



הקדמה ל-Android

안דרואיד זו מערכת הפעלה מבוססת לינוקס אשר מיועדת למכשירים ניידים ולמכשירים אשר בח מה חשוב שלהם קטן במיוחד.

מערכת הפעלה אנדרואיד נרכשה מחברת אנדרואיד על ידי גול ב-2005. בשנת 2007 הוכרז על הקמתה של ברית חברות (OHA- Open Handset Alliance), שטרתה פיתוח סטנדרטים פתוחים למכשירים ניידים. הפיתוח הראשון עליו הכריזה ה-OHA הייתה מערכת הפעלה אנדרואיד המבוססת על לינוקס קרנל גרסה 2.6. בפברואר 2009 הופצה הגירסה הראשונה לשימוש כלל של לקוחות.

תוכניות עבור אנדרואיד נכתבות בדרך כלל בשפת Java.

לצורך יצירה אפליקציות באנדראוד, משתמש בסביבת הפיתוח החדשה ביותר:

.Android Studio

צעדים ראשוניים

לצורך יצירה פרויקט חדש נבצע את הצעדים הבאים:

File -> New -> **New Project...**

Application name:	שם האפליקציה, מתחילה באות גדולה
Company Domain:	user.example.com
Package name:	com.example.user.myapplication
Edit	
Project location:	C:\Users\user\AndroidStudioProjects\MyApplication2

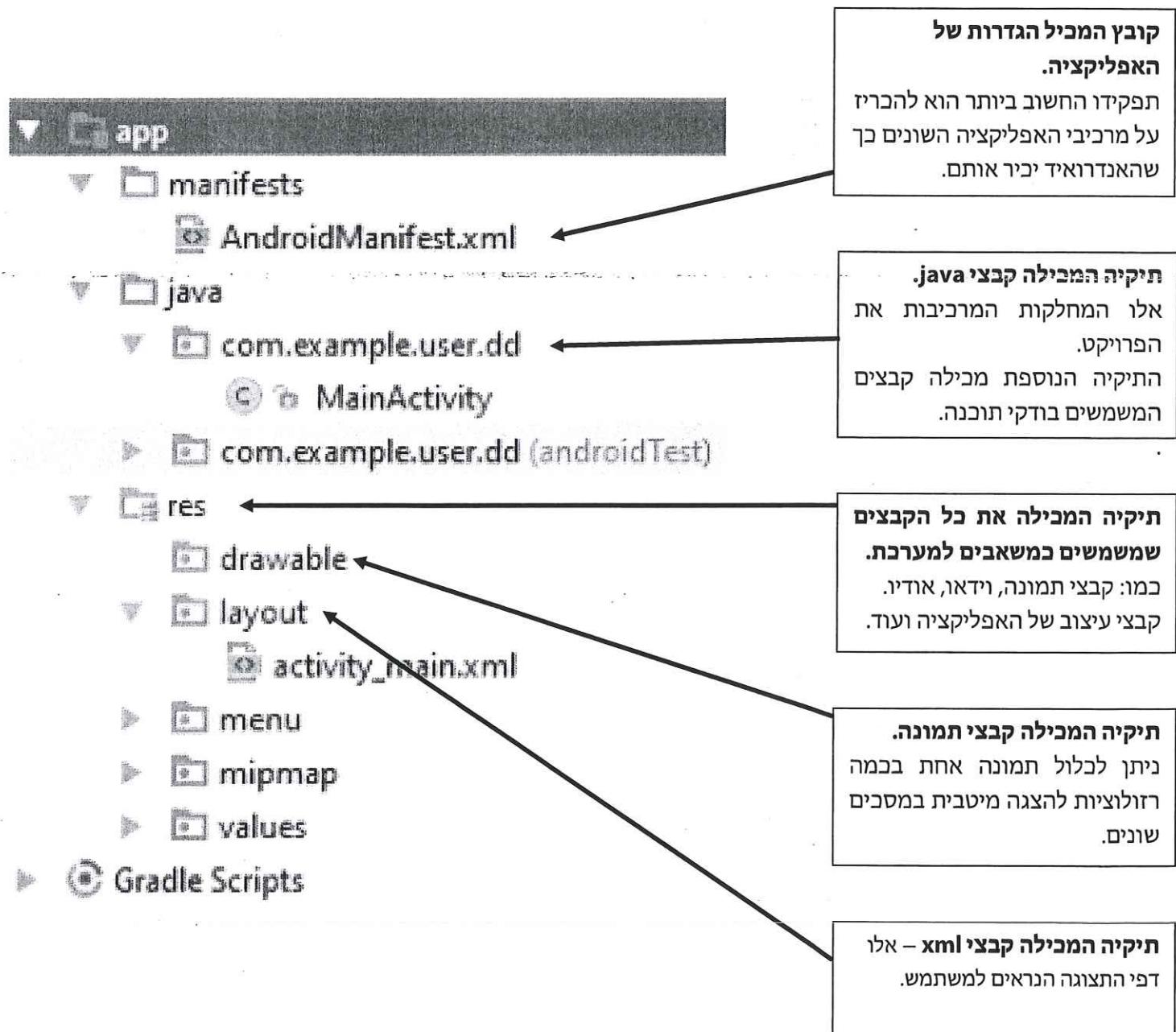
בחולון שנפתח נבחר שם לאפליקציה. השם חייב להתחיל באות אנגלית גדולה.

בשלבים הבאים נוכל לבחור בהגדרות מתקדמיות, כמו תבניות מוכנות שימושיות לנו לבנות את האפליקציה, הטקסט שיוצג ככותרת בדף האפליקציה ועוד.
בעת לחיצה על **Finish** יוצר הפרויקט.



בסייטה דשמיा

מבנה הפרויקט





מרכבי אפליקציה באנדרואיד

אפליקציה באנדרואיד מורכבת מארכבה **מרכבים עיקריים**, אלו הן מחלקות אותן נוכן לרשות כדי ליצור אפליקציה תקינה.

Activities	מחלקה המנהלת את המשק המוצג למשתמש.
Services	מחלקה המנהלת הפעולות ברקע הקשורות ל애�ליפצייתך.
Broadcast Receivers	מחלקה המחברת בין מערכות הפעלה לאפליקציות.
Content Providers	מחלקה המנהלת מידע ובסיסי מידע.

Activities

מחלקה היורשת ממחלקה **Activity** זהה למחלקה המייצגת מסך המוצג למשתמש. (אקטיביטי)

כל אפליקציה באנדרואיד מורכבת מ- **activity** אחד או יותר, שהם בעצם מסכים שונים, שהמשתמש עובר ביניהם.

