13장. 커스텀 뷰 개발

13.1 커스텀 뷰 작성 방법

- API에서 제공하는 뷰를 그대로 이용하면서 약간 변형시킨 뷰
- 여러 뷰를 합쳐서 한번에 출력하기 위한 뷰
- 기존 API에 전혀 존재하지 않는 뷰

```
public class MyView extends TextView {

}
```

```
public class MyView extends ViewGroup {

}
```

```
public class MyView extends View {

}
```

13.1 커스텀 뷰 작성 방법

- 커스텀 뷰를 레이아웃 XML에 등록해서 이용하려면 생성자 3개를 모두 정의

```
public class MyView extends View {
    Context context;

    public MyView(Context context) {
        super(context);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public MyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }
}
```

- onDraw( ) 함수
- 이 함수에서 그린 내용이 뷰 영역에 출력

13.1 커스텀 뷰 작성 방법

```
protected void onDraw(Canvas canvas) {
    canvas.drawColor(Color.alpha(Color.CYAN));

    RectF rect = new RectF(15, 15, 160, 160);
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setColor(Color.RED);
    canvas.drawArc(rect, 0, 360, false, paint);
}
```

- 커스텀 뷰를 레이아웃 XML에 등록할 때는 클래스명만 등록하면 안 되고, 전체 패키지명으로 등록

```
<com.example.test4_13.MyView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
```

- 커스텀 속성 이용
- res/values 폴더 하위에 attrs.xml 파일을 이용하며, <declare-styleable> 태그로 속성을 등록

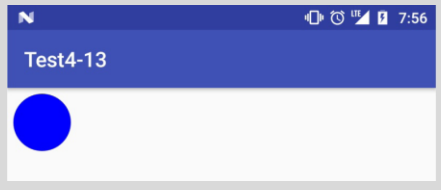
```
<resources>
    <declare-styleable name="MyView">
        <attr name="customColor" format="color"/>
    </declare-styleable>
</resources>
```

13.1 커스텀 뷰 작성 방법

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.test4_13.MyView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        custom:customColor="#0000FF"
        />
</RelativeLayout>
```

- AttributeSet을 이용하여 속성값을 추출

```
public MyView(Context context, AttributeSet attrs) {
    super(context, attrs);
    this.context=context;

    if(attrs != null){
        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.MyView);
        color=a.getColor(R.styleable.MyView_customColor, Color.RED);
    }

}
```

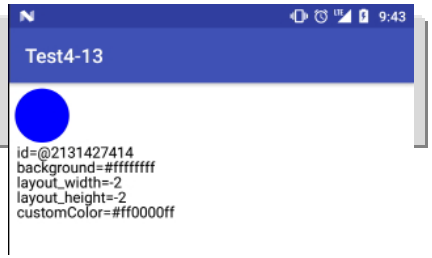


13.1 커스텀 뷰 작성 방법

속성값을 획득하는 함수

- int getAttributeCount(): 속성 개수
- String getAttributeName(int index): 속성명 획득
- String getAttributeValue(int index): 속성값 획득
- int getAttributeIntValue(int index, int defaultValue): 속성값 획득
- boolean getAttributeBooleanValue(int index, boolean defaultValue): 속성값 획득
- float getAttributeFloatValue(int index, float defaultValue): 속성값 획득

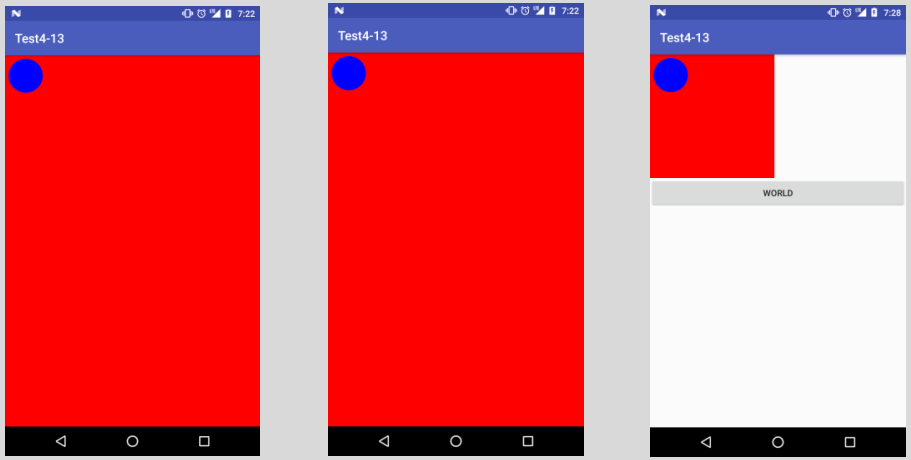
```
for (int i=0;i<attrs.getAttributeCount();i++) {
    attributes[i]=attrs.getAttributeName(i)+"="+attrs.getAttributeValue(i);
}
```



13.1 커스텀 뷰 작성 방법

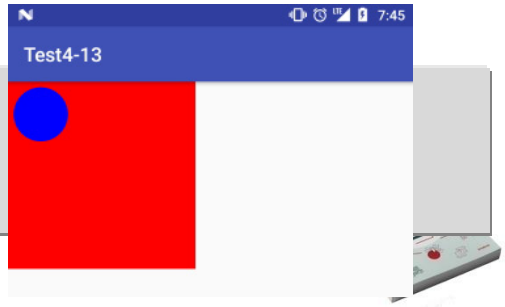
- 크기 결정

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <com.example.test4_13.MyView
        android:id="@+id/myView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        custom:customColor="#0000FF"
        android:background="#ff0000"
        />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="world"/>
</LinearLayout>
```



