

## Android- lesson 2

### תיקיית res

תיקייה זו מכילה המון תיקיות ובהם הריסורסים הדרושים לקוד, נפרט:

#### תיקיית drawable

בתוך תיקיית res נמצא חמש תיקיות שכולן מתחילות במילה drawable וכולן מיועדות לשמירת תמונות.

ההבדל בין התיקיות מתבטא ברזולוציות השונות של התמונות. אנדרואיד מזהה איזה תיקיות רלוונטיות לאיזה מכשיר - כלומר אפשר להשתמש באותה תמונה, עם אותו שם ורק עם רזולוציה שונה והאנדרואיד יודע לבד, מתוך המכשיר שבו נפתחה האפליקציה לזהות לאיזה תיקייה עליו לגשת כדי לפתוח את התמונה בתצוגה מותאמת למכשיר. העתקת תמונה לתיקיית תמונות - לשנות את השם שיתחיל ויכיל אותיות קטנות - **חובה!** R.drawable - פנייה מג'אווה לתמונה בצורה ישירה.

ב- XML הפנייה לסורס תמיד תתחיל ב- @ - ניגון - ונפתחים כל הסורסים הקיימים ובחירה.

#### scaleType

חיתוך התמונה וקביעת מקום.

#### editText

טקסט שיופיע ברירת מחדל - android:text

ואז נוסיף שינוי טקסט עם פתיחה.

יש גם מגוון - אימייל, סיסמה ובהתאם הוא פותח מקלדת מותאמת.

hint - מופיע בתוך תיבת הטקסט באפור (הקש את שמך כאן) - רמז למה שצריך להזין בתיבת הטקסט.

inputType - מכאן נבחר את סוג המקלדת שאנחנו רוצים שתעלה.

Visibility/invisibale - שומר על המקום של האלמנט כשהוא לא נמצא.

#### תיקיית values

בתוכו קובץ סטרינג שבו נגדיר משתני סטרינג שונים.

תמיכה במולטילנגוויז'. העבודה בקונספט הזה שכל הסטרינגים שמאפיינים את הטקסט שלי אז התרגום הרבה הרבה יותר קל, התרגום נעשה בקובץ הסטרינג בלבד ומשנה את הטקסט אוטומטי.

#### תיקיית Color

בתוך values נפתח קובץ חדש ונקרא לו קולור

קולור - תגית ריסורס - תגית קולור - ניתן לה שם בתוך התגית ונכתוב בין פתיחה לסגירה את הערך. אפשר להוריד באינטרנט רשימה של כל הצבעים ולהעתיק לקובץ קולור.

#### העדיפות תמיד לעבוד ב-XML ולא ג'אווה

ביצועים - XML נטען במהירות עם פתיחת האפליקציה.

- בג'אווה נשים את הדברים הדינאמיים.

#### Listener

המאזין לאירועים

בקובץ מיינ אקטיביטי ג'אווה נוסיף: (את מה שמודגש)

```
public void MainActivity extend Activity implements OnClickListener {
```

גם נרשום את הכפתור שיהיה ברשימת הנלחצים - קודם כל נזהה את הכפתור ואז נבקש: `Btn.setOnClickListener(this);` כלומר, הליסטנר מאזין לו ובגלל זה נוסיף `this`

## Switch(v.getId)

Case R.id.button1

נשתמש בסוויץ' כדי לשלוט במספר כפתורים, כאשר כל כפתור עושה משהו אחר. ואז במקרה שכפתור 1 נלחץ תעשה כך ובמקרה שכפתור שני נלחץ אז תעשה אחרת.

תמיד עדיף XML חוץ מבמקרה של און קליק. כי הקומפיילר לא מזהה 2 פונקציות באותו שם והדיבאגר של XML לא כל כך מזהה בו.

- עכשיו אנחנו מכירים 3 גישות:
- 1- XML שבה נכתוב את אוןקליק והיא לא טובה – **אסור!**
- 2- implement – אוןקליק ליסטנר.
- 3- (new View.OnClickListener()) – ובתוכו נוצר אוןקליק.

שיטות 2 ו-3 אותו הדבר.

