

תוכן

2	מבוא לאנדרואיד.....
2	מה זה אנדרואיד?.....
3	איך קבצי APK בנויים?.....
7	איזה אפליקציה נפתחת, מתי, למה ואיך?.....
8	מבנה הפרויקט ב android studio.....
13	Asset Packaging Tool – apk.....
14	מחזור החיים של האפליקציה - <i>Life Cycle</i>
14	תרגיל 1 – מחזור החיים.....
15	מבוא לעיצוב אפליקציה.....
15	View.....
16	Layout Managers.....
16	Linear Layout.....
17	תרגיל 2 – כתיבת בתצוגה בקוד java.....
18	תרגיל 3 – כתיבת התצוגה ב XML.....
20	תרגיל 4 – שימוש ב RelativeLayout.....
23	תרגיל 5 – שימוש ב TableLayout.....
26	תכונות.....
27	יצירת אובייקט מ- XML.....
28	אירועים:.....
29	הצגת רשומות במסך באמצעות Adapter (עיין ב- OSF בקובץ AdapterView).....
29	List View.....
29	Grid View.....
33	האופציה ההיברידית - רכיב WEBVIEW.....
34	דוגמא לשימוש ברכיב WebView.....

התרגיל הכיתי הזה מבוסס על אתר המפתחים בקישור: <https://developer.android.com/guide/topics/ui/declaring-layout.html>
חייבים גם לעיין עיין ב- OSF בקובץ "מבוא למערכת אנדרואיד", בקובץ "עיצוב ממשק גרפי באנדרואיד" ובקובץ "AdapterView"

מבוא לאנדרואיד

מה זה אנדרואיד?

אנדרואיד היא פלטפורמת תוכנה המביאה אליכם סט שלם של תוכנה עבור מכשירים ניידים, הכולל מערכת הפעלה, אפליקציות גישור ואפליקציות מפתח" (גוגל).

ראשית נבדיל בין המונחים. כולם מכירים את התרשים המפורסם שבו מפורטים השכבות השונות (הספריות) של המערכת, אך זו לא הדרך שבה היא עובדת (לא בדיוק בכל אופן) אלא הדרך שבה היא בנויה. אם נעלה ונסתכל על עבודת המערכת ממבט הציפור, נבין שהיא עובדת בצורה שונה ממערכות אחרות (מערכת ההפעלה הזאת לא מציאה את הגלגל, אך יש פה כמה הברקות)

כדי להבין כיצד היא עובדת יש צורך להבין כיצד קבצי APK עובדים

למה הכוונה? אנדרואיד כמערכת הינה חבילה של שורות קוד המריצות אפליקציות (או קבצי APK)

החייגן שלנו הוא אפליקציה (כלומר קובץ APK שבמקרה הזה, מצורף באופן מובנה לקוד המקור של אנדרואיד) כאפליקציה אנחנו יכולים גם להסיר אותו (כדי לא ליצור בלאגן גוגל הגדירה את האפליקציות המובנות כ- 'לא ניתנות למחיקה', אך אין זה אומר שהן באמת לא ניתנות למחיקה, עם ההרשאות המתאימות תוכלו לבצע הכל) כתוצאה מכך ניתן לבצע שינויים בכל דבר במערכת, לדוגמה: ניתן להסיר את החייגן המובנה ולהתקין במקומו אחד אחר.

אני לא אתווכח אם המושג אפליקציה הוא נכון או לא נכון, אך על מנת להבין עדיף לקרוא לאפליקציות קבצי APK, כי הם בפועל חבילות (Packages) והחבילות האלו יכולות להכיל 4 מרכיבים כדלהלן:

1. **Activity** אפליקציה מורכבת מהמון מסכים (Activities) שמשתמשים עוברים ביניהם (ולרוב כל מסך מבצע פעולה שונה) המסך הזה נקרא Activity (ברבים Activities) כותב האפליקציה בעצם כותב כמה וכמה Activity ועל ידי כפתורים שונים המשתמש מבצע מעבר ביניהם ובמעבר מאפליקציה לאפליקציה אתם בעצם עוברים מ- Activity ל- Activity **דוגמה להבנה**: נכנסתם לג'מייל נכנסתם ל- Activity בתוך ה- Package שנקרא Gmail משם עברתם לדואר הנכנס אז עברתם ל- Activity חדש בתוך ה- Package שנקרא Gmail משם נכנסתם לאיזשהו מייל אז עברתם לעוד Activity בתוך ה- Package של Gmail ובתוך המייל שכתבו לכם יש כתובת ולחצתם עליה ומשם עברתם ל- Activity בתוך ה- Package שנקרא maps וראיתם שבכתובת שבמפה יש מידע נוסף, לחצתם עליו והוא קישר אתכם לויקיפדיה ואז עברתם ל- Activity בתוך ה- Package שנקרא: chrome. ועוד הסבר להבנה, למשל, אם יש לנו אפליקציה שבה יש רשימה שמכילה שמות של תמונות, וכל בחירה (לחיצה) על אחד מרשימת השמות פותחת מסך אחר עם תמונה, אז הרשימה מוצגת ב- Activity והמסך של התמונה הוא מוצג גם כן Activity נפרד. Activity היא בעצם מחלקת אב לכל מסך שהמשתמש רואה. שימו לב, Activity אינה התוכן הגרפי, אלא רק **המסגרת** שבו הוא מוצג. התוכן הגרפי מוכל בתוך אובייקטים מסוג View. ה- Activity כלומר המסך מורכב (לרוב) מכפתורים, תיבות טקסט, תמונות וכד', את ה- Activity מעצבים באמצעות מסמך XML שמאפשר לנו לתאר כיצד המסך יראה והיכן יוצב כל רכיב. בעת מעבר מ- Activity אחד למשנהו (לדוג' בתוך אפליקציה-חבילה) ה- Activity הראשון נעצר וה- Activity השני מתחיל (אבל המערכת שומרת את ה- Activity הקודם במחסנית – FIFO כלומר האחרון שנכנס הוא הראשון שיוצא)

2. **Service** – ה- Service עובד ברקע, ולהבדיל מה- Activity אין לו ממשק משתמש (הוא מתואר על ידי שורות קוד בלבד ושקוף

למשתמש) ה- Service עוזר לנו לתת שירות לאפליקציה שלנו מאחורי הקלעים. **דוגמה להבנה:** הדוגמה הכי קלה היא חבילת (אפליקציית) המוזיקה, חבילת המוזיקה מורכבת מכמה Activity אנחנו עוברים בין השירים ובחרים שיר להשמעה, כנראה שלאחר מכן גם עוברים ל- Activity חדש שבו ניתן להריץ את השיר אחורה-קדימה, להפסיק, להפעיל וכד'. אבל מה קורה כשאנחנו יוצאים מה- Activity של המסך של המוזיקה, ועוברים לדוגמא ל- Activity של משלוח הודעה אך רוצים שהמוזיקה תמשיך לנגן ברקע ? לצורך הזה נועד ה- Service לתת מענה. ה- Service הינו רכיב הפועל **ברקע** ובא לתת לנו עזרה כדי לספק שירות כלשהו למשתמש, כמו למשל להמשיך לנגן את המוזיקה.

3. **Content provider** – ה- Content provider מנהל גישה למערך הנתונים של החבילה (האפליקציה) בין תהליך אחד לשני.

כאשר מפתח רוצה לגשת לנתונים של אפליקציה **אחרת** הוא משתמש באובייקט שנקרא: ContentResolver הוא שולח בקשות ל- Content provider שמנגד מקבל את הבקשה, מבצע את הפעולה ומחזיר את התוצאה. **דוגמה להבנה:** אנשי הקשר - אנו יכולים לפתח יישום שניגש ל- Content provider של אנשי הקשר על מנת לגשת לנתונים של החבילה (במקרה הזה הנתונים הם: רשימת אנשי הקשר שלנו) ולבקש ממנו לקרוא את הרשימה, לשנות אותה וכד'.

4. **Broadcast receiver** - הרכיב הזה מאפשר למפתח להגיב לאירועי מערכת שונים. מערכת האנדרואיד משחררת הכרזות שונות במהלך פעולתה, (ישנן גם הכרזות שרירותיות של אפליקציה מסוימת) לפעמים ההכרזות האלו מיועדות ל- Intent Filter ספציפי ולפעמים הן פשוט "נזרקות לאוויר". הרכיב **Broadcast receiver** הנ"ל מאפשר לנו "להקשיב" להכרזות הנזרקות במערכת ולבצע פעולות שונות בהתאם להכרזות. **דוגמה להבנה:** אני מניח שרובכם מכירים את האפליקציה שמקפיצה חלון Popup קטן ברגע שאנו מקבלים הודעת SMS האפליקציה הזאת משתמשת ב- Broadcast receiver היא מאזינה כל הזמן למערכת וברגע שהמערכת מודיעה שקיבלה הודעת SMS האפליקציה נכנסת לפעולה ומבצעת כמה פעולות על מנת להקפיץ לכם הודעה על המסך שתציג את תוכן ה- SMS.

איזה אפליקציה נפתחת, מתי, למה ואיך ?

כדי להבין קצת יותר לעומק איך אנדרואיד עובדת, אנחנו צריכים להבין למה אפליקציה אחת נפתחת ולא השניה וכיצד עובד המעבר בין Activity של חבילה א' לחבילה ב' ?

אז קודם כל, למושגים:

Intent ו- Intent Filter - כל חבילה (אפליקציה) גדולה יחסית תכיל לרוב Activity , Service , ו- Broadcast receiver אלה שלושה

מבין ארבעה מרכיבי הליבה של כל חבילה. שלושת אלה מופעלים באמצעות הודעות הנקראות: **intents** אובייקט מסוג Intent מייצג "כוונה" של המשתמש או של מערכת ההפעלה. קיימים 2 סוגים של Intents כדלהלן:

הסוג הראשון הוא Intent שמכוון באופן ברור ל- Activity או ל- Component כלומר, רכיב חומרה, **ספציפי** ונקרא: Explicit Intent

הסוג השני הוא Intent **שלא** מכוון באופן ברור ואפשר להגיד שהוא פשוט "נזרק לחלל האוויר". במקרה כזה המערכת (Intent Filter) תאתר את ה- Activity או את רכיב החומרה אליו הוא מכוון בעזרת המאפיינים של ה- Intent שהינם: Action, Category ו- Data. יש להדגיש, השימוש הנפוץ ביותר של Intent הינו מעבר בין Activities שהם מסכי UI בתוך החבילה.

