

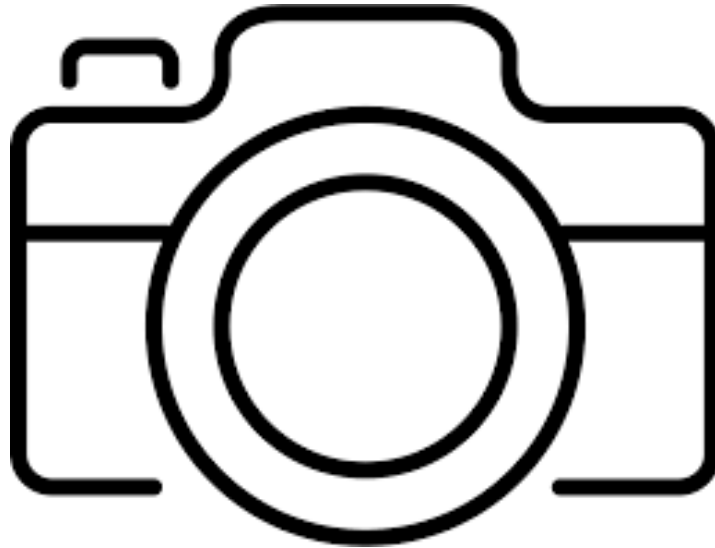


APLIKACJE MOBILNE

Fotografia
Grafika

Camera

Wykonywanie zdjęć za pomocą kamery smartfona



Najprostszą metodą wykonania fotografii jest wysłanie do systemu intencji niejawnej, która będzie obsługiwana przez wbudowaną aplikację obsługi kamery.

```
Intent fotka = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
startActivityForResult(fotka, requestCode: 1);
```

MediaStore.**ACTION_IMAGE_CAPTURE** – pobieranie obrazu

MediaStore.**ACTION_VIDEO_CAPTURE** – pobieranie wideo

Intencję wysyłamy do systemu poleceniem `startActivityForResult()` gdyż oczekujemy odpowiedzi wywołanej aplikacji – czyli fotografii

Aby odebrać odpowiedź aplikacji obsługi kamery nadpisujemy metodę systemową `onActivityResult()`

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if (requestCode == 1 && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap bitmap = (Bitmap) extras.get("data");
        ramka.setImageBitmap(bitmap);
    }
}
```

Kamera zwraca bitmapę zapakowaną jako Bundle – standardowy identyfikator to „data”.

Zawartość paczki możemy zapisać w zmiennej typu Bitmap i przekazać do programu (np. wyświetlić w komponencie ImageBox).

ZADANIE PRAKTYCZNE:

```
public class MainActivity extends AppCompatActivity {  
    final static int REQUEST_IMAGE_CAPTURE = 1;  
    Button buttonTakePicture;  
    ImageView imageView;
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    buttonTakePicture = findViewById(R.id.button_take_picture);  
    imageView = findViewById(R.id.imageView);  
    buttonTakePicture.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);  
        }  
    });  
}
```

Przykład 1 – aplikacja wykonuje zdjęcie i wyświetla na ekranie.

Przykład 1. c.d.

@Override

```
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {  
        Bundle extras = data.getExtras();  
        Bitmap imageBitmap = (Bitmap) extras.get("data");  
        imageView.setImageBitmap(imageBitmap);  
    }  
}
```

Kamera

Przykład 2 – aplikacja wykonuje zdjęcie i zapisuje na dysku – z numerem kolejnym w nazwie.

```
Button migawka;  
TextView opis;  
int licznik=0;  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    migawka=findViewById(R.id.button01);  
    opis=findViewById(R.id.textView01);  
    migawka.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            akcja();  
        }  
    });  
}
```


Przykład 2 – c.d.