אחד מהוות את נקודת הפתיחה לאפליקציה, שהוא בעצם המסך הראשון שираה המשתמש. את שמו אנו מגדירים ביצירת הפרויקט. בירית מחלול – **MainActivity**.

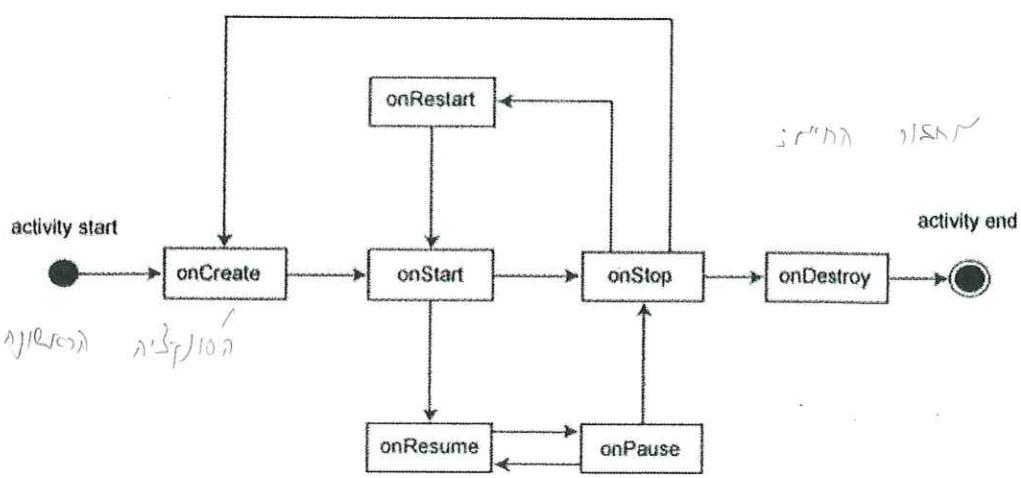
לדוגמא: אפליקציה לניהול ספרייה, מכילה **Activity** אחד שבו יש את פרטי המנויים. **Activities** נוספים לצורך רישום מנוי חדש, ועוד **Activity** שבו מבנים פרטי ספר להשאלה. וכן **Activities** נוספים.

במחלקה בותבים רק את הקוד התכונתי, בעוד את התצוגה של המסך נגדיר במסמך XML בתיקיה **.layout**.

מחזור החיים של Activity

לבב למחלקה היורשת מ- **Activity** יש מספר פונקציות אותן ניתן להשתמש למשתמש, והן מהוות את "מחזור החיים" של **activity**, בולם, הפונקציות מתבצעות אחת אחרי השנייה, מהרגע שה액טיביטי נכנס לפעולה ועד לסיוםו.

← אפשר להוסיף אותן דרך התפריט **(Ctrl+O) .Code -> Override Methods...**





הfonקציה (OnCreate)

הfonקציה OnCreate היא פונקציה המתבצעת פעם אחת בלבד בעת **יצירת ה-activity**.

כאשר המשמש עבור בין activity ל-activity המופיע קודם נשמור בזיכרון ובאשר יחזרו אליו הוא לא יוצר מחדש.

בתוכה נכתבת כל הקוד שנרצה שיקרא בעת **יצירת ה-activity**.

את הקישור בין קובץ התצוגה לקוד נבצע גם בfonקציה זו באמצעות שורת הקוד:

```
SetContentView(R.layout.main)
```

בתוך הסוגרים נרשום R.layout ונוסיף את שם דף התצוגה אחרי הנקודה. בדוגמה שמו הוא .main

שאר הfonקציות:

OnStart()	fonקציה המתבצעת לפני שマーך ה-Activity נראה לעין המשתמש, לאחר יצירתו.
OnResume()	fonקציה המתבצעת בעת שマーך ה-Activity נראה לעין המשתמש.
OnPause()	fonקציה המתבצעת כאשר המשתמש בוחר לעבור ל액טיביטי אחר. ה-Activity הנוכחי נמצא בהפסקה.
OnStop()	fonקציה המתבצעת כאשר ה-Activity הנוכחי כבר לא מופיע יותר.
OnDestroy()	fonקציה המתבצעת לפני שה-Activity "מחוסל" ע"י המערכת.
OnRestart()	fonקציה המתבצעת במידה שה-Activity חזר לחיים אחרי הfonקציה .OnStop()



עיצוב דף ה-Layout

בأنדרואיד ממשק המשתמש מוגדר בקובץ XML מיוחד המותקשר לדף ה-Activity. לכל דף XML קימת בעין "משטח" עליו מונחים הפקדים השונים – הכפתורים, תיבות הטקסט וכו'.

והיא בעצם מדירה את צורת הפרישה של הפקדים. ViewGroup משטח זה יורש מהמחלקה .ViewGroup בעת יצירת Activity חדשה, נוצר אוטומטית דף Layout המותקשר אליו.

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

כדי ליצור דף Layout חדש באופן ידני, נלחץ להצן ימני על התיקיה "layout": New->Layout resource file שם קובץ ה-Layout באיכות קענות בלבד.

קייםים סוגים שונים של Layouts עליהם נפרט בהמשך.

פקדים

כדי להוסיף פקדים לדף התצוגה (ה-Layout), קימות 2 אפשרויות.

1. בחירת הפקד מהתיבה לצד שמאל של המסך, וגרירתו למקום הרצוי בדף.
2. כתיבת שמו של הפקד בתוך תגית בדף ה-xml.

TextView

פקד זה מציג טקסט למשתמש.

← נוכל לבחור אותו מהתיבה:

Widgets

- [Ab] Plain TextView
- [Ab] Large Text
- [Ab] Medium Text
- [Ab] Small Text

בחירת "Plain TextView" תציג לנו את הפקד כאשר הטקסט שבתוכו בגודל סטנדרטי. לחילופין, אפשר לבחור "Large", "Medium" או "Small" וכך לקבל TextView שהtekst שבתוכו בגודל ענק, בינוני או קטן.