להלן סיטואציה: במכשיר שלנו יש 2 חייגנים, החייגן המובנה, ו- Skype ואנו נמצאים ב- Activity של אנשי הקשר, ורוצים להתקשר לחבר. בלחיצה על איש הקשר ולאחר מכן על המספר שלו, עלה לנו חלון קטן שמבקש מאתנו לבחור דרך איזה אפליקציה אנו רוצים לחייג לאיש הקשר (דרך ה חייגן המובנה או דרך ה- Skype) למה זה קורה ?

כשאנו לוחצים על המספר של איש הקשר אנו שולחים Intent "לחלל האוויר" (ה- Intent לא מיועד באופן ספציפי לאף Activity או רכיב חומרה. ה- Intent Filter מאתר את ה- Activity או רכיב החומרה שמוכן לתת מענה לאותו Intent לפי המאפיינים שהזכרנו לעיל שהם: Action, Category ו- Data. במקרה שלנו: הפעולה (Action) הינה ביצוע חיוג / התקשרות. הקטגוריה (Category) הינה חייגנים. והמידע (Data) הינו המספר שאליו אנו נחייג.

ה- Intent Filter (שמופיע בקובץ AndroidManifest.xml של אפליקציה) מזהה 2 אפליקציות (חבילות) שעונות על המאפיינים האלו ויכולות להתמודד עם המשימה (החייגן המובנה ובנוסף ה- Skype והוא נמצא בבעיה, יש 2 אפשרויות, במי לבחור ? למערכת אין העדפה, ולכן קופצת לנו הודעת Popup קטנה שמבקשת מאתנו להתערב ולבחור באיזה מבין שתי האפשרויות לבחור ולחייג.

מבנה הפרויקט ב android studio

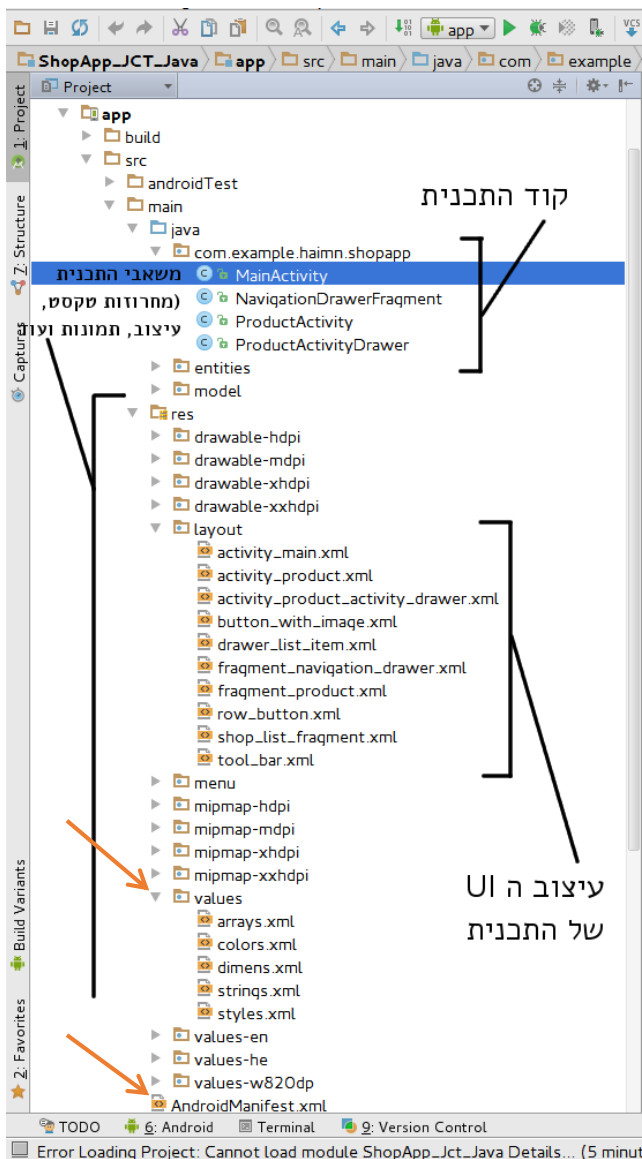
בכל פרויקט באנדרואיד יהיו לנו כמו מרכיבים בסיסיים:

1. `AndroidManifest.xml` המכיל רשימה של רכיבי התוכנית שלנו ותכונותיהם.
2. תיקיית `src` המכילה את כל קוד המקור של המחלקות שלנו.
3. תיקיית `res` שבה יופיעו כל שאר הרכיבים שלהם אנו זקוקים בכדי לייצר אפליקציה, כמו תמונות, קבצי `XML` שונים, קבצי אאודיו וכו'.

למשל:

- כל קבצי `.java` שמורים ב: `app → src → java → package(namespace)`
- כל קבצי `.xml` שמורים ב: `app → src → res → layout`
- כל התמונות שמורים ב: `app → src → res → drawable - ...`

לאחר פתיחת הפרויקט החדש, כחלק מהחלון הנפתח מופיע עץ התיקיות:



נפרט בקצרה את התיקיות בהם נשתמש במהלך הפרויקט, ותפקידן:

Java-

מכילה את קטעי קוד ה- `java`, כברירת מחדל קיבלנו `com.example.haimn.shopapp` package-
שמכיל את קטעי הקוד המטפל בחלק הלוגי של חלק העיצוב לדוגמא: מה יתחולל כאשר ילחץ כפתור. ה-
packages האחרים כמו `model` ו-`entities` הוספנו לחלק ה-`backend` של הפרויקט.

Res/layout —

לכל אובייקט עיצובי ממש: חלונות, כפתורים, text view למיניהם, layouts וההגדרות המיוחדות

תיקייה זו מתכתבת ישירות עם התיקייה הקודמת כפי שיילמד.

Res/drawable –

בתיקייה זו יוכנסו תמונות, קטעי xml עיצוביים וכד' שבאמצעותם יעוצבו הפריטים שבתיקייה הקודמת.

במוקצית תיקייה לכל גודל בפיקסלים של אובייקט שכזה, על מנת שעבור כל מכשיר יהיה זמין עיצוב ברזולוציה התואמת את אותו המכשיר.

Res/menu –

מכילה קבצי xml שבאמצעותם ניתן לעצב את ה-Action Bar שהוא הפס העליון המופיע בראש כל Activity. בעזרת הקבצים בתיקיה זו ניתן להוסיף כפתורי שליטה ל- bar זה כגון: כפתור חזרה לעמוד קודם, חיפוש, settings וכד'.

AndroidManifest – קובץ xml שמכיל **מידע** הגדרות כללי ובסיסי על האפליקציה (עבור האנדרואיד). ניתן באמצעותו לקבוע מי מבין ה- Activities תופעל ראשונה באפליקציה כלומר ה- Top level activity, כל זה יוגדר בתוך ה- Intent Filter בתוך ה- **Manifest** וגם שם ייקבע מי יפעיל (מבחור) את ה- Activity שלנו. ניתן לשנות מתוך ה- **Manifest** למשל את צבע ה- ActionBar עבור **האפליקציה**, לשנות את שמות עמודי האפליקציה וכד'. כלל ניתן לומר כי באמצעות מסמך זה ניתן לבצע שינויים עיצוביים **כוללים** בכל האפליקציה, עבור שינויים מקומיים/מינוריים יערכו שינויים בקבצי xml בעיקר בחתיכה `res/layout`.

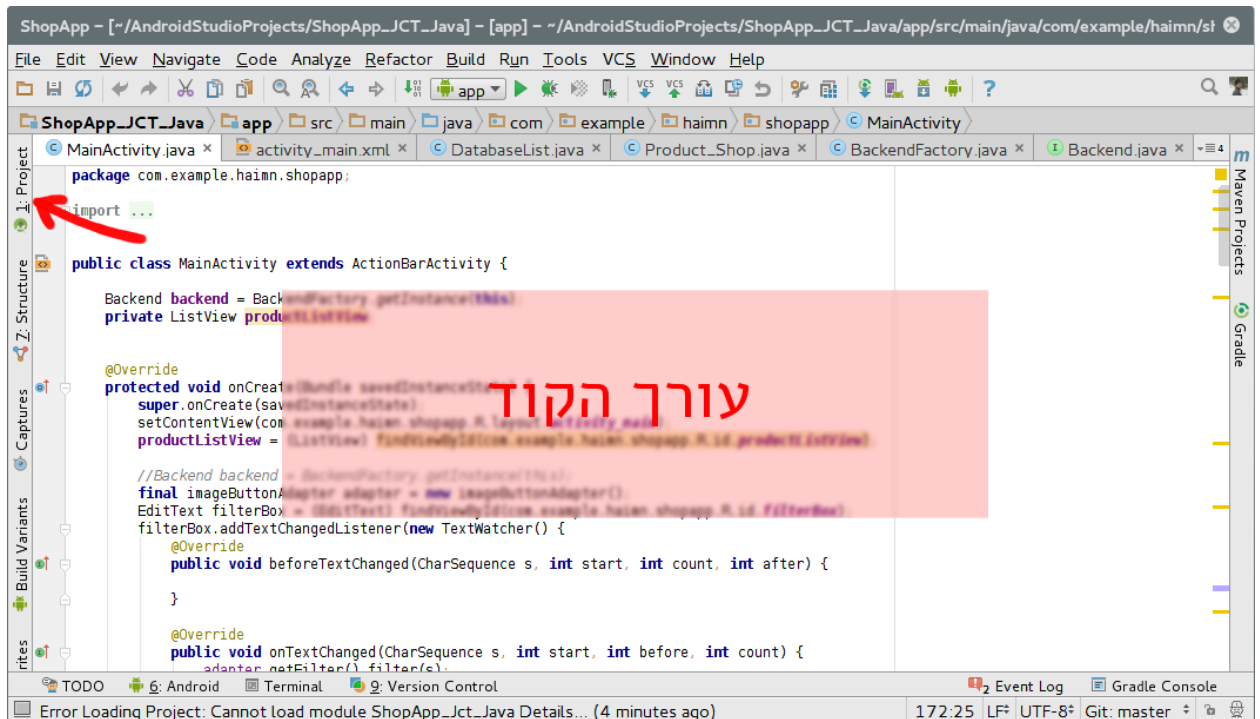
כל הזכויות שמורות למחלקת המחשבים של המרכז האקדמי לב אין להעתיק ולהעביר בדרך כלשהי ללא רשות

הדוגמא הבאה מכילה חלק מקובץ ה- AndroidManifest.xml ה-Activity בשם: **LoginWindow** שעבורה מוגדר הפילטר תהיה הראשונה שיעולה.

