תדריך

הטכניון – מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל

המעבדה למערכות תוכנה

Android Programming

Version 6.0

הניסוי תוכנן ע"י:

ד"ר אילנה דוד

ויקטור קוליקוב

הניסוי עודכן ע"י:

ויקטור קוליקוב

התדריך מנוסח בלשון זכר אך מיועד לגברים ונשים כאחד.

בטיחות

כללי:

הנחיות הבטיחות מובאות לידיעת הסטודנטים כאמצעי למניעת תאונות בעת ביצוע ניסויים ופעילות במעבדה למערכות תוכנה. מטרתן להפנות תשומת לב לסיכונים הכרוכים בפעילויות המעבדה, כדי למנוע סבל לאדם ונזק לציוד. אנא קראו הנחיות אלו בעיון ופעלו בהתאם להן .

מסגרת הבטיחות במעבדה:

- אין לקיים ניסויים במעבדה ללא קבלת ציון עובר בקורס הבטיחות של מעבדות ההתמחות באלקטרוניקה (שהינו מקצוע קדם למעבדה זו).
- לפני התחלת הניסויים יש להתייצב בפני מדריך הקבוצה לקבלת תדריך ראשוני.

הנחיות בטיחות:

- אין לקיים ניסויים במעבדה ללא השגחת מדריך ללא אישור מראש.
- מדריך הקבוצה אחראי להסדרים בתחום פעילותך במעבדה; נהג על פי הוראותיו.

עשה ואל תעשה:

- יש לידע את המדריך או את צוות המעבדה על מצב מסוכן וליקויים במעבדה או בסביבתה הקרובה.
- לא תיעשה במזיד ובלי סיבה סבירה, פעולה העלולה לסכן את הנוכחים במעבדה.
- אסור להשתמש לרעה בכל אמצעי או התקן שסופק או הותקן במעבדה.
- היאבקות, קטטה והשתטות אסורים. מעשי קונדס מעוררים לפעמים צחוק אך הם עלולים לגרום לתאונה.
- אין להשתמש בתוך המעבדה בסמים או במשקאות אלכוהוליים, או להיות תחת השפעתם.
- אין לעשן במעבדה ואין להכניס דברי מאכל או משקה.
- בסיום העבודה יש להשאיר את השולחן נקי ומסודר.
- בניסיון לחלץ דפים תקועים במדפסת - שים לב לחלקים חמים!

בטיחות חשמל:

- בחלק משולחנות המעבדה מותקנים בתי תקע ("שקעים") אשר ציוד המעבדה מוזן מהם. אין להפעיל ציוד המוזן מבית תקע פגום.
- אין להשתמש בציוד המוזן דרך פתילים ("כבלים גמישים") אשר הבידוד שלהם פגום או אשר התקע שלהם אינו מחוזק כראוי.
- אסור לתקן או לפרק ציוד חשמלי כולל החלפת נתיכים המותקנים בתוך הציוד; יש להשאיר זאת לטפול הגורם המוסמך.
- אין לגעת בארון החשמל המרכזי, אלא בעת חירום וזאת - לצורך ניתוק המפסק הראשי.

מפסקי לחיצה לשעת חירום:

- במעבדה ישנם מפסקים ראשיים להפסקת אספקת החשמל. זהה את מקומם.
- בעת חירום יש להפעיל מפסקי החשמל הראשיים.

תדריך

בטיחות אש, החייאה ועזרה ראשונה:

- במעבדה ממוקמים מטפי כיבוי אש זהה את מקומם.
- אין להפעיל את המטפים, אלא בעת חירום ובמידה והמדריכים וגורמים מקצועיים אחרים במעבדה אינם יכולים לפעול.

יציאות חירום:

- בארוע חירום הדורש פינוי, כגון שריפה, יש להתפנות מיד מהמעבדה.

דיווח בעת אירוע חירום:

- יש לדווח **מידית** למדריך ולאיש סגל המעבדה.
- המדריך או איש סגל המעבדה ידווחו מיידית לקצין הביטחון בטלפון; 2740, 2222.
  במידה ואין הם יכולים לעשות כך, ידווח אחד הסטודנטים לקצין הביטחון.
- לפי הוראת קצין הביטחון, או כאשר אין יכולת לדווח לקצין הביטחון, יש לדווח, לפי הצורך:

  משטרה 100,

  מגן דוד אדום 101,

  מכבי אש 102,

  גורמי בטיחות ו/או ביטחון אחרים.