← נוכל לכתוב את שמו בתוך תגית בדף ה-xml.



```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:text="Large Text"      />
```

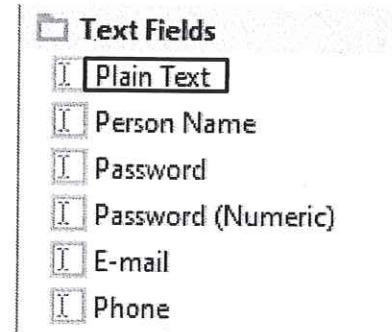
תכונות הפקד:

android:id	קוד הפקד.
android:text	הטקסט שיוצג בפקד.
android:textsize	גודל הטקסט. לדוג'ה: 30dp
android:textcolor	צבע הטקסט. את הצבע מצינים באמצעות סולמית ואות'B' מספר. לדוג'ה: #333333 זהו צבע אפור.
android:textstyle	סגנון הטקסט. רגיל – normal, מודגש - bold, נטוי – italic. אפשר יותר מסגנון אחד באמצעות שימוש בקו מפריד. לדוג'ה: bold italic
android:fontFamily	גופן הטקסט.
android:drawableBottom	תמונה שתוצג מתחת לינקסט. במו"ב יש אפשרות לתמונה מעל הטקסט, לדוגמה ולשםallo. דוג'ה: android:drawableBottom="@drawable/ic_launcher"

EditText

פקד זה מאפשר למשתמש להזין טקסט.

← נוכל לבחור אותו מהתיבה:



בחירת "Plain Text" תציג לנו את הפקד בצורתו הבסיסית עם אפשרות להזנת טקסט.

אפשרות עוד אפשרות רבות.

בחירת "Password" תציג לנו פקד המאפשר הזנת סיסמה, אשר רושמים את הסיסמא, נרשמים נקודות במקום התווים.

במו"ב יש תיבת המשלימה אוטומטית טקסטים מוכרים, תיבת המכילה שורות רבות ועוד.

← נוכל לכתוב את שמו בתוך תיגית בדף ה-xml.



Android

בסיועו של שמי

```
<EditText  
    android:id="@+id/editText1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
  
    </EditText>
```

תכונות הפקד:

android:id	קוד הפקד.
android:text	הטקסט שיוצג בפקד.
android:autoText	מזהה שגיאות כתיב ונותן אפשרות תיקון. (באנגלית)

במור"כ יש עוד תכונות רבות זהות לפקד `TextView`.

Button

זהו פקח "כפתור" המאפשר לחיצה עליון.

← נוכל לבחור אותו מהתיבה:



בחירה 'Button' תציג לנו את הפקד בגודלו הסטנדרטי.

בחירה 'Small' תציג לנו בפתחו קטן יותר. בבחירה "ToggleButton" תציג לנו בפתחו של לחיצה עליון משנה את מצבו ל-ON והפס הדק בחלקו התחתון נדלק, ובאשר לוחצים שוב הפס הופך ל-OFF והפס הדק נכבה.

← נוכל לכתוב את שמו בתוך תגיית בדף ה-xml.

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:text="Button" />
```



תכונות הפקד:

android:id	קוד הפקד.
android:text	הטקסט שיוצג בפקד.
android:background	רקע הפקד. צבע/תמונה.
android:onClick	שם הפונקציה בקובץ Java המקשר ל-Layout אליה הגיע כאשר נלחץ על הבפתור. שם הפונקציה ללא סוגרים עגולות אחרת. לדוגמא: "func". android:onClick="func"
android:visibility	אם הפקד נראה למשתמש או לא. נראה – visible. לא נראה – invisible.

במור"כ יש עוד תכונות ובות זהות לפקד TextView.

ImageView

פקד המאפשר הצגת תמונה.

← נוכל לבחור אותו מהתיבה:

- ImageButton
- ImageView

← נוכל לבתוב את שמו בתוך תגיית בדף ה-.xml.

```
<ImageView  
    android:id="@+id/imageView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_launcher" />
```

תכונות הפקד:

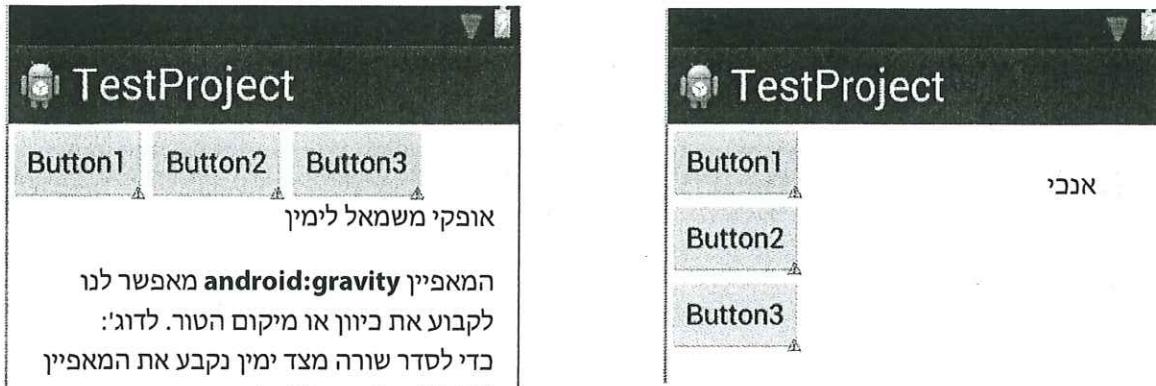
android:id	קוד הפקד.
android:src	מקור התמונה. ניתן לשולף אותה מהתיקייה drawable. עיין בדוגמה.



Layouts

LinearLayout

בצורת פרישה זו הפקדים מסודרים אחר ליד השני, בטור אופקי או אנכי.



בדי לקבע את צורת הפרישה נשתמש במאפיין **.android:orientation** (ברירת מחדל)

אופקי: **android:orientation="horizontal"** (ברירת מחדל)

אנכי: **android:orientation="vertical"**

נוכל לקבע שצורת הפרישה תהיה אופקית, ואות 'ב' לשנות לאנכית והפקדים ישודרו בצורה אוטומטית.

בצורת פרישה זו קיים עוד מאפיין חשוב הנקרא **.android_weight** בעזרת מאפיין זה נוכל לקבע בעין "משקל" לכל פקד, שהוא בעצם מספר יחסילסה'ב יחידות המשקל שניתנו לפקדים.
במקרה שצורת הפרישה היא אופקית – חשוב לקבע את המאפיין רוחב הפקד **width=0dp**, לכל פקד המסודר לפי משקל, כדי שהצגוה תסתדר באופן אוטומטי.

width=0dp

במקרה שצורת הפרישה היא אנכי – נקבע את המאפיין גובה הפקד:

height=0dp