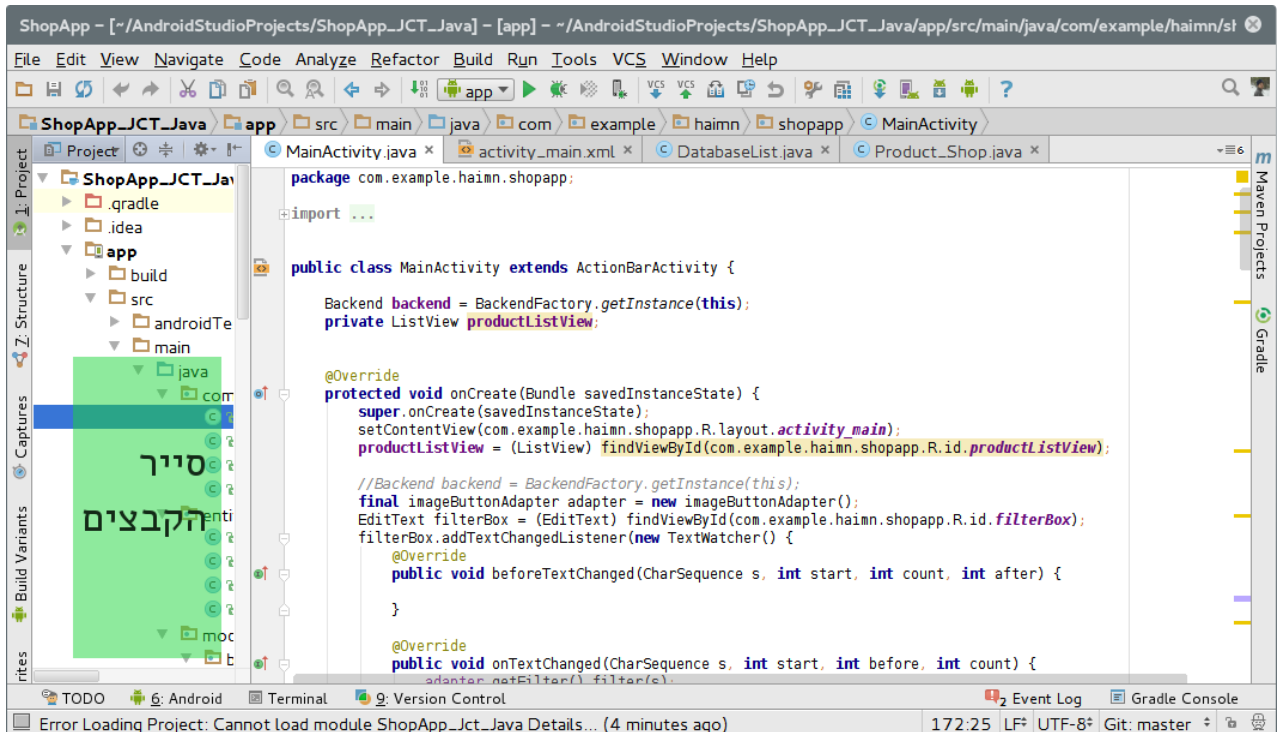
```
<activity android:name=".LoginWindow" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

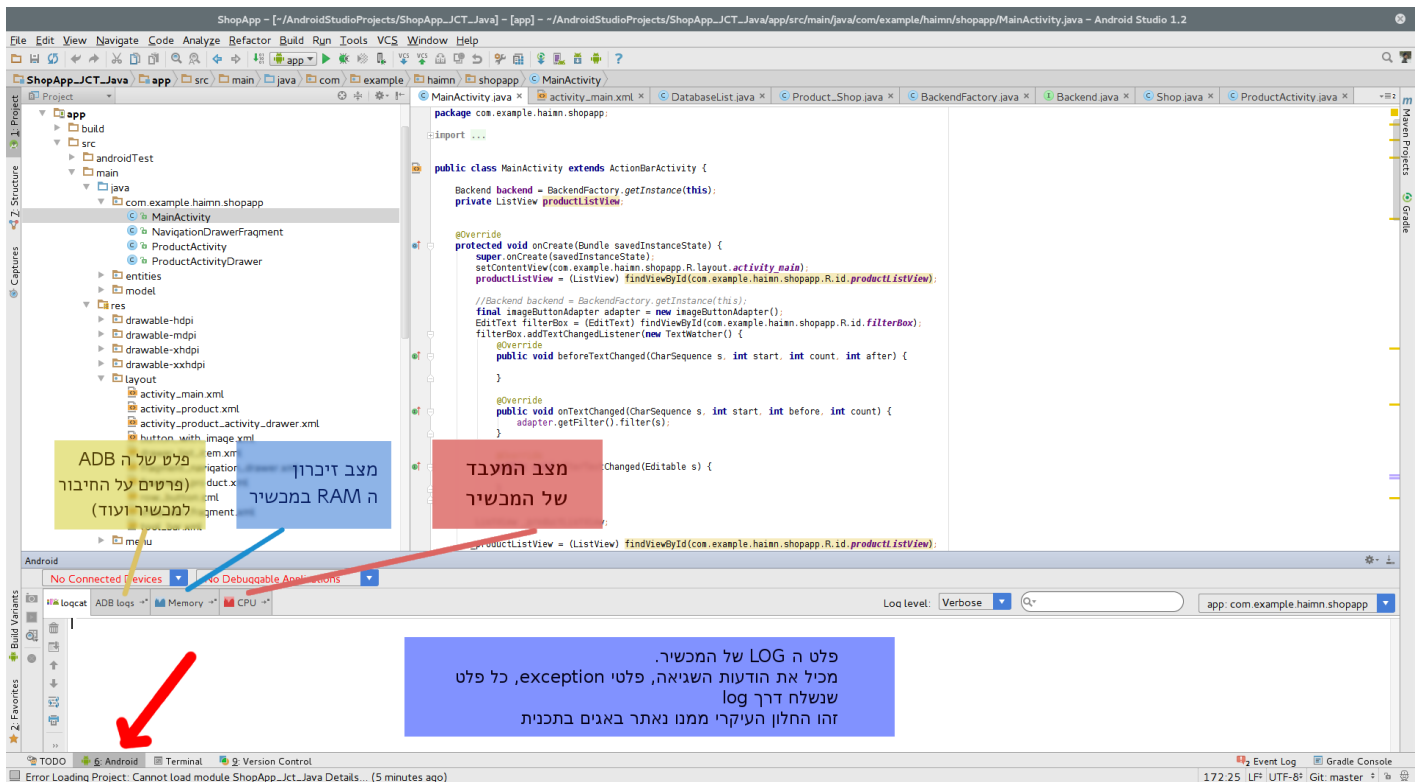
נעבור בקצרה על ממשק המשתמש של Android Studio, כאשר נפתח תוכנה נראה את עורך הקוד שלנו שם בעצם נבצע את מרבית העבודה - כתיבת הקוד.



כאשר נלחץ על הכפתור בקצה השמאלי של המסך כמתואר בתמונה למעלה (על מה שהחץ האדום מצביע) ייפתח לנו סייר הקבצים בו נוכל לראות את כל קבצי הפרויקט שלנו.

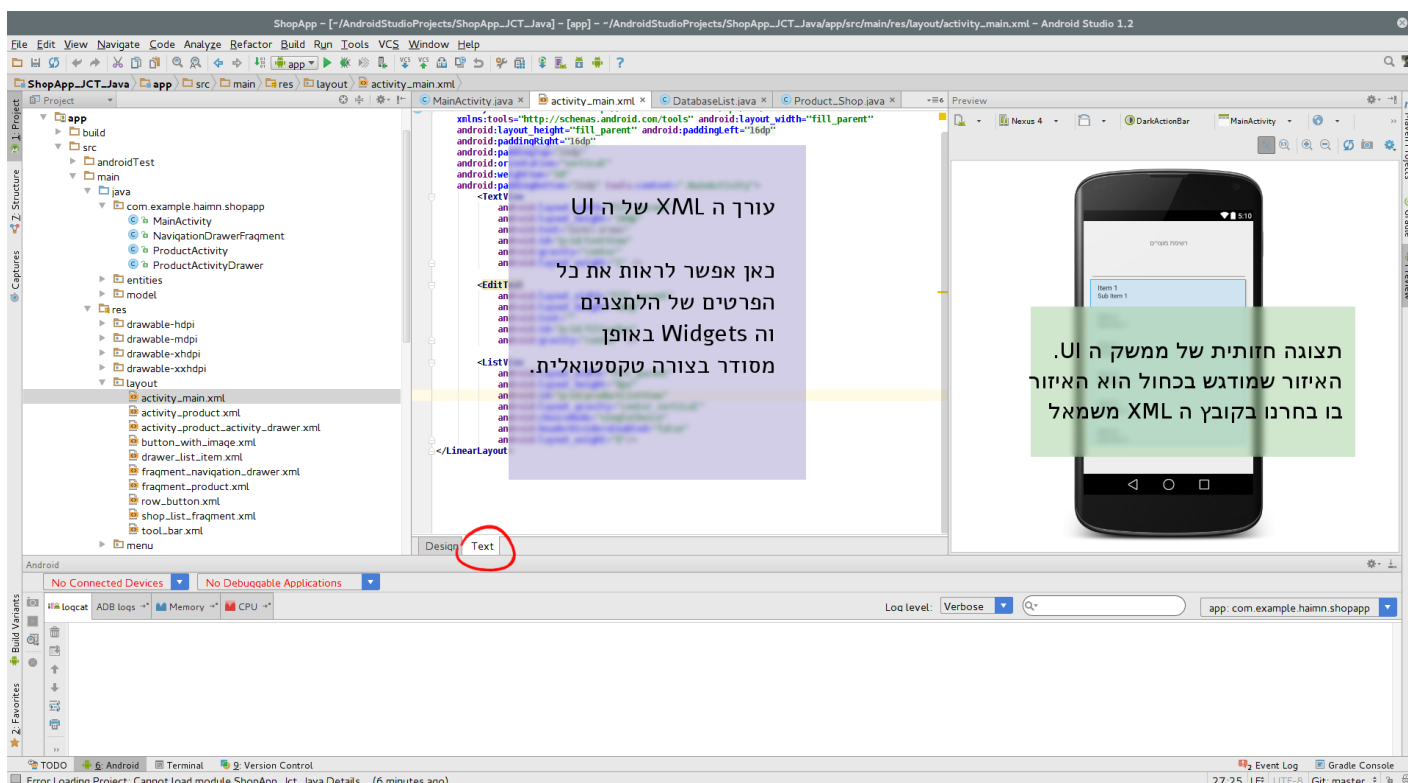


כל הזכויות שמורות למחלקת המחשבים של המרכז האקדמי לב אין להעתיק ולהעביר בדרך כלשהי ללא רשות



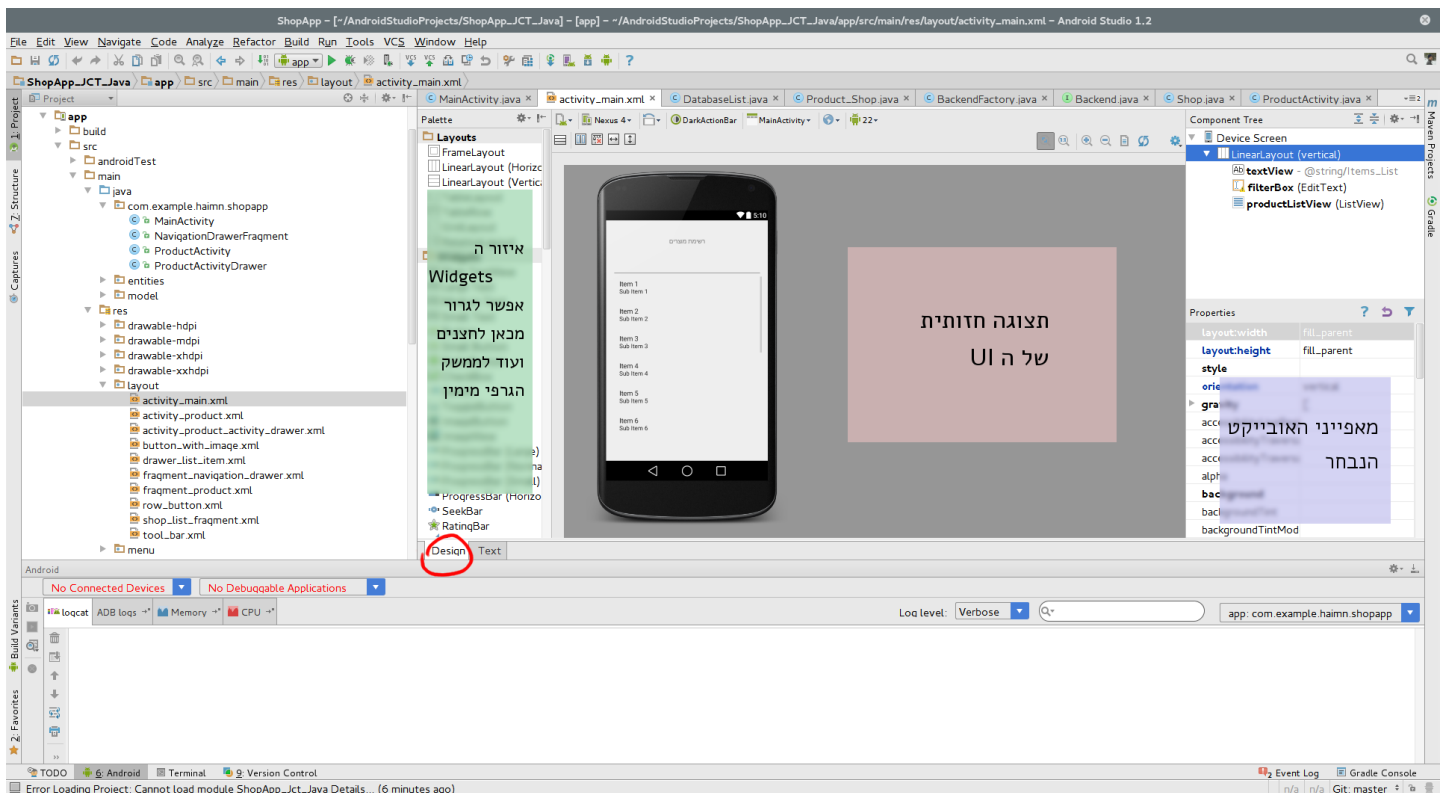
אם נלחץ על ה-TAB המסומן בחץ אדום בתמונה למעלה ייפתח לנו חלון ה-debugging של התוכנה, בחלון זה נוכל לראות את הודעות השגיאה של התוכנית וכך לדעת מה גורם לתקלה, כמו כן נוכל לראות את מצב משאבי המערכת של המכשיר כדי שנוכל לזהות התנהגות חשודה כמו דליפת זיכרון למשל או צריכת מעבד מוגברת.

כאשר תתרחש שגיאה קריטית באפליקציה שתגרום לקריסה מלאה שלה, נוכל לראות הודעת שגיאה מודגשת בצבע אדום בחלון ה-debugging, בתוך הודעה זו נקבל רשימה של קבצים הקשורים לשיאה זו, כדי שנוכל לדעת איזו שורה בקוד שלנו גרמה לשיאה נחפש בתוך הודעת השגיאה שורה המודגשת בצבע כחול, כאשר נלחץ על שורה זו נעבור ישירות לקטע הקוד הבעייתי.



כאשר נפתח קובץ xml שמכיל את העיצוב של האפליקציה שלנו ייפתח לנו החלון למעלה, חלון זה מחולק לשני חלקים, קוד ה- xml ותצוגה הגרפית שלו, כאשר נלחץ על קטע קוד כלשהו בקובץ ה- xml יודגש הרכיב המתאים שלו בממשק הגרפי.

נוכל לעבור לתצוגה גרפית מלאה על ידי לחיצה על הכפתור Design המוקף בעיגול בתמונה למטה.



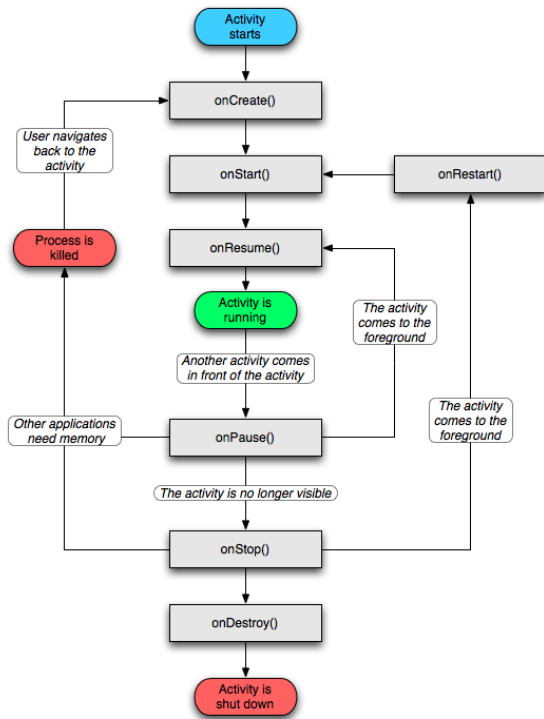
בחלון זה נוכל לגרור רכיבים בקלות מתוך רשימת הרכיבים אל תוך מסך האפליקציה שלנו.

Asset Packaging Tool – apk

כשנרצה לייצא פרויקט באנדרואיד, נשתמש ב- APK. ה- APK אורז לנו את כל הפרויקט שלנו בקובץ בינארי (לא טקסטואלי) אחד, ומכונן אותו. מערכת ההפעלה יודעת להתקין ולהפעיל קבצי APK.

כל הזכויות שמורות למחלקת המחשבים של המרכז האקדמי לב אין להעתיק ולהעביר בדרך כלשהי ללא רשות

מחזור החיים של האפליקציה - Life Cycle



כאשר אנו מריצים אפליקציה, Android הולכת קודם כל לקובץ AndroidManifest.xml כדי לדעת איזה Activity להריץ, יוצרת אובייקט Activity כזה ומריצה את השיטה onCreate() אחרי זה את השיטה onStart() ואת onResume().

התמונה על יד מעלה את כל הפעולות שעושה המערכת הפעלה לפי פעולת המשתמש.

עוד כמה שיטות:

onPause – שיטה הנקראת כאשר הפעילות מוקפאת, למשל כאשר עוברים לפעילות אחרת ע"י Intent (דוגמא באתר).

onResume – שיטה הנקראת כאשר הפעילות חוזרת לפוקוס, כלומר חוזרת מהקפאה.

