a) {
                        super(c, a);
                        init();
            }
```

Exercise : Canvas with Bitmap

- MyDrawEx.java

```
public boolean onTouchEvent(MotionEvent event) {
    // if it's an up (“release”) action
    if (event.getAction() == MotionEvent.ACTION_UP) {
          nAngle = [________?________];   // rotate 5 degree angle upon a click
          invalidate();
    }
    // indicates that the event was handled
     return true;
  } // end of onTouchEvent

  protected void onDraw(Canvas canvas) {
          canvas.drawBitmap(mAndroidGreen, 0, 0, mPaint);
          canvas.save();                  // Save the current state

          canvas.rotate(nAngle);
          canvas.drawBitmap(mAndroidRed, 0, 0, mPaint);

          }

}
```

Exercise : Canvas with Bitmap

- Layout : activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

  <com.example.labondraw.MyDrawEx
    android:layout_width="match_parent"
    android:layout_height="match_parent"
  />

</LinearLayout>
```

and RUN!

Animation

- Android supports built-in tween animation
  - Transition Animation
    - Position Change
  - Rotation Animation
    - Rotation
  - Scale Animation
    - Scaling
  - Alpha Animation
    - Transparency

Animation

- Tween animation transformations
  - Alpha (Transparency changes)
    - <alpha>
      - android:fromAlpha , android:toAlpha
  - Rotate (Rotations)
    - <rotate>
      - android:fromDegrees, android:toDegrees,
  - Scale (Scaling)
    - <scale>
      - android:fromXScale, android:toXScale,
      - android:fromYScale, android:toYScale
  - Translate (Movement)
    - <translate>
      - Android:toXDelta, Android:toYDelta

Animation

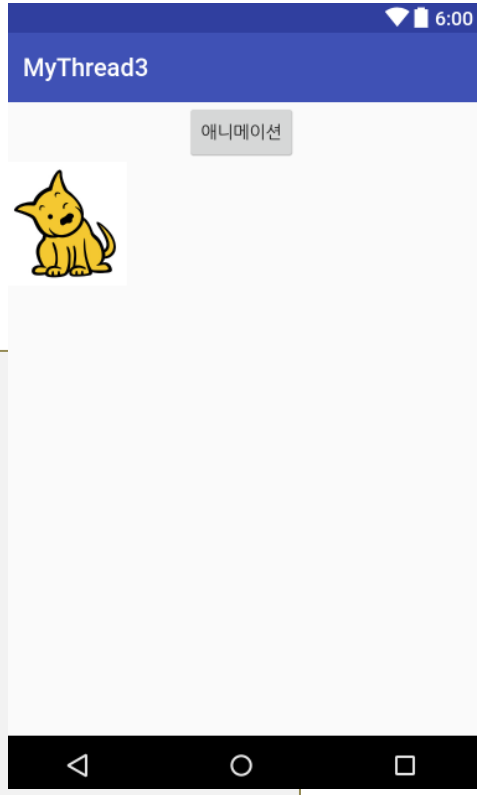
- Steps of Tween Animation
  - Create XML animation resource file in /res/anim/directory
  - Load the XML animation resource file into Animation object
  - Start Animation

Exercise

- Let's make an animation effect.
  - Make a layout!