```
<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"
```

```
    tools:context="com.example.user.myapplication.MainActivity5">
```

```
        <TextView
```

```
            android:layout_width="0dp"
```

```
            android:layout_height="wrap_content"
```

```
            android:text="Hello World!"
```

```
            android:weight="2"
```

```
        />
```

```
        <TextView
```

```
            android:layout_width="0dp"
```

```
            android:layout_height="wrap_content"
```

```
            android:weight="1"
```

```
        />
```

```
    </LinearLayout>
```

בדוג', הפקד הראשון תופס 2/3 מהמסך והשני 1/3.



RelativeLayout

בצורת פרישה זו הפקדים לא מסודרים בצורה אוטומטית אלא לפי מה שנקבע בקובץ ה-.xml.
אפשר להציב פקדים במרכז המסקן, צמוד לחלק העליון של המסקן, צמוד לפקד אחר, מימין
לפקד אחר ועוד.
כל פקד נקבע如此:

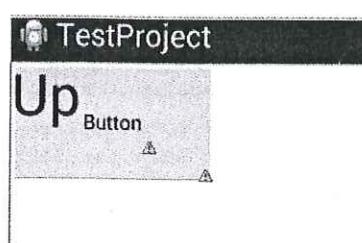
לדוגמה - android:id="@+id/button3"

ובאשר נרצה להציב פקד ליד פקד אחר נשתמש בכך זה ליזיהו הפקד.

android:layout_above="@+id/button2"	ממקם את הפקד מעל פקד שהא' שלו הוא .button2
android:layout_below="@+id/button2"	ממקם את הפקד מתחת לפקד שהא' שלו הוא .button2
android:layout_alignLeft="@+id/button2"	מיישר את הפקד לקו שמאל של הפקד שהא' שלו הוא .button2
android:layout_alignRight="@+id/button2"	מיישר את הפקד לקו ימין של הפקד שהא' שלו הוא .button2
android:layout_centerHorizontal="true"	ממקם את הפקד במרכז המסקן לרוחבו.
android:layout_centerVertical="true"	ממקם את הפקד במרכז המסקן לאורכו.
android:layout_marginLeft="50dp"	ממקם את הפקד במרחק מדויק, 50dp מהצד השמאלי של המסקן.
android:layout_marginTop="50dp"	ממקם את הפקד במרחק מדויק, 50dp מהצד העליון של המסקן.

FrameLayout

בצורת פרישה זו הפקדים יכולים לעמוד שכבות אחד מעל השני.



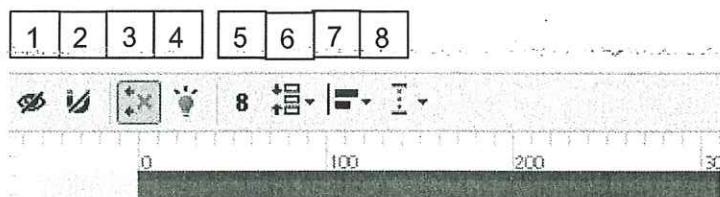


ConstraintLayout

ConstraintLayout היא צורת פרישה מתקדמת, המגדירה אילוצים לפקדים. היא בעצם משלבת בתוכה צורות פרישה שונות. כמו"כ ניתן להמיר כל צורת פרישה ל-ConstraintLayout הפקדים יסתדרו לפי אילוצים שנקבע להם, בצורה המזכירה את צורת הפרישה RelativeLayout.

ניתן לקבוע אילוצים ע"י גירסה של נקודות הקבועות בכל פקד, לנקודות ע"פ פקד אחר, וכן ניתן להסיר אילוצים ע"י לחיצה על נקודה הקשורה לאילוץ מסוים. כמו"כ ניתן לקבוע או להסיר אילוצים ע"י תגיות ה-*אוֹחָם* כפי שמופיע בטבלה בהמשך.

באשר נקבע את צורת פרישת הדף ל-ConstraintLayout, תתווסף שורת כפתורים לחלק **העלון של דף ה-Layout**:



1 – אפשרות לראות/הסתרת האילוצים.

2 – כפתור ה"מגנט" קובע אילוצים לפקדים לפי הצורה בה מיקמו אותם. לדוגמה, אם נמקם פקד לימין פקד אחר, מיד יקבע קשר בין שמאלו של הפקד שמיקמו לימין הפקד الآخر.

3 – מאפשר לבטל אילוצים שקבענו.

4 – כפתור זה מאפשר קביעת אילוצים מתאימים לכל הפקדים לפי השערת המערכת. באשר נלחץ על כפתור זה אוטומטית יתווסף אילוצים לכל הפקדים.

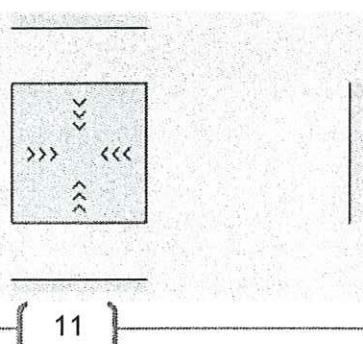
5 – קביעת מרוחה בריית מבדיל בין הפקדים באשר נסדר אותם בקבוצות לאורך/לרוחב וכו'.

6 – אפשרות לסדר הפקדים בקבוצות. לאורך, לרוחב ועוד.

7 – ישר הפקדים לימיין, לשמאלי, לאמצע המספר וכו'.

8 – הוספה קווי אורך/רוחב באמצעות המאפיינים של הפקד.

כמו"כ ניתן לטפל בגודלים של הפקדים וביחס שלהם לפקדים אחרים וכן לדף בו בamusות הכלים המופיע ברשימה המאפיינים של הפקד.





להלן יובאו חלק ממאפייני האילוצים:

layout_constraintTop_toBottomOf="@+id/button2"	مالץ את חלקו העליון של הפקד לעמוד מתחת לפקד שהפכו שלו הוא 2 button2.
layout_constraintLeft_toRightOf="@+id/button2"	مالץ את חלקו השמאלי של הפקד לעמוד מימין לפקד שהפכו שלו הוא 2 button2.
layout_constraintLeft_toLeftOf="@+id/button2"	مالץ את חלקו השמאלי של הפקד לעמוד משמאלו לפקד שהפכו שלו הוא 2 button2.
layout_constraintBaseline_toBaselineOf="@+id/button2"	مالץ את קו בסיס הפקד להתיישרuko בסיס הפקד שה-pו שלו הוא 2 button2.
layout_constraintRight_toLeftOf="@+id/guideline5"	مالץ את חלקו הימני של הפקד לעמוד משמאלוuko עד שהגדרנו.

הערה בלאית וחשובה: לכל אלמנט חיבים להגדיר מאפייני אורך ורוחב:

android:layout_width
 android:layout_height

מאפיינים אלו מקבלים 4 אפשרויות:

match_parent	מתאים את עצמו לרוחב/אורך הדף לפי הפקדים הקיימים.
wrap_content	מתאים את עצמו לאורך/רוחב של התוכן שלו. לדוגמה: טקסט של בפתור.
fill_parent	מתאים את עצמו לכל רוחב/אורך הדף.
30px/20dp	אורך ורוחב קבוע. אפשר להשתמש במידות של פיקסלים או אינצ'ים ועוד.

פונקציות

כasher נרצה לכתב פונקציה, נכתב אותה בקובץ Java.

כל פונקציה שמתבצעת באשר לוחצים על כפתור, או על כל פקד אחד **חייבת לקבל משתנה מסוג View**. (שים לב: קיימת רגשות לאותיות גדולות/קטנות, לבן View באות גדולה בהתחלה. בתיבת View תיצור שגיאה)

לדוגמא:

Public void func(**View v1**)