עוד פרטים על נושא זה תמצאו כאן:

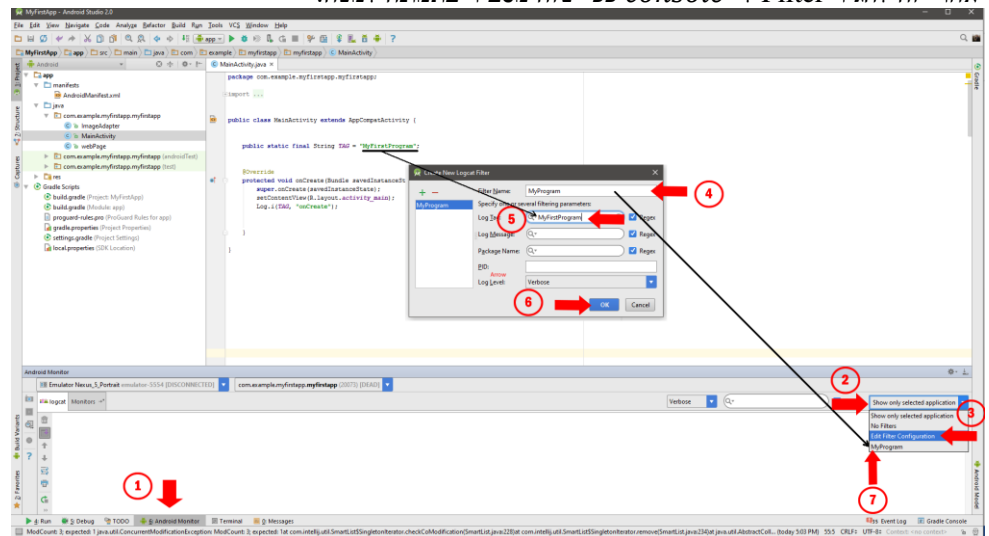
<http://developer.android.com/reference/android/app/Activity.html>

בשיטה כמו"כ ניתן לקרוא בשפה העברית מתוך [הקובץ הבא](#):

בשיטה onCreate אנו מגדירים את פעולות ה Activity שלנו, לדוגמה שמירת האובייקטים במופעה שלנו.

לצורך העניין, אנו הולכים לבנות תוכנה קטנה שתכתוב ב-console בכל פעם שתופעל אחת השיטות המופיעות בתרשים.

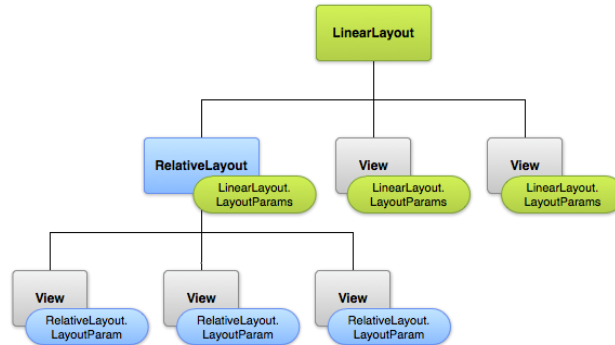
בשביל זה אנו נגדיר מופע בשם: TAG מסוג final string ולתת הערך של "MyFirstProgram" אחרי זה להגיר Filter ל-console כפי שזה מסביר בתמונה למטה:



נממש את השיטה onCreate כפי שמופיע לעיל, נפעיל את האפליקציה ונקבל פלט "onCtreate" נבצע את זה עבור כל שיטה;

תרגיל 1 – מחזור החיים

בצע את [התרגיל הבא](#) במלואו



View היא אב הטיפוס של כל רכיבי ממשק המשתמש באנדרואיד

כל View מתפרס על שטח מרובע כלשהו, והוא אחראי על ציור הרכיב וטיפול באירועים שלו

ה-View הוא בעצם מעין Canvas כמו בלוק ציור, שעליו אנו יכולים "לצייר" ככל העולה על רוחינו. בד"כ בכל מסך בתוכנית יש הרבה אובייקטים מסוג View.

למשל, אם יש לנו מסך עם שתי תיבות טקסט, אז בפועל יש לנו שלושה אובייקטים של View – כל תיבת טקסט היא View בעצמה, וכן הרקע שעליו תיבות הטקסט מצוירות גם הוא View.

View Groups הם מעין "מכלים" שקופים המכילים בתוכם רכיבים גרפיים אחרים, ומגדירים את סידורם היחסי במסך.

על סוגי ה-View Groups נעסוק בהמשך ב-Layout Managers.

Layout Managers

בפלטפורמת אנדרואיד ישנו אוסף של אובייקטים View שמשמשים כ-"מכלים" לרכיבים אחרים. המכלים האלו מכונים: Layout Managers.

כל Layout Manager מנהל את הרכיבים שהוא מכיל באופן אחר, הן מבחינת מיקום והן מבחינת גודל.

Linear Layout



כמו ה- *StackPanel* ב-WPF, המכל הזה מארגן את הרכיבים שהוא מכיל אחד אחרי השני, בצורה אופקית או אנכית, בהתאם לערך התכונה "orientation" שלו.

בשלב הראשון ניצור את התצוגה שלא באמצעות XML אלא באמצעות קוד. נשתמש ב- Linear Layout

תרגיל 2 – כתיבת בתצוגה בקוד java

עליך לבצע את התרגיל.

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState)
3  {
4      super.onCreate(savedInstanceState);
5
6      Button button1 = new Button(this);
7      button1.setText("button1");
8
9      TextView textView1 = new TextView(this);
10     textView1.setText("text view 1");
11
12     Button button2 = new Button(this);
13     button2.setText("button2");
14
15     LinearLayout layout = new LinearLayout(this);
16     layout.setOrientation(LinearLayout.VERTICAL);
17
18     layout.addView(button1);
19     layout.addView(textView1);
20     layout.addView(button2);
21
22     setContentView(layout);
```