  בנוסף לכך יש לדווח ליחידת סגן המנמ"פ לעניני בטיחות; 2146/7, 3033.

- בהמשך, יש לדווח לאחראי משק ותחזוקה; 4776
- לסיום, יש לדווח ל:

  מהנדס המעבדה (טל. 3220)

  בעת הצורך ניתן להודיע במקום למהנדס המעבדה לטכנאי המעבדה.

מבוא

חוברת זו מהווה תדריך והכנה לניסוי בתכנות בסביבת Android. הניסוי מעודכן לגרסת Android 6. בתדריך זה נפרט את מטרות הניסוי, מבנהו, ומבנה הציון .

התדריך מכיל חומר רקע תיאורטי אותו יש לקרוא, וחומר רקע ויזואלי המקושר לאינטרנט ומכיל קטעי וידאו בהם יש לצפות **כחלק מההכנה לניסוי**.

כמו כן, התדריך כמובן מקוצר ואינו מכיל את כל המתודות אליהן תידרשו בניסוי. לשם כך עומד לרשותכם מאגר API גדול שזמין באינטרנט ותצטרכו להשתמש בו גם עבור הדוח המכין וגם במהלך הניסוי.

דרישות קדם הכרחיות לביצוע הניסוי :

[מבוא למערכות תכנה 044101 (או) מבוא לתכנות מערכות 234122]

(וגם)

[ניסוי Java במסגרת מעבדה 2/3 בחשמל (או) 046271 תכנות ותכן מונחה עצמים]

הערה חשובה: התדריך מניח היכרות בסיסית עם שפת Java ובהכרח הבנה של מילות המפתח הבאות: Class, Subclass, Abstract Class, Interface, של ההבדלים בין (extends ירושה) לבין implements (מימוש), וכן של חוקי דריסה והעמסה ב Java-ופולימורפיזם .

אם אינכם בקיאים בנ"ל, מומלץ לחזור ולהתרענן בחומר לפני קריאת תדריך זה.

מטרות הניסוי

- היכרות עם צד הפיתוח של מערכת ההפעלה Android באמצעות Java, והייחודיות שבפיתוח למערכות מבוססות אנדרואיד.
- היכרות עם סביבת הפיתוח של Android Studio.
- היכרות עם פיתוח סביבת ממשק משתמש גרפי (GUI) באמצעות קבצי XML.
- התנסות בכתיבת אפליקציות קצרות מבוססות Activity.

מבנה הניסוי

- קריאת התדריך ולימוד החומר התיאורטי.
- צפייה בסרטונים בקישור הרלוונטי.
- פתרון שאלות הכנה במסגרת דו"ח מכין. יש להגיש את הדו"ח מודפס בתחילת הניסוי.
- בוחן הכנה לניסוי (מבוצע בזוגות).
- ביצוע הניסוי מול המחשב ומכשיר ה-Android במעבדה, בשני חלקים בני 4 שעות כל אחד.

מבנה הציון

הציון על הניסוי מורכב כדלקמן:

- דו"ח מכין – 10%
- בוחן – 10%
- ביצוע הניסוי – (כתיבת תכנית ללא תקלות, הספק) – 80%

תדריך

הניסוי מחולק לתרגילים ברמות קושי שונות.

דרישות דו"ח מכין

לפני ההגעה למפגש הראשון של הניסוי, יש לקרוא את התדריך במלואו ולהבינו לעומק.

כמו כן, יש לצפות בקטעי הווידאו המצורפים ולהבינם.

התדריך אינו מחייב להתקין את המערכת בבית ולנסות (ההתקנות וההתנסות הממשית יתקיימו במסגרת הניסוי), אך **מומלץ בחום** להתקין גרסת Android Studio על המחשב בבית (כמפורט בסעיף הבא) ולהתנסות תוך כדי קריאה. הדבר יחסוך זמן רב ועשוי להביא להצלחה גדולה יותר של הניסוי.

לאורך התדריך ישנם קישורים (Hyperlinks) על-גבי מילים חשובות – ניתן להקליק על קישורים אלו (בגרסת ה PDF-) כדי לקבל מידע נוסף.

לנוחיותכם, בחוות המחשבים מותקנת גרסת Android Studio וניתן גם להתנסות עליה.

חומר רקע – וידאו

סרטון הדגמה

להלן קישור לסרט וידאו (באורך כשעתיים), בו מסביר Marko Gargenta איך מתחילים לכתוב אפליקציה באנדרואיד.

הצפייה בסרט הינה חלק מדרישות ההכנה .