```
{  
    גוף הפונקציה  
}
```



משתנה זה בעצם מכיל את הכפתור או הפקד שלחצו עליו.

הגדרת הפונקציה בלחיצה על כפתור

לכל פקד שמקבל מאפיין בשם android:onClick בשם android:onClick נובל להציג פונקציה שתתבצע כאשר נלחץ עליו.

לאחר הסימן "=" נציין את שם הפונקציה שבקובץ ה-Java הקשור ל-Layout הרצוי.
לדוגמא: android:onClick="func"

אין להוסיף לאחר שם הפונקציה סוגרים עגולות!

באשר נרצה להשתמש בפקד דרך הקוד נctruck לבצע המרה.
לדוגמא, אם הפקד שלחצנו עליו הוא בפתחו נבצע המרה כך:

```
Button b=(Button)v1
```

יש לשים לב שיש רגשות, לאותיות גדולות/קטנות. לכן לא יכתבו button אלא Button.

findViewById

באשר נרצה "למצוא" פקדים שלכארה קיימים רק בדף ה-Layout, ולא לחצנו עליהם, נשתמש בפונקציה הבניה findViewById.

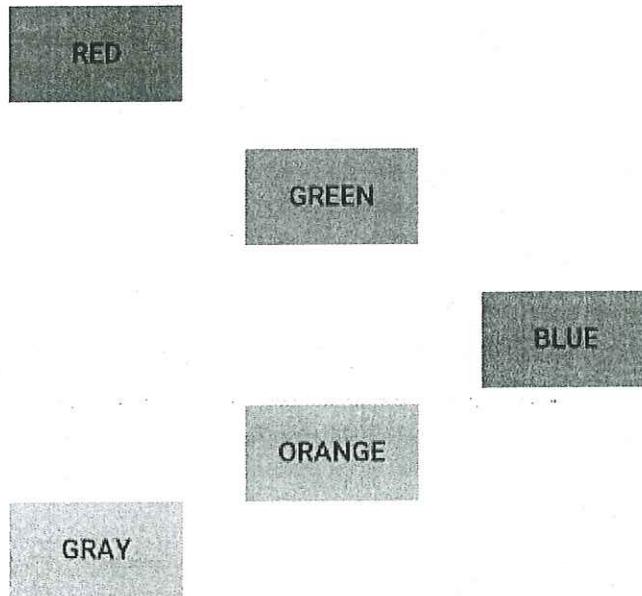
הפונקציה מחזירה את הפקד, שאט ה-ID שלו נכנס בסוגרים. הפקד המוחזר הוא מסוג View ולכן נדרש BD"ב לבצע המרה לסוג הפקד החדש.

לדוגמא:

```
Button b=(Button)findById(R.id.button1)
```

מדובר בפתחו שה-ID שלו הוא button1. נשלף את ה-ID בצורה הנ"ל.

תרגיל RelativeLayout



1. צרי אפליקציה המכילה דף בצורת הפרישה `RelativeLayout`, יש ליצור את העיצוב באופן ייחסי, כך שבחזזה של הכפתור האדום, יוזו אותו בכל שאר הכפתורים.

הגדיר מרווחים מתאימים בין הכפתורים כך שייתפרסו על המסך כפי שמוספי בתמונה.

2. קשרו את כל הכפתורים לפונקציה אחת, בעת לחיצה על כפתור, הוא יתווסף `(v.setVisibility(View.INVISIBLE);)` ל- `ArrayList` מסוג `View`, ויעלם מהמסך.

כאשר נלחץ על הכפתור האחרון, יתגלו בחזרה כל הכפתורים, ייצבו בצדע של הכפתור האחרון שנלחץ. (צריך לקבל את הצבע לפי הטקסט שעל הכפתור)



Intent

מהו Intent?

Intent הוא אחד מהמרכיבים הכי חשובים באנדรอยיד. Intent הוא בעצם אובייקט המשמש להעברת הודעות במערכת.

שימושי Intent מגוונים:

1. **הפעלת דף Activity.** זהו השימוש הנפוץ ביותר. בעזרת ה-Intent נוכל בעצם לעבור בין מסכים באפליקציה.
2. **הפעלת Service.**
3. **הפצת הודעות Broadcast Receiver במערכת.** מחלוקת Broadcast Receiver מטפלת בהודעות המופצות, שבעצם מודיעות על אירועים מיוחדים במערכת. כמו: שינוי במצב הסוללה, שינוי בשעון ועוד.

קיימים 2 סוגי Intent:

1. Intent – **Explicit Intent**, שבו בתובת היעד מצוינת במפורש.
2. Intent – **Implicit Intent** מרומז, שבו בתובת היעד לא מצוינת במפורש, אלא היעד נקבע ע"י מסנן המערכת – Intent Filter – לפי פרמטרים מסוימים.

בשלב זה נלמד כיצד לעבור לדף Activity חדש בעזרת **Explicit Intent** כאשר דף ה-Activity אליו נרצה לעבור יכתב בצורה מפורשת.

כיצד נעבור לדף חדש?

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    public void Next(View v)  
    {  
        Intent in = new Intent();  
        // נזכיר מופיע חדש של Intent // Intent  
        // נגיד ל-Intent לעבר מהדף הנוכחי - this, לאחר מכן אחרית של Intent  
        // שם מחלוקת Java של הדף אליו נרצה לעבור. ולא הדף .layout.  
        in.setClass(this, NewPage.class);  
        // הפעלת ה-Intent לפי ההגדירות שנתנו, עבור לדף המתאים/  
        startActivity(in);  
    }  
}
```



העברת נתונים באמצעות Intent

לעתים נרצה להעביר נתונים לדף אחר. לדוגמה:
 בנו דף המבקש להכניס שם משתמש וסיסמה, ונרצה להציג את שם המשתמש ב-`TextView`
 בעמוד אחר.
 מכיוון שאפשר לשולף את הנתון רק לדף הנוכחי, נדרש לשולח את הנתון בעזרת ה-`Intent`.
 השתמש בפונקציה המובנית `putExtra`.
 בעזרה פונקציה זו נוכל לשולח את כל סוגי הנתונים (integer, string וכו') לדף אחר.