- 뷰 내부에서 크기 결정을 위해 onMeasure( ) 함수를 이용

```
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);
    setMeasuredDimension(500, 500);
}
```

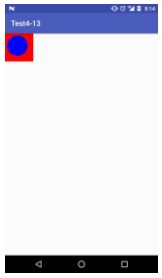
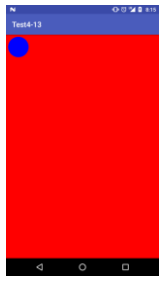


13.1 커스텀 뷰 작성 방법

- 레이아웃 XML 파일의 크기 설정 정보는 onMeasure( ) 함수의 매개변수로 전달

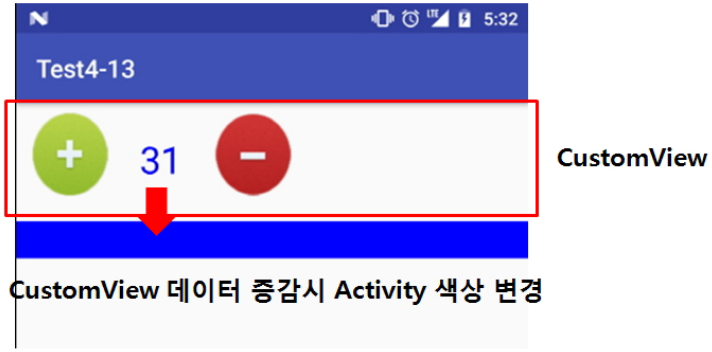
```
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int widthMode = MeasureSpec.getMode(widthMeasureSpec);
    int widthSize = MeasureSpec.getSize(widthMeasureSpec);

    int heightMode = MeasureSpec.getMode(heightMeasureSpec);
    int heightSize = MeasureSpec.getSize(heightMeasureSpec);

    //.....................
}
```

- MeasureSpec.AT_MOST: 뷰 내부에서 지정하라는 의미. 레이아웃 XML에서 wrap_content 로 선언한 경우
- MeasureSpec.EXACTLY: 뷰를 이용하는 액티비티 쪽에서 크기를 결정한 경우. 레이아웃 XML에서 fill_parent, match_parent, 100px 등으로 선언한 경우
- MeasureSpec.UNSPECIFIED: 모드가 설정되지 않았을 경우

13.1 커스텀 뷰 작성 방법

- 이벤트 추가




```
public interface OnMyChangeListener {
    void onChange(int value);
}
```

- 뷰 내부에 setOnMyChangeListener( ) 함수를 호출하여 객체를 등록

```
public class MyPlusMinusView extends View {

    //.....
    //Observer를 등록하기 위한 객체
    ArrayList<OnMyChangeListener> listeners;

    //Observer 등록을 위한 함수
    public void setOnMyChangeListener(OnMyChangeListener listener){
        listeners.add(listener);
    }
}
```

13.1 커스텀 뷰 작성 방법

- 뷰에서 이벤트를 처리하기 위한 함수를 재정의

```
public boolean onTouchEvent(MotionEvent event) {
    //...
        //데이터 변경
        value++;
        //화면 갱신
        invalidate();
        for(OnMyChangeListener listener : listeners){
            //observer에 데이터 전달
            listener.onChange(value);
        }
    //...
}
```