הסבר:

בשורות 6-12 יוצרים רכיבי תצוגה.

בשורה 14-15 מוגדר layout מסוג ליניארי לאורך.

בשורות 17-19 מוסיפים את הרכיבים שהגדרנו (ה- View השונים) לתוך ה layout

בשורה 21 קובעים שה layout הנ"ל יהיה התצוגה של ה- Activity

הפעם נבצע את התרגיל הנ"ל בהפרדה של התצוגה מהקוד להלן דוגמה של קוד תצוגה ב- XML:

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

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="New Button" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="New Button" />



    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="New Button" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="New Button" />

</LinearLayout>
```

- א. צור empty project
- ב. גש לקובץ activity_main.xml ב-res->layout->
- ג. החלף את תוכן הקובץ activity_main.xml בקובץ ה- xml הנ"ל
- ד. הרץ את התוכנית

הרץ את התוכנית בערך **vertical** וגם בערך **horizontal** עבור התכונה orientation ותקבל:

orientation="horizontal"	orientation="vertical"
	

כדאי לשים לב, מספיק מבט פשוט על הקוד כך שניתן לראות שכל הרכיבים מוכלים ב- `LinearLayout` כמו כן ניתן לראות כל תכונה של כל רכיב ורכיב ואת סדר הופעתם.

שימו לב שבמקרה של הגדרת רכיב ב- `xml` יש חובה להגדיר עבורו את האורך והרוחב באמצעות:

`android:layout_width`

`android:layout_height`

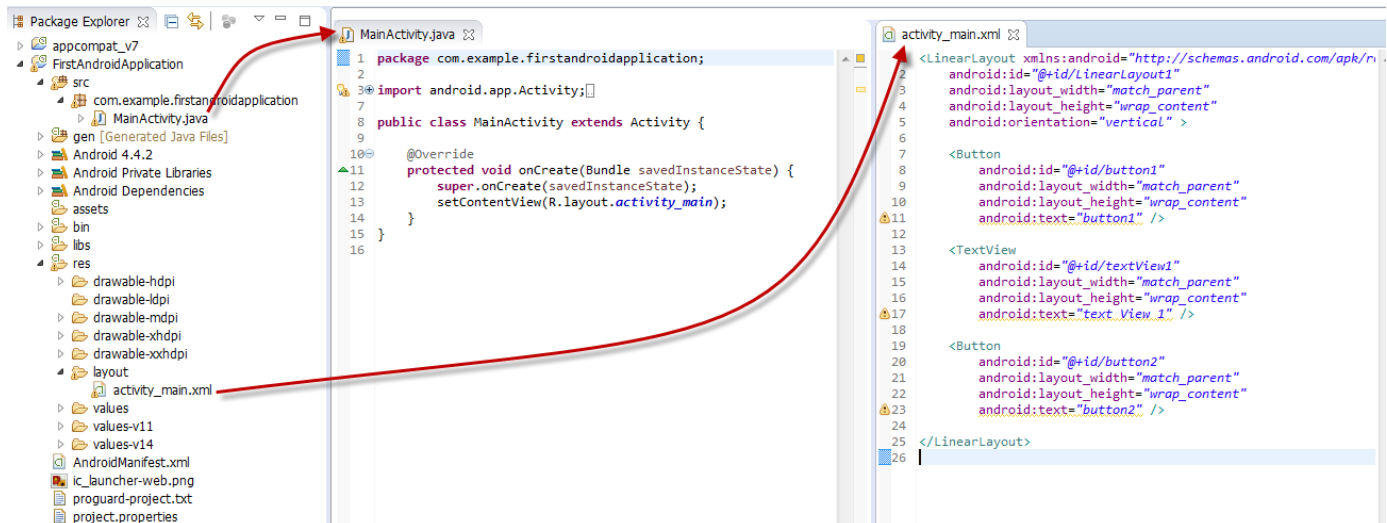
ניתן לציין ערך מספרי (עם `px` או `dp` וכו..) וניתן להשתמש גם בהגדרה קבועה כפי שנעשה בקוד הנ"ל:

- `match_parent` (או `fill_parent`) מציין שהרכיב ימלא את השטח על מנת להיות באותו גודל של הרכיב שהוא מוכל בו
- `wrap_content` מציין שהרכיב ימלא את השטח לפי מה שהוא צריך.

על מנת שה- `Activity` שלנו יציג את התצוגה שהגדרנו בקובץ ה- `xml` נגדיר בפונקציה `onCreate` מי ה- `contentView` שלו:

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState)
3  {
4      super.onCreate(savedInstanceState);
5
6      setContentView(R.layout.activity_main);
7  }
```

כך ש- `activity_main` זה שם קובץ ה- `xml` שמגדיר את התצוגה והוא צריך להיות בתוך הפרויקט בתיקייה: `res\layout`





המכל הזה מאפשר לנו לציין את המיקום של רכיב אחד ביחס לרכיב אחר לדוגמה: רכיב A משמאל לרכיב B או רכיב C למעלה/למטה מרכיב A.