במצב שבו יש ריבוי כפתורים עדיף שיטה 3 כי היא עוסקת בכל כפתור לבד. כשיש מעט כפתורים נשתמש ב-2 שהיא מקוצרת יותר.

## intent

החדשנות שמערכת אנדרואיד הביאה נוגעת לזיכרון המכשיר, עד כה, כאשר נפתחה אפליקציה היא רצה כולה על הראם, כולל כל הדפים שלה, מה שהעמיס מאוד על זיכרון המכשיר. באנדרואיד האפליקציה נחלקת לדפים, כל דף פועל כאפליקציה נפרדת, כאשר דף אחד רץ כל השאר מכובים והמשתנים שבהם מתים (לפעמים) אנדרואיד הוא המנהל של הזיכרון ולא המתכנת, והוא זה שמחליט מתי להשמיד משתנים על פי שיקולי זיכרון שהוא צריך להפעלה שוטפת. ולכן, על המתכנת לדעת שיש ערובה לחיי משתנים שנמצאים בדף היחיד שפתוח בזמן נתון ואין שום ערובה לחיי משתנים שנמצאים בדף אחר באותה אפליקציה- כי זה נתון לשיקולי האנדרואיד. עם זאת, קיים צורך באזור שמור ומוגן – זהו ה- **intent**.

## השלבים:

- 1- תוספות לקובץ MainActivity
- 2- יצירת מחלקה נוספת בתיקיית src ולהוסיף בה מספר דברים
- 3- יצירת xml נוסף.
- 4- עריכה קטנה בקובץ AndroidManifest

לפי הסדר אני מוסיפה צילומי מסך:

```
package com.example.tryu;

import android.os.Bundle;

public class MainActivity extends Activity {

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

```
package com.example.intentosh;

import android.os.Bundle;

public class MainActivity extends Activity {

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

        Button btn = (Button)findViewById(R.id.button1);
        btn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent intent = new Intent (MainActivity.this,SecondActivity.class);
                intent.putExtra("Kalmar", "iparon");
                intent.putExtra("food", "banana");
                startActivity(intent);
            }
        });
    }
}
```

שורה  
1

### משמעות השורות שהוספנו לקובץ זה: (עפ"י סדר הופעתן)

- 1- איתרנו כפתור עפ"י ID
- 2- אמרנו לליסטנר להקשיב לאירוע- במקרה זה, אירוע לחיצה על כפתור.
- 3- ייבוא override לפונקציית onClick
- 4- אימפלמנט לפונקציית onClick
- 5- הצהרה על intent חדש: בסוגריים כתבנו שני דברים מופרדים בפסיק:  
הראשון- מאין אני בא? שם נרשום את שמו של הקובץ הנוכחי.this  
השני- לאן אני הולך? שמה של המחלקה החדשה שיצרנו.class
- 6- העמסנו מידע על intent- מסירת המידע בסוגריים, גם כאן שני חלקים מופרדים סוגריים:  
החלק הראשון הוא המשתנה, החלק השני הוא הערך.
- 7- כמו שורה 6, רק משתנה אחר.
- 8- ציווי ל- intent- תתחיל לפעול עכשיו!

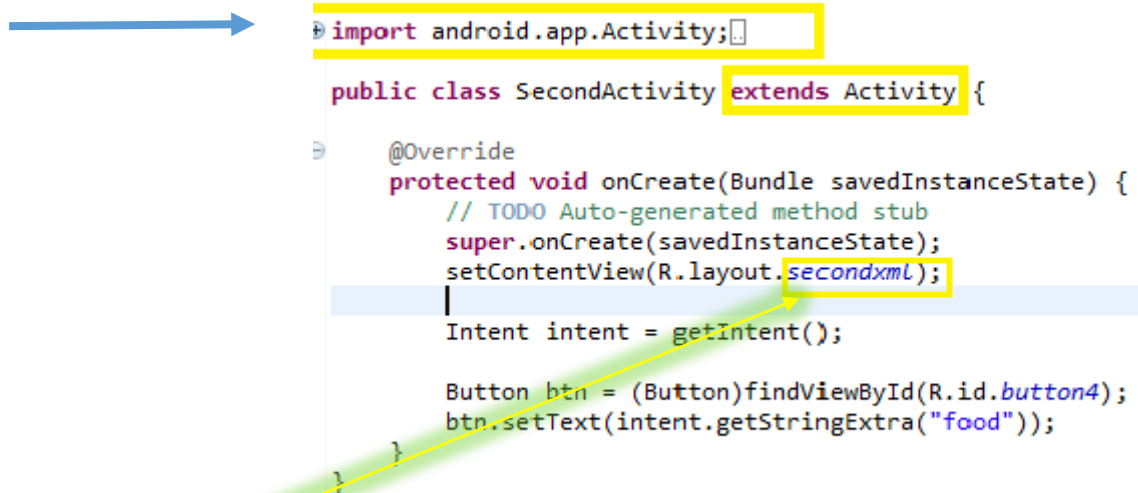
**לפני:**