היא מדגימה איך בפועל מפתחים לאנדרואיד ונותנת תחושה מעשית שמשלימה את חומר הקריאה .

http://www.youtube.com/watch?v=rm-hNlTD1H0

שימו לב: ההרצאה בווידאו היא מעט ישנה, על גרסת Android 1.6 ו .Eclipseיתכנו שינויים.

קורס וידאו מGoogle https://developer.android.com/training/index.html

מומלץ מאוד

הגדרת סביבת עבודה על המחשב האישי

להלן קישור לסרטון של המעבדה בו מתואר שלב אחר שלב איך ניתן להתקין את כל סביבת העבודה על המחשב האישי:

- התקנת JDK
- התקנת Android Studio

לתשומת לבכם, התקנת הנ"ל על המערכת עשויה לקחת זמן מה (כמה שעות, כתלות באיכות המחשב, בטיב החיבור לאינטרנט, ובכמות הגרסאות שברצונכם להתקין.)

https://developer.android.com/studio/install.html

חומר רקע – תיאוריה

מבוא

Androidהינה מערכת תוכנה מבוססת Linux המיועדת למכשירים ניידים (בעיקר מבוססי מגע) כגון טלפונים ומחשבי לוח (Tablets). המערכת מופצת על-ידי חברת.Google

Androidהיא מערכת הבנויה בשכבות (Layers). מערכת Android כוללת מערכת הפעלה, תשתית לבניית אפליקציות, מכונה וירטואלית להרצת האפליקציות, ספריות גרפיות, בסיס נתונים, תמיכה במדיה, טלפוניה, רשת אלחוטית, מצלמה, ניווט, GPS חיישנים - וכן סביבת פיתוח עשירה המשתלבת במערכת הפיתוח Android Studio.

כאמור, Android מבוססת על ליבה של מערכת ההפעלה .Linux היא מגיעה עם קבוצת אפליקציות בסיסיות הכוללות : אפליקציה לדואר אלקטרוני, מסרונים (SMS), לוח שנה, מפות, דפדפן, אנשי קשר ועוד .

מרבית האפליקציות ל Android-כתובות בשפת Java, אך לא רק – ישנן גם אפליקציות הכתובות בשפת C ו-C++.בניסוי זה נתמקד, כאמור, ב.Java-

סביבת הפיתוח הפתוחה מאפשרת למפתחים לבנות אפליקציות חדשניות ועשירות בקלות:

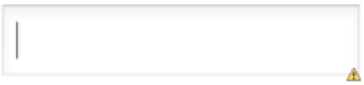
המבנה הייחודי של Android מאפשר לנהל בנוחות את הגישה לחומרת המכשיר, וכן לרכיבים הפריפריאליים שלו (למשל, מערכת המיקום, GPS – מערכת הודעות, התראות, וכיוצא באלה .)

באופן כזה, ניתן להריץ שירותים (Services) ברקע ועוד הרבה אופציות אחרות. כל זאת, תוך שמירה על אבטחת המידע של המשתמש, באופן שבו כל גישה כזאת דורשת את אישורו של המשתמש (בעת התקנת האפליקציה.)

מערכת Android מספקת לנו ממשק משתמש גרפי (GUI) קל לתחזוקה וליצירה באמצעות קבצי .XML את הממשק הגרפי ניתן ליצור באמצעות "אבני בניין" של אלמנטים גרפיים, פשוטים ומסובכים כאחד, הנקראים (Viewsאו Widgets [1] .(לדוגמא כפתורים, תיבות טקסט, וכו', כמתואר:

Button: 

Text View: Hello world!

Text Field: 

ועוד שלל אפשרויות שונות.

האלמנטים הללו מתחברים לכדי Layout באופן נוח להגדרה כפי שתראו בסרטון ההדרכה.

קישור בין אפליקציות גם הוא מתבצע בצורה קלה, וניתן לגשת למידע של אפליקציות אחרות בכמה שיטות, כגון באמצעות ספקי תכן (Content Providers), או על ידי שיתוף המידע של האפליקציה באופן ישיר.

רכיבי האפליקציה

[1] המונח Widget הוא מונח מבלבל וחשוב לשים לב באיזה הקשר נאמר. בהקשר של Web או תוכנה, לרוב המונח מציין אפליקציה קטנה בעלת תכלית ייעודית. בהקשר של ממשק גרפי (GUI) המונח מציין, כאמור, אלמנט גרפי בודד.