שליחת נתונים:

```
public class MainActivity extends AppCompatActivity {

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

    public void Next(View v)
    {
        Intent in = new Intent();
        in.setClass(this, NewPage.class);
        in.putExtra("id1", 123); //intent
        startActivityForResult(in);
    }
}
```

הסבר הדוגמא:

"`id1`" – קוד הנתון. כך נדע איזה נתון לשולף בדף שנעביר אליו.
 123 – זה הנתון אותו אנו מעבירים. סוג `integer`.

שלילת הנתון:

כדי לשולף את הנתון בדף הבא, ניצור `Intent` חדש, לתוכו נקבע את ה-`Intent` שאותו העברנו
 את הנתון.

```
public class NewPage extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.Page2);

        Intent dugma=getIntent(); //intent
        //שלוף מתוכו את ה-int שלחו כאשר: "id1" זהו הקוד של הנתון,
        //ו-0 מספר ברירת מחדל במקרה של תקלה //
        Int i=dugma.getIntExtra("id1", 0);
    }
}
```

אין זיין צעיף



אם נרצה לשולף String נשימוש בפונקציה:
 getStringExtra()
ובן הלאה לשאר סוגים הנחוצים.

startActivityForResult()

לעתים נעבור לדף הבא, ונרצה לחזור לדף הקודם ממנו עברנו.
כאשר נעבור לדף הבא באמצעות הפונקציה startActivityForResult נוכל לחזור לדף הקודם
וללחזר "תוצאה".

```
public class MainActivity extends AppCompatActivity {

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

    public void Next_with_Result(View v)
    {
        Intent in=new Intent();
        in.setClass(this,NewPage.class);
        startActivityForResult(in,1);
    }
}
```

הסביר הדוגמא:

in – שם ה-intent
1 – "קוד בקשה", כלומר כך נזהה בהמשך מאייה דף חזרנו.

← לאחר מכן, נוסיף את הפונקציה **onActivityResult** (נתחיל לרשום אותה ותכתב
אוטומטית).
לתוך פונקציה זו נחזיר מהדף הבא.



פונקציה זו מקבלת שלושה פרמטרים:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);
```

זהו קוד הבקשתו, אותו שלחנו.

התוצאה אותה החזרנו.

משתנה ה-intent אותו החזרנו מהדף הקודם. מכיל בתוכו נתונים שנשלחו אליו.
}

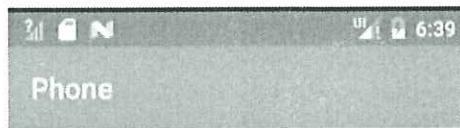
דף הבא, אשר נרצה לחזור לדף הקודם, נרשום את השורות הבאות:

```
public class NewPage extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.Page2);  
    }  
  
    public void Return(View v)  
    {  
        //נדיר את ה"תוצאה" שנרצה להחזיר לדף ממנו הגיעו וכן intent שוביל מידע (אם נרצה)//  
        setResult(RESULT_OK, intent2);  
  
        //שרה זו תעביר לדף הקודם  
        finish();  
    }  
});
```

- התוצאה שאנו מחזירים. אפשר גם:
RESULT_CANCELED, RESULT_FIRST_USER

- משתנה אליו אנו מעבירים נתונים לדף הקודם.
(putExtra)

תרגיל 2



1 2 3

4 5 6

7 8 9

* 0 #



צרו 2 דפים.

1. דף הראשון יציגו כפתורים להקשת מספר טלפון. בעת לחיצה על מספר, הכפתור הנלחץ **בלבד** יצביע בכחול (שומרים את הכפתור הקודם במופע של Button וכל פעם מוחזרים אותו לצביע השקוף) ויתווסף למחוזת בלשניה. (שלוח את כולם במאפיין onClick לאותה פונקציה)
2. לאחר שהקשנו את המספר כולו, לחיצה על הכפתור תעביר את המספר (המחוזת) לדף השני. באמצעות Intent, נציג אותו על הדף השני.

הארות:

א. דאגו לצורת פרישה מתאימה. (הכי טוב—**TableLayout**)

ב. כפתורי המספרים יהיו מוקושרים לאותה פונקציה.

3. דף השני יופיע המספר שהוקש על TextView. הוסיף תיבת טקסט, ובפטור Back.



אירועים – Events

מהו אירוע?

כאשר משתמש לוחץ על כפתור, מキー על המקלדת, נוגע במסך וכו', הוא בעצם מעורר אירוע המתאים לאוֹתָה פֻּולָּה.

כל אירוע קיים Listener – מקייב לאירוע. כאשר אירוע מסוים קורה, ה-Listener מעורר פונקציה בתוכה יכתב הקוד שנרצה שיקורה.

דוגמא:

בעת לחיצה על כפתור, נרצה שركע הכפתור ייצב באדום.

קיימים מספר רב של אירועים. בטבלה הבאה יוצגו השימושיים שבהם.

הסבר	שם המקייב לאירוע	שם האירוע
אירוע המתעורר בעת לחיצה על פקק.	OnClickListener()	onClick()
אירוע המתעורר בעת לחיצה ארוכה על פקק.	onLongClickListener()	onLongClick()
אירוע המתעורר באשר פקק מסוים מקבל/מאנך את הפוקוס.	onFocusChangeListener()	onFocusChange()
אירוע המתעורר בעת לחיצה בפקק על מושך במקלדת/עכבר.	onKeyListener()	onKey()
אירוע המתעורר בעת נגיעה בפקק מסוים.	onTouchListener()	onTouch()
אירוע המתעורר בעת גירית פקק.	onDragListener()	onDrag()

כדי שנוכל להשתמש באירועים, נצטרך לחבר את הפקד לאירוע המתאים. וליצור את הפונקציה המתאימה לאירוע. אחרת, גם אם נלחץ על הפקד, או נגרור אותו, לא יתעורר האירוע, ובעצם לא יקרה כלום.

רישום פקד ל-EventListener

קיים 2 דרכי נפוצות "לרשום" – לחבר את הפקד לאירוע.

1. שימוש ב-Anonymous Inner Class

משתמשים במחלקה אונונימית כדי "לרשום" פקד אחד לאירוע מסוים. לדוגמה:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        מציאת הcpfator שה-id שלו הוא button1
        Button b = (Button) findViewById(R.id.button1);
        רישום הפקד למקיב לאירוע לחיצה:
        b.setOnClickListener(new OnClickListener() {
            פונקציה המתואימה לאירוע לחיצה:
            public void onClick(View arg0) {
                // TODO Auto-generated method stub
                b.setBackgroundColor(Color.CYAN);
            }
        });
    }
}
```