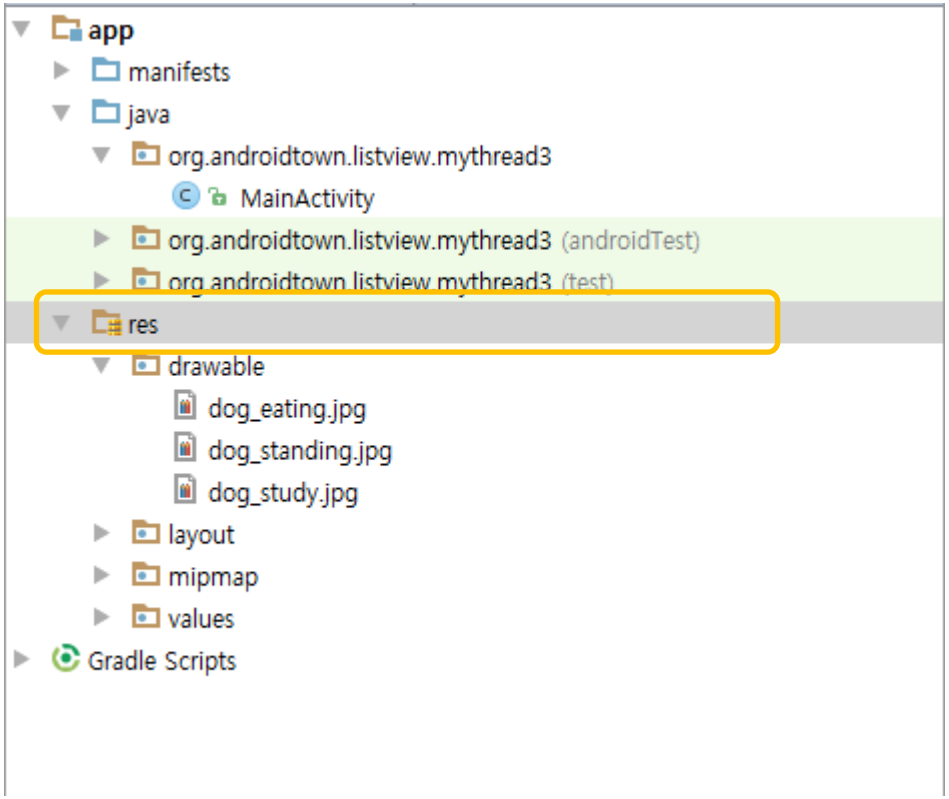
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <Button
        android:id="@+id/button"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="애니메이션"
        />

    <LinearLayout
    android:id="@+id/linear"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_below="@+id/button">
```

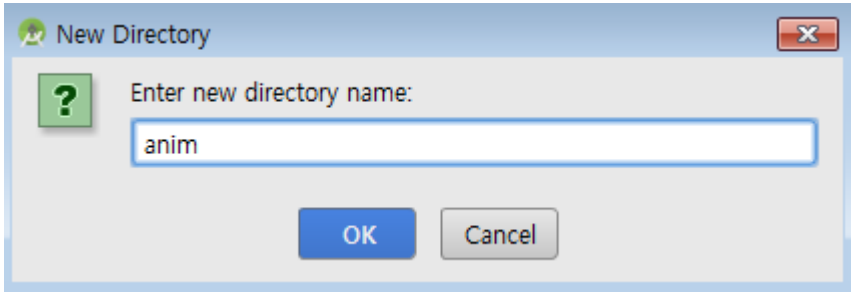
```
 <ImageView
    android:id="@+id/imageView1"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:src="@drawable/dog_standing"
    android:layout_gravity="left"
    />
    </LinearLayout>

    <EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="350dp"
    android:layout_below="@+id/linear"
     />