```
package com.example.tryu;

public class SecondActivity {

}
```

**אחרי:**



```
import android.app.Activity;

public class SecondActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.secondxml);

        Intent intent = getIntent();

        Button btn = (Button)findViewById(R.id.button4);
        btn.setText(intent.getStringExtra("food"));
    }
}
```

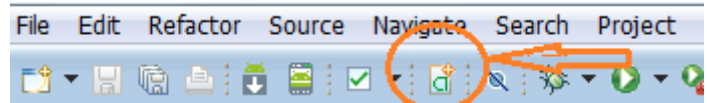
לשים לב! קובץ ה- xml המקושר לדף זה הוא קובץ ה-XML החדש שיצרנו!

### משמעות השורות שהוספנו לקובץ זה: (עפ"י סדר הופעתן)

- 1- ייבוא מחלקת אקטיביטי
- 2- הגדרת אקטיבי כמחלקת אב.
- 3- @override ייבוא של פונקציה אבסטרקטית
- 4- ייבוא בפועל של פונקציה אבסטרקטית (מוגדרת במחלקת האב כברירת מחדל)
- 5- כלום
- 6- בעת יצירה תפעיל את הקונסטרקטור של עצמך
- 7- קישור לקובץ XML
- 8- תביא את intent לכאן ותכין אותו לפריקת המידע.
- 9- איתור כפתור עפ"י ID
- 10- תמשוך מידע מה-intent, במקרה זה הערך של המשתנה food ושלח אותו לכפתור.

## יצירת xml חדש בתיקיית res/layout

נפתח את תיקיית <---res בתוכה נפתח את תיקיית <---layout בתפריט עליון נבחר:



לשים לב! בקובץ src החדש (תמונה אחת למעלה) הצהרנו על כפתור- קראנו לו button4 לא לשכוח להוסיף בקובץ XML זה את הכפתור!

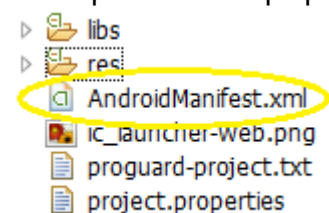
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

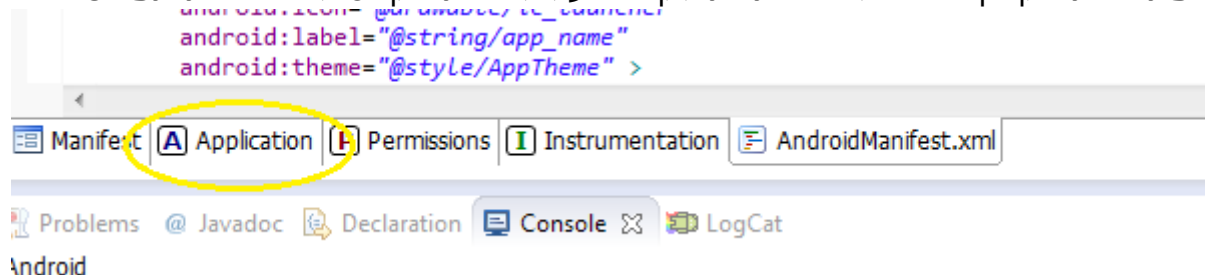
</LinearLayout>
```