היחסים מוגדרים בעזרת השדות:

- layout_alignParentTop - קובע אם הרכיב הנתון יהיה בראש המיכל.
- layout_alignTop - קובע אם הרכיב הנתון יהיה באותו גובה של מיכל אחר.
- layout_below - קובע מתחת לאיזה רכיב ימצא הרכיב הנוכחי.
- layout_toLeftOf - קובע משמאל לאיזה רכיב ימצא הרכיב הנוכחי.
- layout_toRightOf - קובע מימין לאיזה רכיב ימצא הרכיב הנוכחי.

תרגיל 4 – שימוש ב RelativeLayout

שנה את קוד ה- XML של התצוגה הקודמת לקוד הבא

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

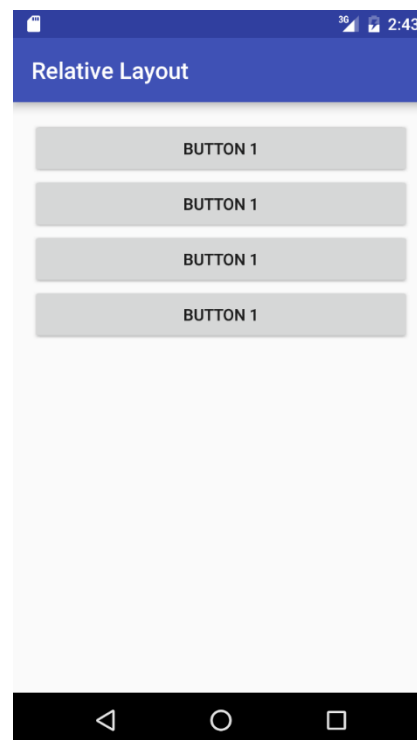
    <Button
        android:id="@+id/b1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button 1" />

    <Button
        android:id="@+id/b2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/b1"
        android:text="Button 1" />

    <Button
        android:id="@+id/b3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/b2"
        android:text="Button 1" />

    <Button
        android:id="@+id/b4"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/b3"
        android:text="Button 1" />

</RelativeLayout>
```



RelativeLayout → Example 1

דוגמה של קוד כאשר רוצים לסדר את הרכיבים ב- **RelativeLayout** בצורה אופקית
שנה את קוד ה- XML של התצוגה הקודמת לקוד הבא

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

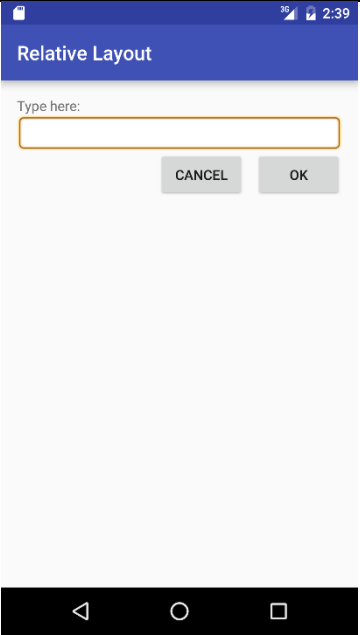
    <Button
        android:id="@+id/b1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1" />

    <Button
        android:id="@+id/b2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/b1"
        android:layout_toRightOf="@id/b1"
        android:text="Button 2" />

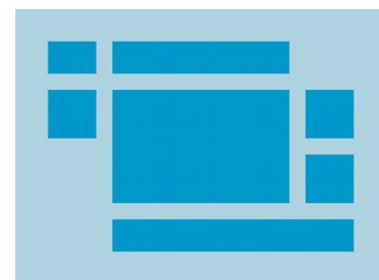
    <Button
        android:id="@+id/b3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/b2"
        android:layout_toRightOf="@id/b2"
        android:text="Button 3" />
</RelativeLayout>
```



RelativeLayout → Example 2

<pre> <RelativeLayout android:layout_width="match_parent" android:layout_height="match_parent"> <TextView android:id="@+id/label" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Type here:" /> <EditText android:id="@+id/entry" android:layout_width="fill_parent" android:layout_height="wrap_content" android:layout_below="@id/label" android:background="@android:drawable/editbox_background" /> <Button android:id="@+id/ok" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignParentRight="true" android:layout_below="@id/entry" android:layout_marginLeft="10dip" android:text="OK" /> <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_alignTop="@id/ok" android:layout_toLeftOf="@id/ok" android:text="Cancel" /> </RelativeLayout> </pre>	
---	---

RelativeLayout → Example 3



TableLayout הוא ViewGroup המציג אלמנטים בשורות ועמודות.

TableLayout - ממקם הילדים שלו לתוך שורות ועמודות. מכילות TableLayout אינם מציגים קווי הגבול עבור שורות, עמודות, או לתאיהם. TableLayout יכול להשאיר תאים ריקים, אבל תאים לא יכולים להקיף עמודות (columnspan), כפי שהם יכולים ב WPF.

TableRow - כל TableRow מגדיר שורה אחת בטבלה. לכל שורה קיימים אפס או יותר תאים, ולכל תא יכול להיות View או ViewGroup (למשל: LinearLayout).

תרגיל 5 – שימוש ב TableLayout

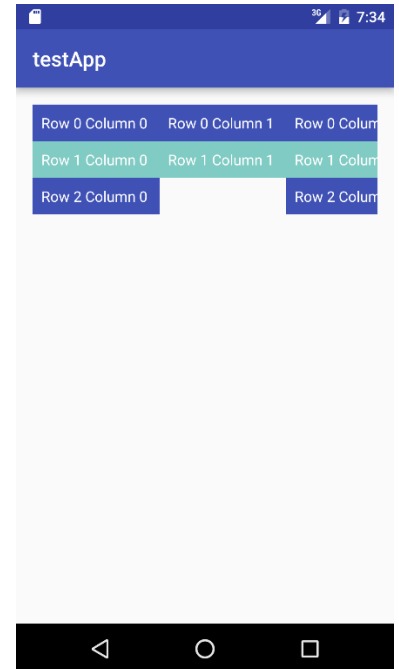
בצע את הדוגמא הבאה,

אל תשכח לתת ערכים להפניות ב- strings.xml שנמצא ב- Res-Values

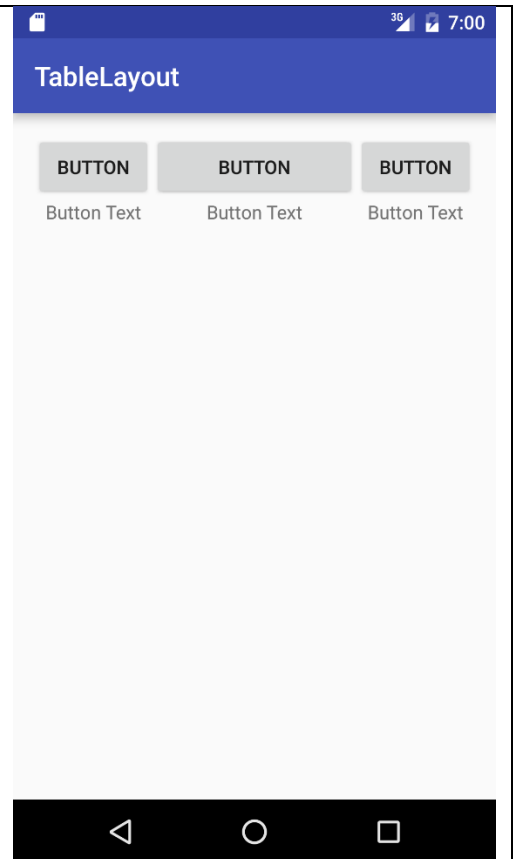
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_open"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_open_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="@string/table_layout_4_save"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_save_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="Row 0 Column 0 "
            android:background="#FF3F51B5"
            android:textColor="#ffff"
            android:padding="8dp"
        />
        <TextView
            android:text="Row 0 Column 1 "
            android:background="#FF3F51B5"
            android:textColor="#ffff"
            android:padding="8dp" />
        <TextView
            android:text="Row 0 Column 2 "
            android:background="#FF3F51B5"
            android:textColor="#ffff"
            android:padding="8dp" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Row 1 Column 0 "
            android:background="#80cbc4"
            android:textColor="#ffff"
            android:padding="8dp" />
        <TextView
            android:text="Row 1 Column 1 "
            android:background="#80cbc4"
            android:textColor="#ffff"
            android:padding="8dp"
        />
        <TextView
            android:text="Row 1 Column 2 "
            android:background="#80cbc4"
            android:textColor="#ffff"
            android:padding="8dp"
        />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_column="0"
            android:text="Row 0 Column 0 "
            android:background="#FF3F51B5"
            android:textColor="#ffff"
            android:padding="8dp" />
        <TextView
            android:text="Row 2 Column 2 "
            android:layout_column="2"
            android:background="#FF3F51B5"
            android:textColor="#ffff"
            android:padding="8dp"
        />
    </TableRow>
</TableLayout>
```