אפשר להשתמש במחלקה אוניברטיית גם כדי לרשום מספר פקדים לאירוע מסוים.
לצורך כך נשמר את המופיע של המחלקה האוניברטיית בתוך משתנה.
לדוגמה:

```
public class MainActivity extends AppCompatActivity {
    int x=0;
    TextView T;
    ImageView Im;
    ציירת מופע של AMAZON לAIRTEL:
    View.OnClickListener Listener1=new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            switch (view.getId()) {
                case R.id.textView:
                    x=1;
                    break;
                case R.id.imageView:
                    x=2;
                    break;
            }
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        מציאת הפקדים לפי ה-id :
        T= (TextView) findViewById(R.id.textView);
        Im= (ImageView) findViewById(R.id.imageView);
        רישום הפקדים לAIRTEL:
        T.setOnClickListener(Listener1);
        Im.setOnClickListener(Listener1);
    }
}
```

2. הרשמה המחלקה הנוכחית במאזין לAIRTEL מסויים.

אנו נממש את המשך המתאים באמצעות המילה:

`implements` ולאחריה שם המქשייב לAIRTEL.

כעת דף ה-`activity` שלנו בולו בעצם מקשיב לAIRTEL, לדוגמה: לAIRTEL לחיצה על כפתור.

לבן בשנרצאה לרשום פקד מסוים, נכתב כך:

`b.setOnClickListener(this)`

`this` – הדף הנוכחי, שהוא כולל מקשיב לAIRTEL.

נចטרך להוסיף פונקציה המוכברת לממשק, שהיא בעצם הfonktsiya של AIRTEL.

הfonktsiya מקבלת משתנה מסוג `View`, שהוא הפוך עליו לחצתי.

```
public void onClick(View arg0)
{
}
```

בשיטה זו, יש לנו רק פונקציה אחת של AIRTEL.



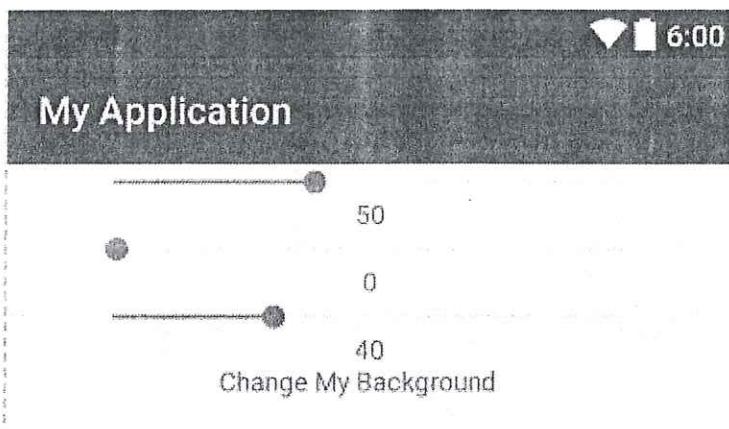
Android

בסיועך דשמי

לונגה:

```
public class MainActivity extends Activity implements onClickListener{  
    // מימוש המשך  
    int x=0;  
    TextView T;  
    ImageView Im;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        T= (TextView) findViewById(R.id.textView);  
        Im= (ImageView) findViewById(R.id.imageView);  
  
        רישום הפקדים לאיורע לחיצה:  
        T.setOnClickListener(this);  
        Im.setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View arg0) {  
        // TODO Auto-generated method stub  
        switch (view.getId())  
        {  
            case R.id.textView:  
                x=1;  
                break;  
            case R.id.imageView:  
                x=2;  
                break;  
        }  
    }  
}
```

תרגיל – SeekBars



הסבר כללי:

עליך ליצר אפליקציה שתכילה 3 seekBars שתפקידם לייצר צבע במבנה של RGB (red,green,blue) שיהווה צבע רקע למסך.

ליד כל seekBar יוצב TextView המוביל מספר. בהתחלה מספר זה יהיה 0. כל תזוזה של כל אחד מה-seekBars תקדם את המספר שלו, ובהתאם תשנה את הצבע של העמוד.

הוראות:

1. הוספי לכל SeekBar מאפיין android:progress (מגדיר כמה צעדים יתקדם הפס בכל פעם) ומאפיין של android:max (חשיبي מה המספר שהוא מקסימום התקדמות)
2. קשיי את ה-S seekBars לאירוע אחד שתצריב במשתנה. לדוג' :

```
SeekBar.OnSeekBarChangeListener L = new SeekBar.OnSeekBarChangeListener()
```

3. בדק מייהו ה-S seekBar שהוזע ע"ז:

```
switch (seekBar.getId())
{
case R.id.seekBar: ..... }
```

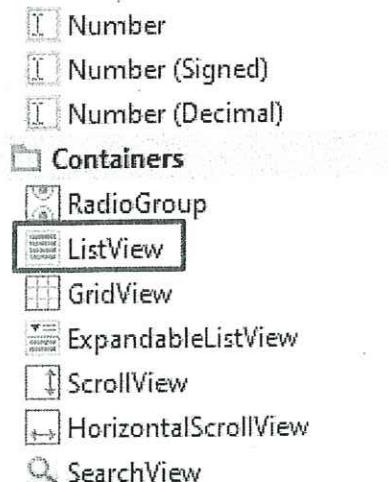
בהתאם נשנה את הטקסט שלו. (ערך התקדמות מתקובל כ-int באירוע
`(onProgressChanged`

4. מצאי את צורת הפרישה לתוך אובייקט מסונה. (לדוג' lin lin מסונה)
 בעת כל תזוזה של אחד מהפסים, צבעי את הרקע של העמוד. השתמשי בפונקציה
`lin.setBackgroundColor(Color.rgb(int,int,int));`
 חשבי מיהican תקבל את המספרים המרכיבים את הצבע.