```
private void akcja() {
    Intent zdjecie = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File fotka=null;

    String nazwaPliku="fotka_"+String.valueOf(++licznik);
    String rozszerzenie=".jpg";
    String sciezka="";
    File katalog = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
    try {
        fotka = File.createTempFile(nazwaPliku,rozszerzenie,katalog);
    } catch (IOException e) {
        e.printStackTrace();
    }
    zdjecie.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(fotka));
    startActivityForResult(zdjecie, requestCode: 1);
    sciezka=katalog.getAbsolutePath()+"/"+nazwaPliku+rozszerzenie;
    Log.d( tag: "LOG:",sciezka);
    opis.setText(sciezka);
}
```

Canvas – rysowanie po ekranie



Wstęp to tworzenia własnej
grafiki – rysowanie na obiekcie
Canvas

Aby wykonywać rysunki, będziemy korzystać z obiektu klasy Canvas - można go uznać za płótno, po którym będziemy rysować.

W najprostszym przypadku Canvas przykryje całość ekranu.
Należy więc utworzyć obiekt klasy Canvas i ustawić go zamiast domyślnego layoutu (tego określanego w pliku XML) – poleceniem `setContent()`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setContentView(new CanvasView(context, this));
}
```

Klasę CanvasView należy dodać do projektu

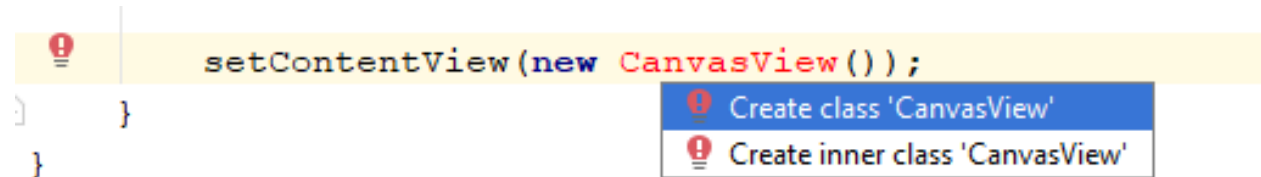
Aby wykonywać rysunki, będziemy korzystać z obiektu klasy Canvas - można go uznać za płótno, po którym będziemy rysować.