```
<TableRow>
  <LinearLayout
    android:orientation="vertical"
    android:text="Row 0 Column 0 ">
    <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button"/>
    <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button Text"
      android:textAlignment="center" />
    </LinearLayout>
  <LinearLayout
    android:orientation="vertical" >
    <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button"/>
    <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button Text"
      android:textAlignment="center" />
    </LinearLayout>
  <LinearLayout
    android:orientation="vertical"
    android:text="Row 0 Column 0 ">
    <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button"/>
    <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Button Text"
      android:textAlignment="center" />
    </LinearLayout>
</TableRow>
```



כל View תומך במגוון של תכונות משלו. חלק מהתכונות הן ספציפיות (למשל, TextView תומך בתכונת textSize) וחלק מהתכונות הן כלליות כי הן נורשות מהמחלקה-View. כמו כן, קיימות תכונות אחרות מסוג "layout parameters", שהן תכונות המתארות אוריינטציות פריסה מסוימות של האובייקט View, כפי שהוגדר על ידי אובייקט האב ViewGroup.

ID

כל משאב שאנחנו יוצרים יקבל מספר זיהוי ייחודי ע"מ לאפשר זיהוי ייחודי בתוך עץ. כאשר ה-Parser (מנתח) עובר על קובץ ה-XML, הוא יוצר מהערך שהוצמד ל-ID שלם וכך האובייקט הזה ירשם בקובץ R.java.

הסמל (@) בתחילת המחרוזת מציין שה-XML Parser צריך לנתח ולהרחיב את שאר המחרוזת המזהה ולזהותו כמשאב מזהה.

לדוגמא:

```
android:id="@+id/my_button"
```

הסמל (+) מציין כי מדובר שם משאב חדש כי יש ליצור והוסיף למשאבים שלנו (בקובץ R.java). קיים מספר משאבי אחרים המוצעים במסגרת אנדרואיד. כשמעונינים לגשת למשאב קיים אין צורך להשתמש בסימן+:

לדוגמא:

```
findViewById(R.id.my_button);
```

כדי ליצור אובייקט מתצוגה ולפנות אליו מתוך הקוד יש:

1. להגדיר view/widget בקובץ XML ולהגדיר לו ID לדוגמה:

```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

2. לאחר מכן ליצור מופע של האובייקט. בד"כ זה מתבצע בתוך השיטה onCreate () כדלהלן:

```
Button myButton = (Button) findViewById(R.id.my_button);
```

הערה: ה-ID לא צריך להיות ייחודי לאורך כל העץ כולו, אבל זה צריך להיות ייחודי בתוך החלק של העץ שאנחנו משתמשים.

כדי להגיב על אירוע שמתקבל מ- views (לדוגמא כפתור) עלינו לממש את השיטה onItemClick שבממשק OnClickListener לדוגמא:

```
// Create a message handling object as an anonymous class.
private OnClickListener mMMessageClickedHandler = new OnClickListener()
{
    public void onItemClick(AdapterView parent, View v, int position, long id) {
        Log.i(TAG,"Event");
        // Do something in response to the click
    }
};
```

ואחר כך להפעיל את setOnClickListener() של ה- view לדוגמא
(ובהנחה שכבר קיים כפתור אם התכוונה android:id="@+id/my_button"):

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

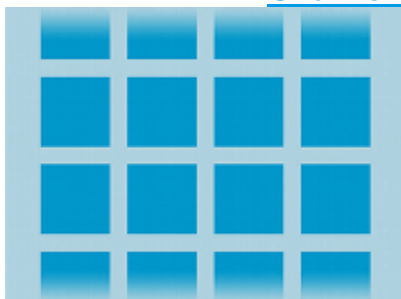
    Button myButton = (Button) findViewById(R.id.my_button);
    myButton.setOnClickListener(mMMessageClickedHandler);
}
```

הצגת רשומות במסך באמצעות Adapter (עייין ב- OSF בקובץ AdapterView)

שימו לב:

כאשר נבצע Copy&Paste לקוד הבא אזי חלק מהמחלקות לא יהיו מוכרות בגלל חוסר ב- import במקרה כזה נציב את הסמן על המחלקה הלא מוכרת ונלחץ ביחד Alt Enter.

Grid View



List View



ההבדל בין שני המתאמים האלה הוא שה- ListView מסדר אחד מתחת לשני ומאידך ה- GridView מסדר אחד ליד השני. נבצע דוגמא דוגמה קטנה של GridView ובאמצעותה נבין כיצד להגדיר Adapter ל- AdapterView.

נתחיל ב- Activity_Main.xml וניצור תצוגת GridView כדלהלן:

```
<RelativeLayout 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.testapp.testapp.MainActivity">

    <GridView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/mygridview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:columnWidth="90dp"
        android:horizontalSpacing="10dp"
        android:verticalSpacing="10dp"
        android:numColumns="auto_fit"
        android:padding="5dp"

    />
</RelativeLayout>
```

הגדרנו לרכיב ID בשם: "mygridview".

כמו"כ הגדרנו שכל "קוביה" תהיה ברוחב של 90dp והגדרנו שבין כל "קוביה" יהיה רווח אופקי ורווח אנכי שווה של 10dp

בשלב הבא ניגש לתיקיה "res → layout" ונבנה layout חדש בשם "item.xml" שמכיל layout ראשי מסוג Linear Layout עם התכונה
orientation = "vertical"

הקוד:

Item.xml

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

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/myButton"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Text"
        android:textAlignment="center"
        android:id="@+id/myText"/>
</LinearLayout>
```

בתוך item יהיה כפתור
שיתפוס כל המקום של המכל בו הוא הוגדר (ע"י הגדרת תכונה android:layout_width = "match_parent")
וכמו"כ גודלו יוגדר לפי התוכן שהוא מכיל (ע"י הגדרת תכונה android:layout_height = "wrap_content")

וכן יהיה TextView עם אותן תכונות של הכפתור כנ"ל.

לאחר מכן נבנה מחלקה חדשה של Java בשם: *ItemsAdapter* שיורשת מ- *BaseAdapter*.

במחלקה יהיו 2 אובייקטים ובנאי שמקבל אותם.
האובייקטים הם:

- *Context* בשם: *context*
- רשימה שכל אלמנט שלה הוא מסוג *String* בשם *content* שתכיל את התוכן (Text) של הכפתורים שנציג בהמשך.

ItemsAdapter.java

```

public class ItemsAdapter extends BaseAdapter{

    List<String> content;
    Context context;

    public ItemsAdapter(Context c, List<String> content) {
        context = c;
        this.content = content;
    }

    @Override
    public int getCount() {
        return content.size();
    }

    @Override
    public Object getItem(int position) {
        return content.get(position);
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View gridView;

        if (convertView == null) {

            gridView = new View(context);

            // get layout from resources
            gridView = inflater.inflate(R.layout.item, null);

            // set image based on selected text
            Button btn = (Button) gridView.findViewById(R.id.myButton);
            btn.setText(content.get(position));

            TextView textview = (TextView) gridView.findViewById(R.id.myText);
            textview.setText("Button " + position);

        } else {
            gridView = (View) convertView;
        }

        return gridView;
    }
}

```

כאשר יורשים מהמחלקה *BaseAdapter* חייבים לממש הפונקציות הבאות.

- `getCount()` - מחזיר הסה"כ של אובייקטים שקיימים ב- *Adapter* (במקרה שלנו הגודל של *content*)
- `getItem(int position)` - מקבל את המיקום ומחזיר את האובייקט (במקרה שלנו, זה מחזיר מחרוזת רלוונטית מתוך *content* לפי המיקום)
- `getItemId(int position)` - מקבל את המיקום ומחזיר את ה- ID (במקרה שלנו הוא מחזיר 0)

הפונקציה `getView(int position, View convertView, ViewGroup parent)`:

הפונקציה מקבלת את המיקום ואת ה- *View*

במקרה שלנו אנחנו נקבל את המיקום ונבדוק האם קיבלנו *View*, במקרה שאכן קבלנו *View* אזי נוסיף אותו ל- *GridView* ובמקרה שלא קבלנו *View* (*convertView = null*) נוסיף אותו *Activity Item* ונעדכן את תצוגת הכפתור והטקסט באלמנט המתאים מתוך הרשימה.

לבסוף נממש את המתודה *OnCreate* של *MainActivity.java*.

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

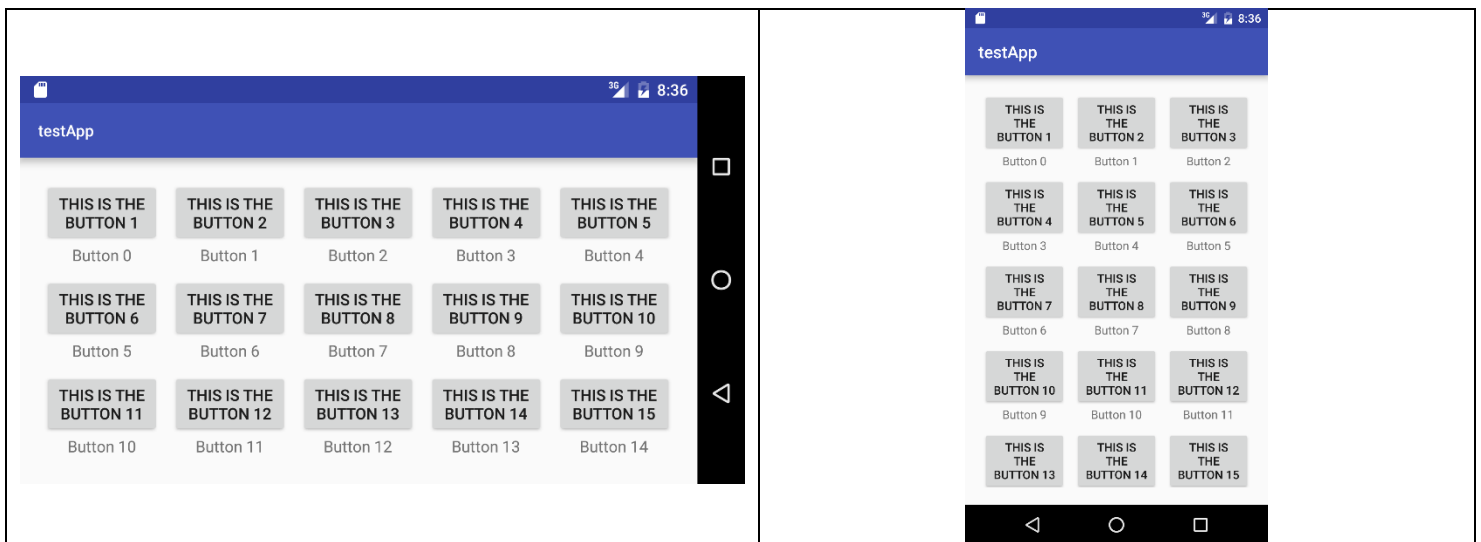
    List<String> text = new ArrayList<String>();

    for (int i =0;i<50;i++)
        text.add("This is the Button " + (i+1) );