5. הוספי בפטור שהרקעם שלו מוגדרים ע"י `<selector>` בפטור זה יAPS את כל ה-S seekBars לאפס. (קשיי לפונקציה בדרך רגילה במאפיין
`(onClick`

List View

היא רשימה המקבלת ערכים.
ניתן להוסיף אותה כמו בפקד מהתייבה הצד שמאל:



או ע"י יצירה תגית <ListView> בדף ה-.xml.

תכונות הפקד:

קוד הפקד.	
android:id	צבע או תמונה המשמשים להפרדה בין פריט לפרט ברשימה.
android:divider	גובה הקוו המפריד. לדוג' 15dp, 20px וכו'.
android:dividerHeight	צבע או תמונה המשמשים ברקע לפרט הנבחר ברשימה.
android:listSelector	פס גלילא מצד ימין לרשימה. True – אם נראה שיראה תמיד.
android:fastScrollAlwaysVisible	

ה-View מקבל ערכים ב-2 דרכים:

.Adapter 1.

Adapter זהו כל המאפשר לנו להעביר ערכים ממערך אל ה-View.

ביצד נשתמש ב-Adapter ?

א. ניצור מערך מסווג <T> ArrayList<T>.

ArrayList זהו אוסף גנרי. לצורך רשימה פשוטה, ניצור מערך מסווג String או int.
דוגמאות:

```
יצירת מערך //();  
ArrayList<String> namearray=new ArrayList<String>();  
הוספת ערכים למערך //();  
namearray.add("item1");  
namearray.add("item2");
```

ב. יצירה ArrayAdapter מאותו סוג של המערך.

דוגמאות:

```
ArrayAdapter<String> adapname=new ArrayAdapter<String>(this,  
        android.R.layout.simple_list_item_1, arrname);
```

- הסביר הדוגמא:
- **this** – דף ה-Activity הנוכחי.
 - **ListView** – הגדרת החזונה של ה-**ListView**.
איזה פונט/גודל/צבע וכו' יהיה לבן פריט ברשימה.
 - ניתן להגדיר גם דף **layout** אותו אנחנו יצרנו. בדוגמה מדובר בדף **layout** מובנה אוטומטית ל-**ListView**.
 - **arrname** – המערך אותו אנו רוצים להעיבר.

ג. נמצאת ה-**ListView** המבוקש ע"י **findViewById** ונגיד לו את ה-**Adapter**:

```
ListView v1=(ListView) findViewById(R.id.listView1);
v1.setAdapter(adapname);
```

2. ע"י הגדרת מערך קבוע **strings.xml** ושימוש בתוכונה **.strings** ניכנס לקובץ: **values->strings.xml**. נ עבור לשונית **strings.xml**
- א. ניצור **:StringArray**

```
<string-array name="arr1"> //  

    <item>item1</item> //  

    <item>item2</item>  

    <item>the_end_of_items</item>  

</string-array>
```

ב. נוסיף את התוכונה **android:entries** בתוכנות ה-**ListView** ונגיד את המערך הנ"ל

```
<ListView  

    android:id="@+id/listView1"  

    android:layout_width="match_parent"  

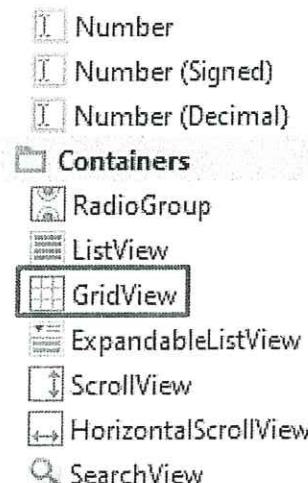
    android:layout_height="wrap_content"  

    android:entries="@array/arr1" //@array  

</ListView>
```

GridView

זוהי טבלה דו מימדית המתקבלת ערכים.
ניתן להוסיף אותה מהתיבה מצד שמאל בcn'ל:



או ע"י יצירה תגית <GridView> בדף ה-.xml.

תכונות הפקד:

android:id	קוד הפקד.
android:numColumns	מספר עמודות. יכול לקבל מספר קבוע, כמו "5" או "auto_fit" כלומר, מספר העמודות מותאם אוטומטית לנحوל הטבלה.
android:columnWidth	רוחב העמודה. יכול לקבל מידת מידה מסווג: px,dp,mm ועוד.
android:gravity	מיקום התוכן בכל תא. לדוגמה: center, bottom,top וכו'.

ה-GridView מקבל ערכים **רק ע"י Adapter**.
צורת השימוש ב-Adapter כמו שהסביר לעיל ב-ListView:
בוצע את שלב **א' וב'**, שלב **ג'** שוניה:

ג. נמצא את ה-GridView המבוקש ע"י `findViewById(R.id.gridView1)` ונגיד לו את ה-Adapter.

דוגמאות:

```
GridView gr=( GridView) findViewById(R.id.gridView1);  
gr.setAdapter(adapname);
```



קישור ListView/GridView לארוע לחיצה

```
ListView l=(ListView) findViewById(R.id.l);  
l.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
        המשנה א' מצין את מספר השורה/התא שנלחץ - כאשר האינדקס מתחילה ב-0//  
    }  
});
```

תרגיל**שלב א'**

צרי רשימה המכילה שפות שונות.
בעת לחיצה על כל אחת מהשפות, ישנה הרקע של הפריט הנבחר. (השתמשי במאפיין של הפקד בשם `listSelector`)

צרי תיבת טקסט, בפתור להוספה שפה חדשה, ובפתור להסרת שפה מהרשימה.
רקע הכתורים יהיה באמצעות `<selector>`.

הכנסי בתיבת הטקסט שפה כלשהיא:
בעת לחיצה על בפתור הוספה, היא תתווסף לרשימה.
בעת לחיצה על בפתור הסרה, היא תוסר מהרשימה.

השתמשי בפונקציות המובנות של ה-`adapter`:
`adapter.add("hebrew");` - להוספה שפה.
`adapter.remove("english");` - להסרת שפה הקיימת ברשימה.

בצעי בדיקת תקינות: בדק אם השפה קיימת במערך.

שלב ב'

צרי `TextView` המוביל מילה או משפט מסוים.
בעת לחיצה על כל אחת מהשפות "יתורגם" המשפט לשפה הנבחרת.