W najprostszym przypadku Canvas przykryje całość ekranu.
Należy więc utworzyć obiekt klasy Canvas i ustawić go zamiast domyślnego layoutu (tego określanego w pliku XML) – poleceniem `setContent()`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setContentView(new CanvasView( context: this));
}
```

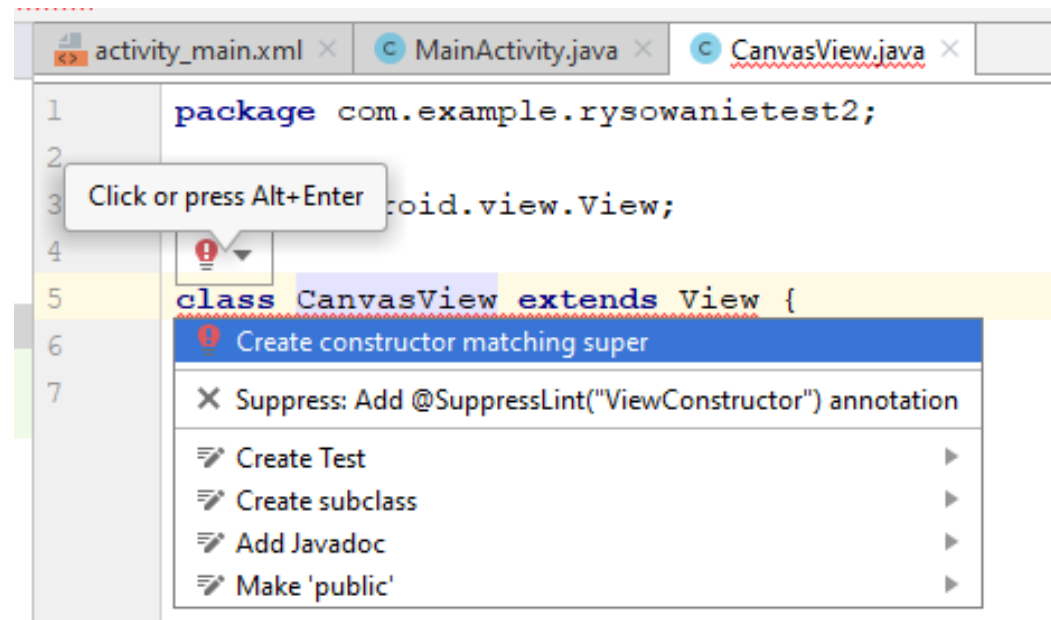
Klasę CanvasView należy dodać do projektu

Klasę CanvasView należy dodać do projektu

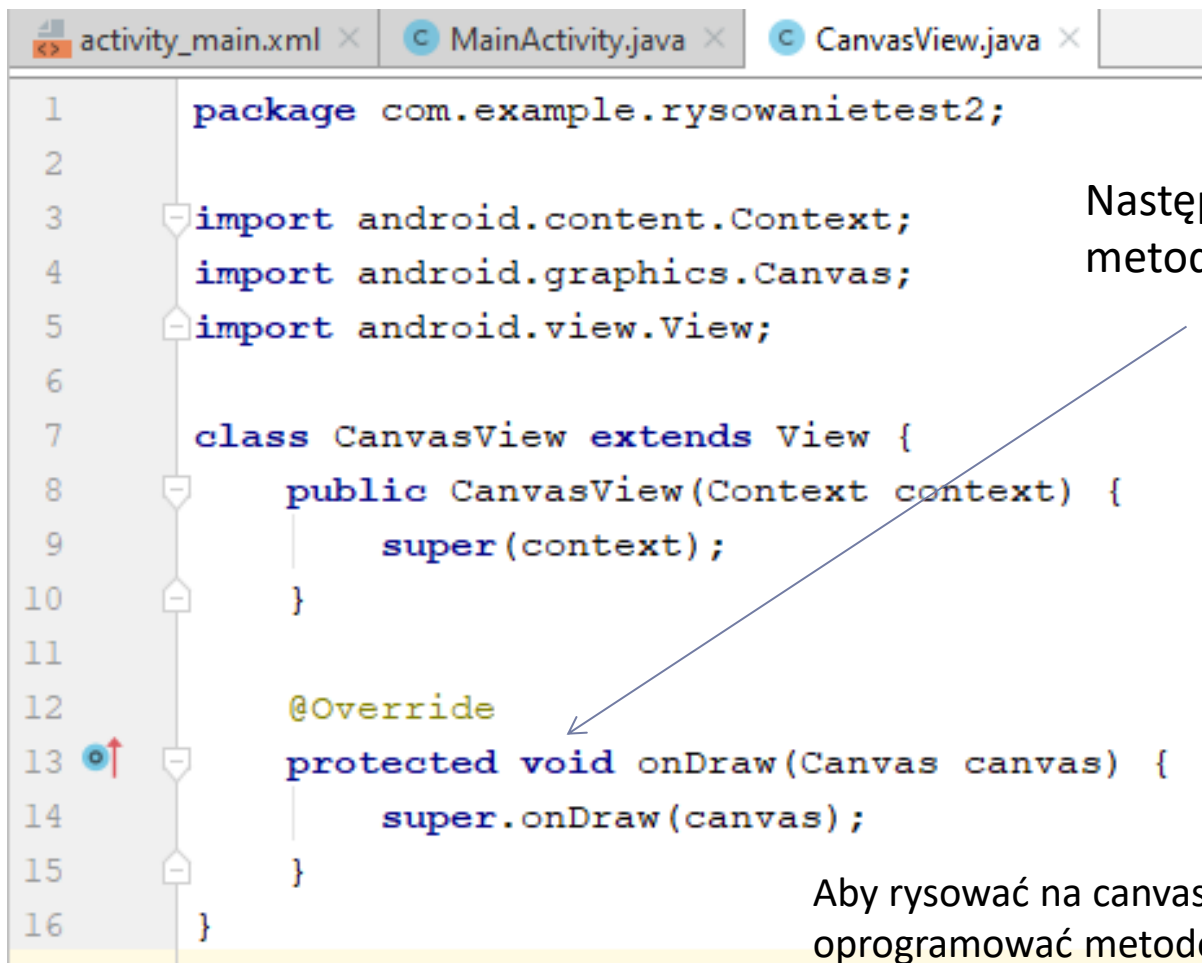


Utworzony zostanie plik CanvasView.java zawierający klasę CanvasView

Należy teraz
dodać
konstruktor klasy



Klasę CanvasView należy dodać do projektu



```
1 package com.example.rysowanietest2;
2
3 import android.content.Context;
4 import android.graphics.Canvas;
5 import android.view.View;
6
7 class CanvasView extends View {
8     public CanvasView(Context context) {
9         super(context);
10    }
11
12    @Override
13    protected void onDraw(Canvas canvas) {
14        super.onDraw(canvas);
15    }
16 }
```

Następnie nadpisujemy metodę onDraw()

Aby rysować na canvas-ie należy oprogramować metodę onDraw

Do rysowania na canvas-ie służy „pędzel” – czyli obiekt klasy Paint.