## עריכה בקובץ AndroidManifest

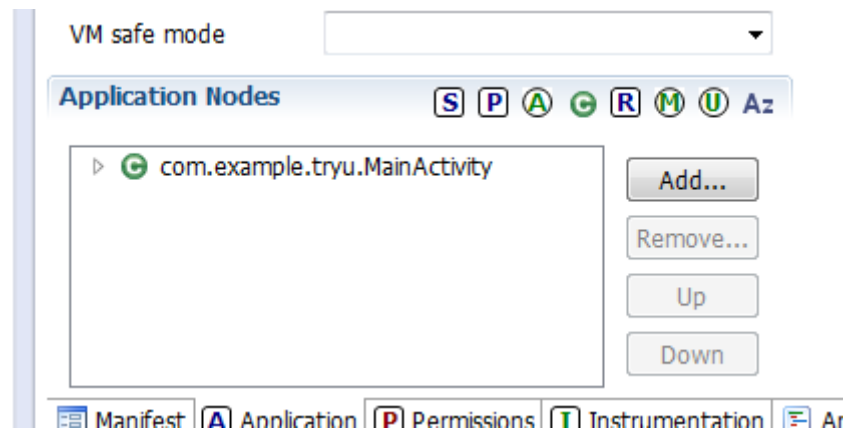
הקובץ נמצא בתיקיית res



נפתח את הקובץ ונשים לב שמתחת לקוד ומעל לשורת הקונסול יש שורת תפריט:



נבחר Application  
נגלול לתחתית הדף:



לשים לב! נסמן את השורה הזאת:



נבחר Browse --- < יפתח לנו חלון ובו נבחר את קובץ האקטיבי (java) החדש שיצרנו.

## דרך חלופית לעריכה בקובץ Manifest:

אפשר להוסיף ידנית בקובץ ה- AndroidManifest את השורה הבאה: לשים לב למיקום בין התגיות!

```

<manifest>
    <application>
        <activity
            android:theme="@style/AppTheme" >
                <activity
                    android:name="com.example.intentosh.MainActivity"
                    android:label="@string/app_name" >
                        <intent-filter>
                            <action android:name="android.intent.action.MAIN" />

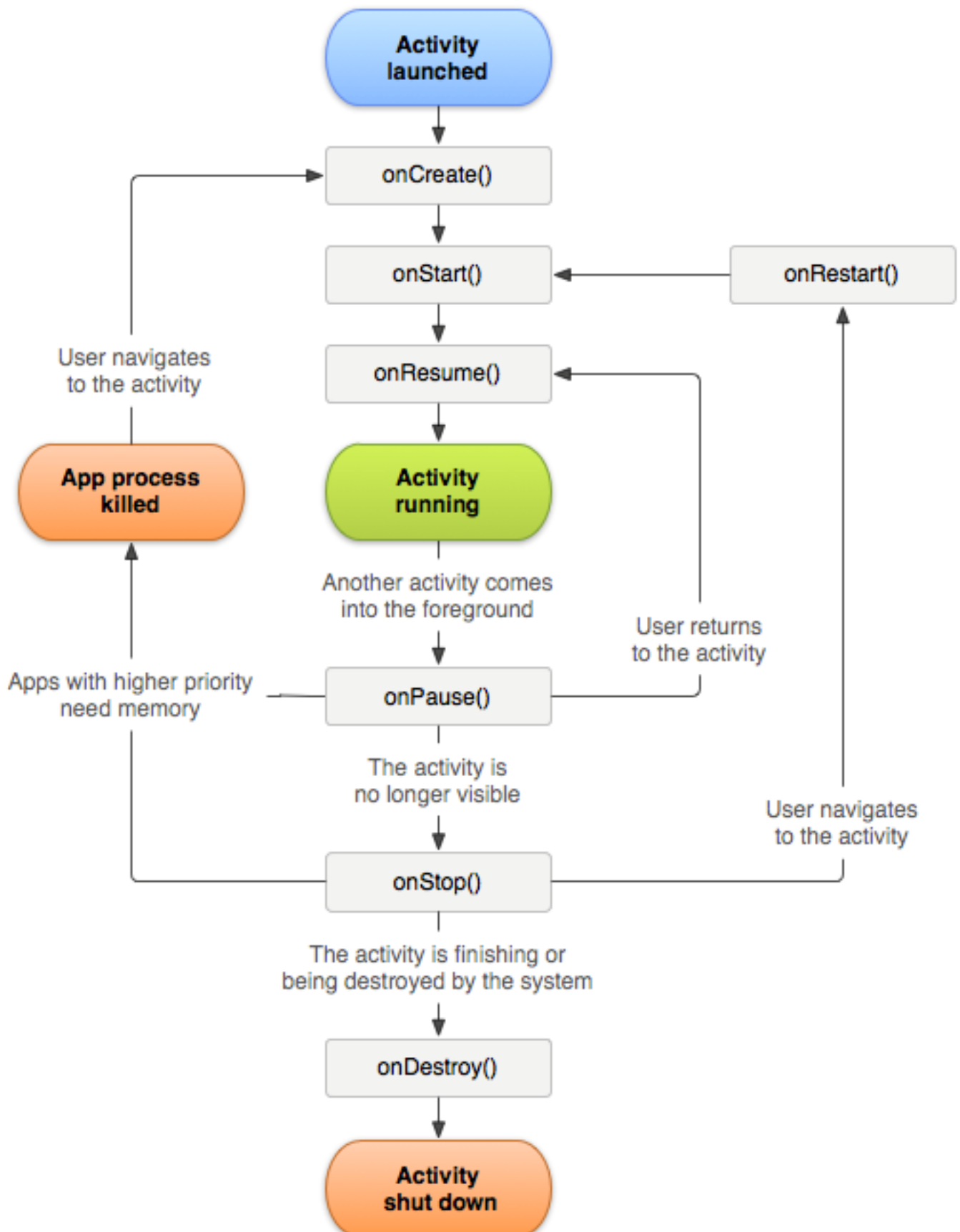
                            <category android:name="android.intent.category.LAUNCHER" />
                        </intent-filter>
                    </activity>
                    <activity android:name="SecondActivity"></activity>
                </application>
            </manifest>