13.1 커스텀 뷰 작성 방법

- 커스텀 뷰를 이용하는 액티비티의 코드

```
public class MainActivity extends AppCompatActivity implements OnMyChangeListener{

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

        MyPlusMinusView plusMinusView=                    (MyPlusMinusView)findViewById(R.id.customView);

        //인터페이스를 구현한 객체를 View에 등록
        plusMinusView.setOnMyChangeListener(this);
    }

    @Override
    public void onChange(int value) {
        //...
    }
}
```

13.1 커스텀 뷰 작성 방법

① 인터페이스 정의

```
public interface OnMyChangeListener {
    void onChange(int value);
}
```

② 인터페이스 구현

```
public class MainActivity extends
AppCompatActivity implements
OnMyChangeListener{
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        //인터페이스를 구현한 객체를 View에
등록

plusMinusView.setOnMyChangeListener(this);
    }

    @Override
    public void onChange(int value) {

    }
}
```

```
public class MyPlusMinusView extends View {
    //Observer를 등록하기 위한 객체
    ArrayList<OnMyChangeListener> listeners;

    //onserver 등록을 위한 함수
    public void setOnMyChangeListener(OnMyChangeListener
listener){
        listeners.add(listener);
    }
    //사이즈 결정
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        //데이터 변경
        value++;
        //화면 갱신
        invalidate();
        for(OnMyChangeListener listener : listeners){
            //observer에게 데이터 전달
            listener.onChange(value);
        }
    }
    @Override
    protected void onDraw(Canvas canvas) {
    }
}
```

③ 객체 등록

④ 이벤트 발생

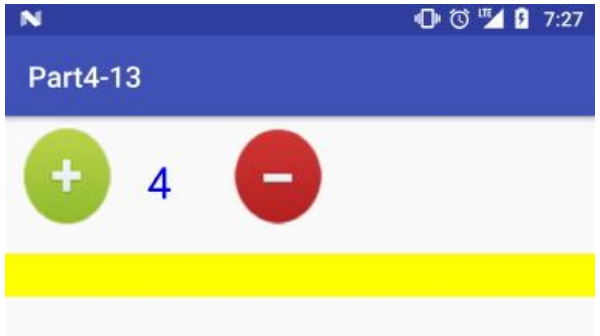
⑤ 등록 객체의 함수 호출

Step by Step 13-1 – Custom View

Custom View를 이용하는 실습

플러스, 마이너스 아이콘으로 숫자 데이터를 발생
데이터를 Activity에 전달해서 이용하게 하는 Custom View

1. Module 생성
2. 파일 복사
3. attrs.xml 생성
4. OnMyChangeListener 인터페이스 생성
5. MyPlusMinusView 작성
6. activity_main.xml 작성
7. MainActivity 추가
8. 실행



13.2 그래픽 프로그램

13.2.1. 뷰를 그리는 방법

- Canvas: 그래픽 함수를 제공해주는 클래스. 이 클래스의 함수를 이용하여 뷰의 화면을 그림
- Paint: 그리기 옵션 지정. 색상, 투명도 등의 속성을 지정

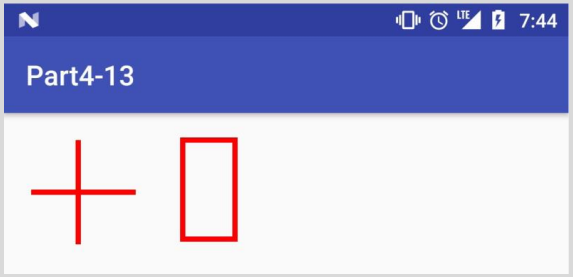
```
protected void onDraw(Canvas canvas) {
    Paint paint=new Paint();
    paint.setColor(Color.RED);

    canvas.drawCircle(50, 50, 50, paint);
}
```

- drawPoint(float x, float y, Paint paint): 점을 하나 찍는 함수.
- drawLine(float startX, float startY, float stopX, float stopY, Paint paint): 선을 그리는 함수.
- drawRect(float left, float top, float right, float bottom, Paint paint): 사각형을 그리는 함수.
- drawRect(RectF rect, Paint paint): 사각형을 그리는 함수.
- drawCircle(float cx, float cy, float radius, Paint paint): 원을 그리는 함수. 원점에 대한 x, y 좌푯값과 원의 반경 정보로 원을 그림