</RelativeLayout>
```



Exercise

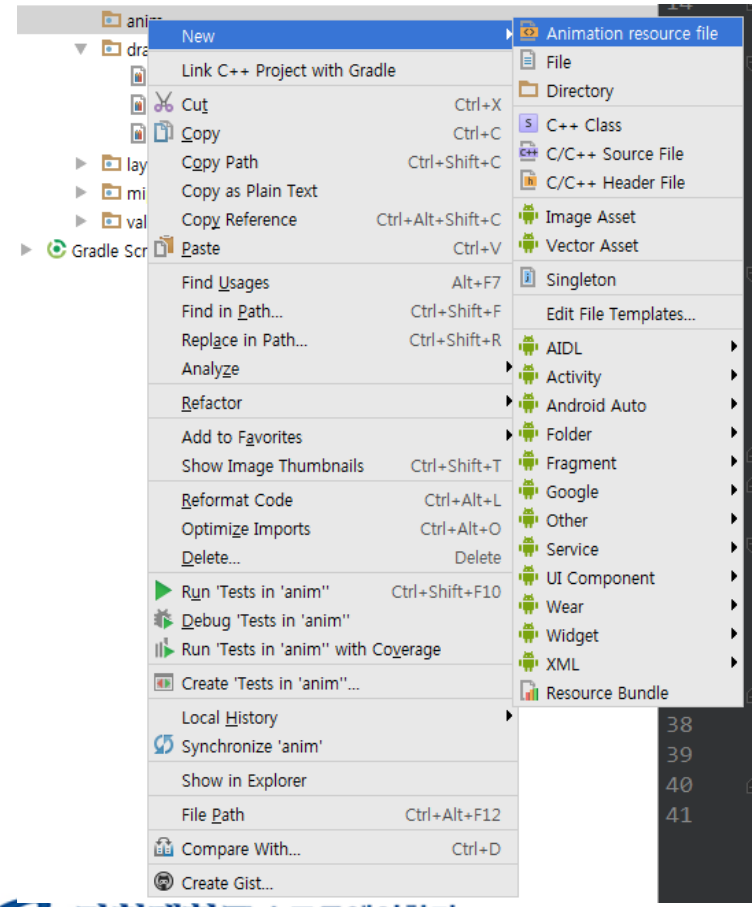
- Make an directory “anim” to res folder

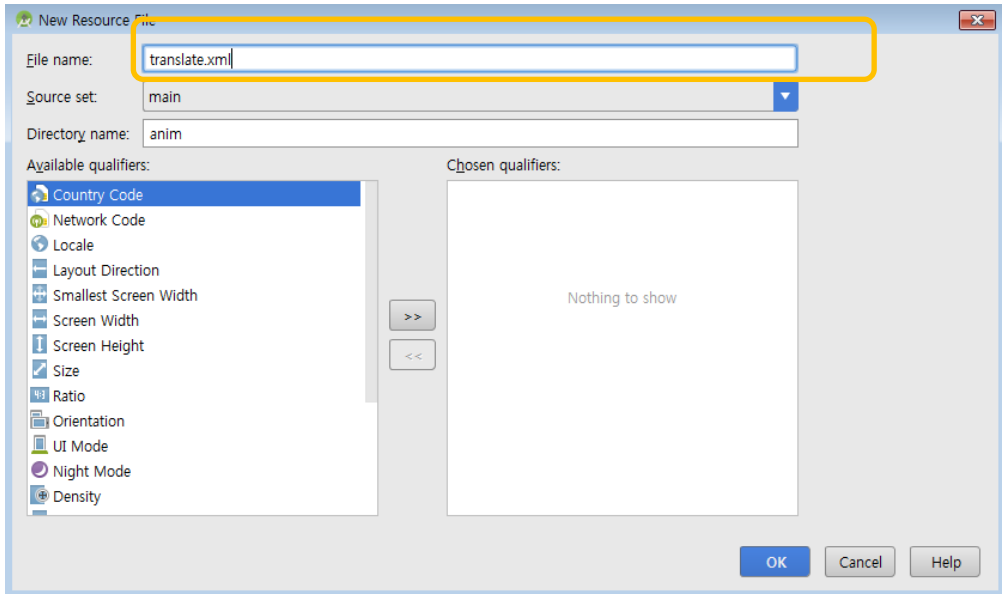




Exercise

- Make an animation resource file at “anim” folder





Exercise

- translate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate android:fromXDelta="0%"
        android:toXDelta="100%p"
        android:duration="1500"
        />
</set>
```

Let's try to change this value as "-100%p"

Exercise

- MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    ImageView imageView1;
    EditText editText;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView1 = (ImageView) findViewById(R.id.imageView1);
        editText = (EditText) findViewById(R.id.editText);
        button = (Button) findViewById(R.id.button);

        button.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
               Animation anim = AnimationUtils.loadAnimation(
                     getApplicationContext(), R.anim.translate);
                imageView1.startAnimation(anim);
                editText.append("애니메이션시작됨.\n");
            }
        });

    }
}
```

```
package org.androidtown.listview.mythread3;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
```

RUN!!!

Exercise

- translate.xml : applying different animation effect
  - Scale action : RUN!!

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<scale
    android:duration="2500"
    android:pivotX="50%"
    android:pivotY="50%"
    android:fromXScale="1.0"
    android:fromYScale="1.0"
    android:toXScale="2.0"
    android:toYScale="2.0"
    />

    <scale
        android:startOffset="2500"
        android:duration="2500"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:toXScale="0.5"
        android:toYScale="0.5"
        />
</set>
```

확대 축소를 적용하기 위한 중심 축

Exercise

- translate.xml : applying different animation effect
  - Rotate action: RUN!!

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<rotate
    android:fromDegrees="60"
    android:toDegrees="180"
    android:duration="2500"
    android:pivotX="50%"
    android:pivotY="50%"
    />
</set>
```

  - Transparent action: RUN!!

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<alpha
    android:fromAlpha="0.5"
    android:toAlpha="1.0"
    android:duration="2500"
    />
</set>
```

Appendix

- Event handling (Java code vs XML attributes)

```
Button btn = (Button) findViewById(R.id.mybutton);

btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        myClickEventMethod(v);
    }
});

// some more code

public void myClickEventMethod(View v) {
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<!-- layout elements -->
<Button android:id="@+id/mybutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!"
    android:onClick=" myClickEventMethod" />
<!-- even more layout elements -->
```