```
Paint paint = new Paint();  
paint.setARGB(a: 255, r: 0, g: 0, b: 0);  
paint.setStyle(Paint.Style.FILL_AND_STROKE);
```

Możemy ustawiać właściwości płótna (np. kolor)

```
canvas.drawColor(Color.CYAN);
```

Pobieranie aktualnego rozmiaru płótna

```
canvas.getWidth();  
canvas.getHeight();
```

Do rysowania na canvas-ie służy „pędzel” – czyli obiekt klasy Paint.

```
Paint paint = new Paint();  
paint.setARGB(a: 255, r: 0, g: 0, b: 0);  
paint.setStyle(Paint.Style.FILL_AND_STROKE);
```

Możemy ustawiać właściwości płótna (np. kolor)

```
canvas.drawColor(Color.CYAN);
```

Pobieranie aktualnego rozmiaru płótna

```
canvas.getWidth();  
canvas.getHeight();
```




Rysowanie prymitywów graficznych:.

```
canvas.drawLine( startX: 0, startY: 0, stopX: 500, stopY: 500, paint);  
canvas.drawRect( left: 10, top: 10, right: 300, bottom: 300, paint);
```

ZADANIE PRAKTYCZNE:

„Wygaszacz ekranu”

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setContentView(new CanvasView( context: this));  
    }  
}
```

```
activity_main.xml x MainActivity.java x CanvasView.java x
4      import android.graphics.Canvas;
5      import android.graphics.Color;
6      import android.graphics.Paint;
7      import android.view.View;
8
9      import java.util.Random;
10
11     class CanvasView extends View {
12         Paint paint1= new Paint();
13         Paint paint2= new Paint();
14         Random random = new Random();
15
16         public CanvasView(Context context) {
17             super(context);
18         }
19     }
```

```
@Override
protected void onDraw(Canvas canvas) {

    int x1, y1, dx1, dy1, x2, y2, dx2, dy2;
    paint1.setColor(Color.argb(random.nextInt( bound: 255), random.nextInt( bound: 255),
                                random.nextInt( bound: 255), random.nextInt( bound: 255)));
    paint2.setColor(Color.argb( alpha: 255, red: 255, green: 255, blue: 255));
    x1= random.nextInt( canvas.getWidth());
    y1= random.nextInt( canvas.getHeight());
    dx1 = random.nextInt( bound: 10)-5;
    dy1 = random.nextInt( bound: 10)-5;
    x2= random.nextInt( canvas.getWidth());
    y2= random.nextInt( canvas.getHeight());
    dx2 = random.nextInt( bound: 10)-5;
    dy2 = random.nextInt( bound: 10)-5;
```

- Losujemy kolor pędzla,
- Losujemy dwa punkty (x1,y1) oraz (x2,y2) - początek i koniec odcinka
- Losujemy wektory o które przesuwać będziemy końce odcinka (dx1,dy1) oraz (dx2,dy2)

```
int i=1;
while (i<200) {
    canvas.drawLine(x1, y1, x2, y2, paint1);
    x1+=dx1;
    y1+=dy1;
    x2+=dy2;
    y2+=dy2;
    if ((x1>=canvas.getWidth()) || (x1<=0) ) {dx1=(-dx1);}
    if ((x2>=canvas.getWidth()) || (x2<=0) ) {dx2=(-dx2);}
    if ((y1>=canvas.getHeight()) || (y1<=0) ) {dy1=(-dy1);}
    if ((y2>=canvas.getHeight()) || (y2<=0) ) {dy2=(-dy2);}
    i++;
}
```

- W pętli rysujemy 200 odcinków, za każdym razem przesuwając ich końce o wektory przesunięcia

```
try {  
    Thread.sleep( millis: 1000); // odroczenie(zatrzymanie wątku) na jedną sekundę  
}  
catch ( InterruptedException e) {  
    System.out.println("np. zosta,em obudzony przedwcześnie");  
}  
invalidate(); //polecenie odrysowania Canvas-u
```

- Aby uzyskać prostą animację możemy uśpić wątek na określony czas, a następnie wydać polecenie przerysowania Canvas-u (ponownego wykonania metody onPaint)