13.2 그래픽 프로그램

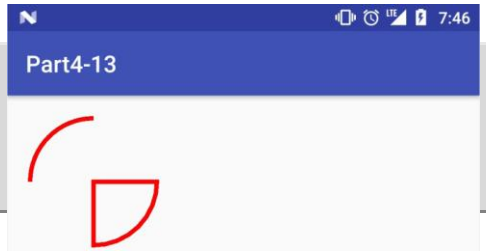
- drawArc(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint): 아크(Arc)를 그리는 함수.
- drawText(String text, float x, float y, Paint paint): 문자열을 지정된 좌표에 그리는 함수
- drawBitmap(Bitmap bitmap, float left, float top, Paint paint): 이미지를 지정된 위치에 그리는 함수
- drawRoundRect(RectF rect, float rx, float ry, Paint paint): 모서리가 둥근 사각형을 그리는 함수
- drawOval(RectF oval, Paint paint): 타원을 그리는 함수

```
Paint paint=new Paint();
paint.setColor(Color.RED);
paint.setStrokeWidth(10);

canvas.drawLine(100, 10, 100, 210, paint);
canvas.drawLine(10, 110, 210, 110, paint);

paint.setStyle(Paint.Style.STROKE);
RectF rect=new RectF(300, 10, 400, 200);
canvas.drawRect(rect, paint);
```

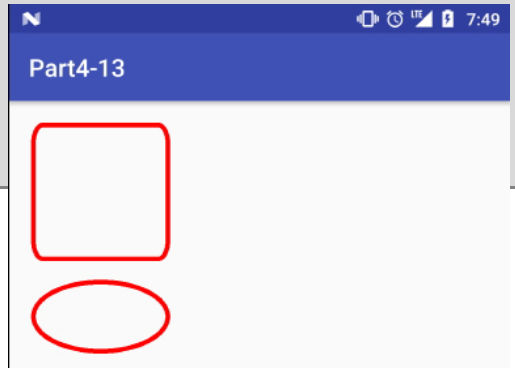


13.2 그래픽 프로그램

```
paint.setStyle(Paint.Style.STROKE);
RectF arcRect=new RectF(10, 10, 300, 300);
canvas.drawArc(arcRect, 0, 90, true, paint);
canvas.drawArc(arcRect, -90, -90, false, paint);
```




```
RectF roundRect=new RectF(10,10,300,300);
canvas.drawRoundRect(roundRect, 20, 40, paint);

RectF ovalRect=new RectF(10,350,300,500);
canvas.drawOval(ovalRect, paint);
```



뷰의 영역 전체를 지정된 색으로 칠하는 함수

- drawRGB(int r, int g, int b)
- drawColor(int color)
- drawPaint(Paint paint)

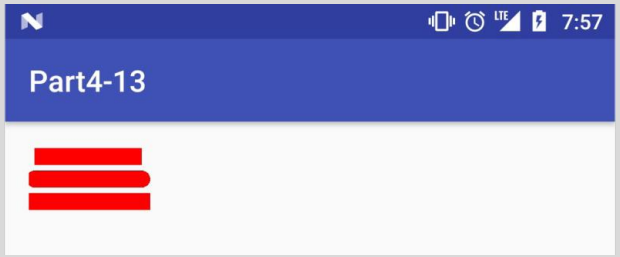
13.2 그래픽 프로그램

13.2.2. Paint 클래스

그리기 효과를 지정하기 위한 클래스

- setColor(int color)
- setARGB(int a, int r, int g, int b)
- setAntiAlias(boolean aa)
- setStyle(Paint.Style style)
- setStrokeWidth(float width)
- setStrokeCap(Paint.Cap cap)
- setStrokeJoin(Paint.Join join)