```

מה שקרה בקובץ זה- הוספת שורת קוד, משמשת כדי שהוא יזהה שנוספה מחלקה חדשה.

## הסבר תאורטי למה שעשינו עד כה:

לכל activity יש מחזור חיים המורכב ממספר שלבים- מחזור החיים בא להמשיך שלא ידוע מתי המשתנה הימות, ועל המתכנת לדאוג לשמור על משתנים חשובים ב- intent.



## משפטים נבחרים מהשיעור:

אחת מפונקציות האקטיביטי היא getIntent

**Activity** מורכב מ- layout שהוא XML.

מחסנית לייאאוטסטאק- כל לייאאוט שפותחים נכנס למחסנית זו.

```
Intent intent = getIntent();
```

"לא יכול לקבל מה שאין לך!" --- מאיפה באה הגטאינטנט?!!?!?

אהאהאה!!! מצאנו אותה באקטיבי והתפקיד שלה- מחזירה את האינטנט שהתחילה את האקטיביטי הזה.

בכדי לשלוף את המידע בדיוק בפורמט שצריך יש לנו את גטסטרינג ובסוגריים את שם התגית. כך:

```
String name = intent.getStringExtra("Name");
```

וכך שלפנו את המידע והצבנו אותו למשתנים החדשים שהגדרנו כעת.

כשעושים גטסטרינג אקסטרא

גטסטרינגאינט הוא ייתן את ברירת המחדל אם לא רשמנו נכון את שם המשתנה.

ועכשיו נדפיס את המידע:

```
tx.setText();
```

והוא מדפיס את ערך המשתנים.

התפיסה במתכנתים היא ההבנה שקים רק מה שנמצא קדימה על המסך ועל המתכנת לדאוג למידע שנמצא מאחורה.

```
Toast.makeText(MainActivity.this.....
```

מעלה רק לרגע.

זה שהאפליקציה מתחבאת לא אומר שהיא מתה

ההצהרה- אנדרואיד צריך לדעת על קיומו- מאניפסט הוא עמוד השדרה של האנדרואיד.