    GridView mygridview = (GridView) findViewById(R.id.mygridview);
    mygridview.setAdapter(new ItemsAdapter(this,text));
}
```

כמו שרואים, יצרנו רשימה ולולאה שמכניסה 50 מחרוזות לרשימה. לאחר מכן הגדרנו את ה *Adapter* של ה *GridView* ע"י המחלקה שלנו *ItemsAdapter* שמקבלת בבנאי שלה *this* ואת הרשימה.

והתוצאה שנקבל תהיה:



האופציה ההיברידית - רכיב WEBVIEW

רכיב ה- WebView הינו הרחבה של רכיב ה- View ונועד להציג דף אינטרנטי מתוך Activity לדוגמא, להציג את המפות של גוגל... אנחנו עושים שימוש ברכיב שהוא בעצם דפדפן. מדובר בדפדפן לצורת עבודה זו קוראים "האופציה ההיברידית" כותבים את ממשק המשתמש בקוד בג'בא ודואגים שהקוד ייצור מחלקה מסוג webview. ה- webview נשען על דפדפן בסיסי WebKit שהינו Open Source Project ושחברת Apple עומדת מאחוריו. אגב, את הדפדפנים ספארי וכרום בנו על בסיס הדפדפן WebKit. בתוך ה- Webview ניתן להציג קוד WEB כמו למשל HTML או JavaScript או CSS, תמונות והלאה. קוד WEB **לא חייב** להגיע מצד שרת אלא יכול להיות קוד WEB שאני אורז אותו יחד עם האפליקציה. (יש כאלה שטועים וסוברים שקוד WEB יכול להגיע רק מצד שרת או צריכים לדעת שזה יכול להיות חלק מהאפליקציה עצמה) אם מריצים JavaScripts שרץ בתוך ה- webView אנחנו יכולים לקבל ממשקי משתמש מאוד מרשימים. ניתן לראות דוגמאות מאוד מרשימות בקישור:

<http://www.iqmgallery.com/>

כמו כן כדאי לעיין ולהתרשם גם בקישורים הבאים:

<http://www.elated.com/articles/jquery-mobile-what-can-it-do-for-you/>

<http://demos.telerik.com/kendo-ui/>

<http://www.telerik.com/kendo-ui>

האפשרות לפתח ממשק משתמש בצורה הזאת של visual designer של drag&drop היא הרבה יותר מהירה מאשר לפתח את ממשק המשתמש בג'בא. היתרון האדיר הוא שזה עובד על כל פלטפורמה. הקריאות מבפנים החוצה של פונקציות ה- javascript משתנות כמובן ע"פ המכשיר. אגב, ה- HTML5 קיבל דחיפה מגוגל ואפל והוא נוצר על בסיס ה- HTML גרסה 4.0 עם תוספת של תגים שמייצג Video או audio אבל יותר מכך עם תוספת של API ב- JavaScript שמאפשר ביצוע פעולות מאוד מתוחכמות.

עיינ בגרסא האחרונה של WebView שמכילה יכולות מדהימות

https://www.google.co.il/search?rls=aso&client=gmail&q=chromium+WebView+&authuser=0&gws_rd=cr&ei=FxeUVtmuFcfqPozJj6gM

https://docs.google.com/presentation/d/1pYAGn2AYJ7neFDIDZ9DmLHpWMIskzMUXjFXyR7yfUko/edit?pli=1#slide=id.g38ade9ef5_041

עם כל זאת עדיין ישנם מקרים שהפיתוח בג'אווה בנסיבות הבאות:

- כשיש מגבלות של כח מיחשוב כמו למשל על מכשירים זעירים שאנדרואיד עובד עליהם.
- אם יש קוד שרץ ברקע זה מכביד על המכשיר.
- רוצים אפליקציה ברמת תגובה גבוהה.
- קוד לדוגמא שפונה ליכולות המובנות של הפלטפורמה כמו למשל לגשת למצלמה המובנית במכשיר.
- אי אפשר לפתח אנימציות בטכנולוגיות ווביות.
- לפתח Launcher (שמראה את הצלמיות על המסך) ניתן רק בג'אווה.
- קוד שאוסף נתונים מסנסורים ניתן לכתוב רק בג'אווה.
- אתה רוצה לפתח מקלדת חדשה למשל מקלדת עבור אנשים מוגבלים אתה חייב לפתח רק בג'אווה.
- הודעות שמתקבלות משרת לאפליקציה ספציפית אתה צריך לדעת ג'אווה בכדי לבצע את הדברים האלה.
- כל הפעולה עם intents לא ניתן לבצע בטכנולוגיות ווביות.
- יש מרכיבים בממשק המשתמש של טבלט שלא ניתן ליצור בטכנולוגיות ווביות. כמו למשל שימוש בפרגמט שזה מאפשר – בין היתר – שימוש גמיש גם בטלפון חכם וגם בטבלט ובשביל זה צריך לדעת ג'אווה.
- ממשק עם action bar חייבים לעשות בג'אווה ולא בטכנולוגיות ווביות.

דוגמא לשימוש ברכיב WebView

מקור הדוגמא:

<http://www.mkkyong.com/android/android-webview-example/>

כיצד ניתן להשתמש ברכיב WebView ?

ראשית יוצרים את תצוגת רכיב ה-WebView כקובץ XML בתוך Layout->Res

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <WebView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:id="@+id/webView1"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6 />
```

שורות 2-6: יש לציין את הרכיב ה-WebView כבר כשמתחילים להקליד ה-Android Studio מזהה את הרכיב. קובץ XML הזה יצורף ל- res-layout. בעצם הגדרנו בקובץ רכיב WebView שתופס את כל העמוד, אבל אפשר גם להוסיף רכיב WebView בתוך כל Layout שנרצה בדיוק כמו שמוסיפים כפתור למשל.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical" >
6     <Button
7         android:id="@+id/buttonUrl"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:text="Go to http://www.google.com" />
11 </LinearLayout>

```

שורות: 6-10 – נגדיר כפתור

נכתוב את ה- Activity של החלון הראשי (הקובץ MainActivity.java) כדלהלן:

```

1 import android.app.Activity;
2 import android.content.Context;
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.view.View.OnClickListener;
7 import android.widget.Button;
8 public class MainActivity extends Activity {
9     private Button button;
10    public void onCreate(Bundle savedInstanceState) {
11        final Context context = this;
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.main);
14        button = (Button) findViewById(R.id.buttonUrl);
15        button.setOnClickListener(new OnClickListener() {
16            @Override
17            public void onClick(View arg0) {
18                Intent intent = new Intent(context, WebViewActivity.class);
19                startActivity(intent);
20            }
21        });
22    }
23 }

```

שורות 17-19: יצירת Intent שמעבירה אותנו ל- Activity השני שמכיל את ה- WebView כפי שמוצג בעמוד הבא.

```

1  import android.app.Activity;
2  import android.os.Bundle;
3  import android.webkit.WebView;
4  public class WebViewActivity extends Activity {
5      private WebView webView;
6      public void onCreate(Bundle savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.webview);
9          webView = (WebView) findViewById(R.id.webView1);
10         webView.getSettings().setJavaScriptEnabled(true);
11         webView.loadUrl("http://www.google.com");
12         //String customHtml = "<html><body><h1>Hello, WebView</h1></body></html>";
13         // webView.loadData(customHtml,"text/html","UTF-8");
14     }
15 }

```

שורה 8: הצגת ה-WebView בתוך ה-Activity.

שורה 9: הפעלת תצוגת ה-WebView לאובייקט.

שורה 10: בפקודה זו מאפשרים הטמעת קוד JavaScript בתוך ה-WebPage. כלומר אם דף ה-WebPage שלנו מכיל גם פקודות JavaScript נצטרך להגדיר את ה-WebView שלנו שיוכל לעבוד עם JavaScript. נבצע זאת על ידי הפונקציה: `setJavaScriptEnabled`.

שורה 11: חיבור רכיב ה-WebView (שמשמש כדפדפן מובנה בתוך ה-Activity) ל-WebPage באמצעות ה-URL שלו.

שורות 12-13: ניתן ליצור WebPage (כמחרוזת בצורה ידנית) ולהציג אותו בתוך ה-WebPage באמצעות המתודה `loadData`.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android=http://schemas.android.com/apk/res/android
3  package="com.example.ezra.webviewexample"
4  android:versionCode="1"
5  android:versionName="1.0" >
6  <uses-sdk android:minSdkVersion="10" />
7  <uses-permission android:name="android.permission.INTERNET" />
8  <application
9      android:icon="@drawable/ic_launcher"
10     android:label="@string/app_name" >
11     <activity
12         android:name=".WebViewActivity"
13         android:theme="@android:style/Theme.NoTitleBar" />
14     <activity
15         android:label="@string/app_name"
16         android:name=".MainActivity" >
17         <intent-filter >
18             <action android:name="android.intent.action.MAIN" />
19             <category android:name="android.intent.category.LAUNCHER"
20             />
21         </intent-filter>
22     </activity>
23 </application>
</manifest>

```

שורה 7: מכיוון שאנדרואיד חוסם בברירת מחדל את ההרשאות של האפליקציות לגשת לרשת, נצטרך להוסיף בקובץ AndroidManifest.xml שלנו את הקוד הבא על מנת שנוכל לגשת לאינטרנט ולכן חייבים להוסיף עבור כל אפליקציה שפונה לרשת האינטרנט חייבים להוסיף בקובץ ה-manifest את ה-permission בשם: **android.permission.INTERNET**