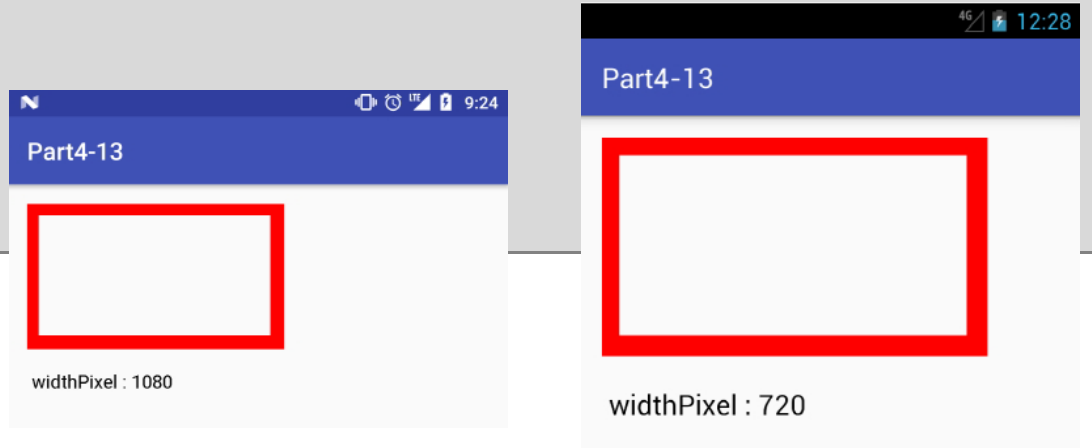
```
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeWidth(30);
canvas.drawLine(10, 20, 200, 20, paint);

paint.setStrokeCap(Paint.Cap.ROUND);
canvas.drawLine(10, 60, 200, 60, paint);

paint.setStrokeCap(Paint.Cap.SQUARE);
canvas.drawLine(10, 100, 200, 100, paint);
```



13.2.3. 코드에서 논리적 크기 획득

- 자바 코드에서 개발자가 직 접 크기를 명시할 때는 논리적 단위를 사용할 수 없으며, 오직 픽셀 단위로만 적용

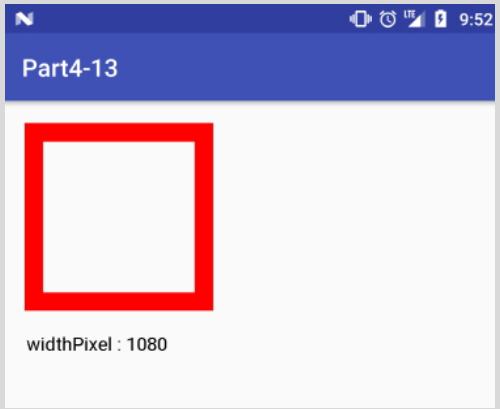
```
Paint paint=new Paint();
paint.setColor(Color.RED);
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeWidth(30);

Rect rect=new Rect(10, 10, 540, 300);
canvas.drawRect(rect,paint);
```

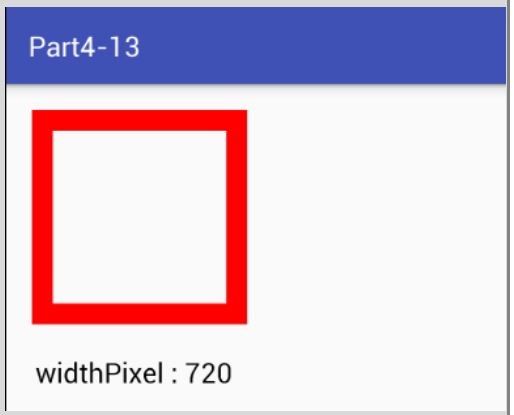


DisplayMetrics 객체를 이용해서 스마트폰의 크기 정보를 획득한 다음, 자바 코드에서 스마트폰 크기 호환성을 고려한 크기 계산

13.2 그래픽 프로그램

```
DisplayMetrics dm=getResources().getDisplayMetrics();

float strokeWidth=15*dm.density;
int rectWidth=(int)(150*dm.density);
int rectHeight=(int)(150*dm.density);

int position=(int)(10*dm.density);

Paint paint=new Paint();
paint.setColor(Color.RED);
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeWidth(strokeWidth);

Rect rect=new Rect(position, position, rectWidth, rectHeight);
canvas.drawRect(rect,paint);
```





크기를 리소스로 등록하여 이용하는 방법

```
<dimen name="strokeWidth">15dp</dimen>
<dimen name="size">150dp</dimen>
<dimen name="position">10dp</dimen>
```

```
int size = context.getResources().getDimensionPixelSize(R.dimen.size);
int strokeWidth=      context.getResources().getDimensionPixelSize(R.dimen.strokeWidth);
int position=
   context.getResources().getDimensionPixelSize(R.dimen.position);
```