תדריך

רכיבי האפליקציה (Application Components) הם למעשה "אבני הבניין" הבסיסיים של ליבת האפליקציה. כל רכיב מהווה נקודת גישה דרכה המערכת יכולה לגשת לאפליקציה; לא כל רכיבי האפליקציה מהווים נקודת גישה למשתמש, וחלקם תלויים אחד בשני – אבל כל אחד מהם מתקיים כישות נפרדת (יכול להיות מופעל באופן אינדיבידואלי) ויש לו תפקיד ספציפי שעוזר לאפיין את התנהגות האפליקציה .

קיימים ארבעה סוגים של רכיבי אפליקציה: כאמור, לכל אחד מהם תכלית שונה ומחזור חיים ייחודי שמגדיר כיצד הוא נוצר ונהרס. להלן נפרט את ארבעת סוגי הרכיבים, כאשר בניסוי זה נתמקד ברכיבים מסוג **Activity.** את הרכיבים האחרים נכיר על קצה המזלג בלבד.

SERVICES

Service הוא רכיב שרץ ברקע ומאפשר ביצוע של תהליכים שרצים למשך זמן רב, או עבודה עבור תהליכים מרוחקים. Services אינם מספקים ממשק משתמש – למשל, Service של מוזיקה יכול לנגן מוזיקה ברקע בזמן שהמשתמש נמצא באפליקציה אחרת; דוגמא נוספת - Service יכול להוריד קובץ גדול מהאינטרנט בזמן שהמשתמש נמצא באפליקציה אחרת, וכו .'

את ה Service-מפעילים, בדרך כלל, מרכיב אחר (כגון.(Activity

CONTENT PROVIDERS

Content Providers מנהלים אוסף של מידע משותף עבור אפליקציות. ניתן לאחסן את המידע במספר דרכים, כגון במערכת הקבצים, במסד נתונים (כמו ,(SQLite ברשת, וכו'. באמצעות ה Content Provider-אפליקציות אחרות יכולות לגשת למידע ואף לשנותו .

למשל, במערכת ה Android-כל המידע הקשור לאנשי הקשר של המשתמש מנוהל על-ידי .Content Provider כל אפליקציה עם ההרשאות המתאימות יכולה לגשת למידע השמור במאגר הנתונים הנ"ל.

BROADCAST RECEIVERS

Broadcast Receivers הינם רכיבים שמגיבים לכריזות (Broadcast Announcements) מערכת. למשל, כריזה שמעדכנת על הדלקת המסך, על בטרייה אוזלת, וכו'. אפליקציות יכולות גם ליזום כריזות (כמו למשל כריזה שמודיעה על קובץ שסיים לרדת מהאינטרנט) וגם להאזין להן.

מעתה נתמקד רק ב.Activities-

ACTIVITIES

כללי

Activity הינה מחלקה המייצגת מסך בודד עם ממשק משתמש.

לדוגמא, באפליקציית דואר אלקטרוני יתכנו מספר מופעים של - Activity אחד המראה את רשימת הודעות הדואר החדשות, השני לכתיבת הודעת דואר חדשה, ואחר לקריאת הודעות - כל אחד מהם עובד בנפרד ואינו תלוי באחרים. אי-התלות הזאת הינה מרכיב המסייע רבות לתכן נכון ולגמישות אפליקציות, ומאפשרת שיתוף Activities בודדים בין אפליקציות – למשל, אפליקציה של מצלמה שיכולה לגשת ישירות למופע של Activity באפליקציית הדואר האלקטרוני שאחראי על שליחת מייל, כדי שהמשתמש יוכל לשלוח את התמונות שצילם.

מתודות CALLBACK

אוסף המתודות שאליהן קוראת המערכת כאשר Activity עובר מצב במחזור החיים שלו נקרא **מתודות CallBack.** מעבר מצב במחזור החיים של Activity מתבצע, למשל, כאשר Activity נוצר, עוצר, מתחיל מחדש (לאחר מעבר ל Activity-אחר) או מושמד .

מתודת ה CallBack-העיקרית הינה המתודה onCreate() אשר לפחות אותה חייבים לממש. היא נקראת כאשר נוצר Activity חדש, ובמימוש שלה אנו מאתחלים את הרכיבים החיוניים לאפליקציה .

בנוסף, במתודה onCreate() אנו נקרא למתודה setContentView(), שמקשרת את המסך ל Layout-הרלוונטי שמוגדר על-ידי קובץ ה XML-(כפתורים, קופסאות טקסט, תמונות וכו'). **דוגמא מוחשית לכך תינתן בהמשך, וניתן לראות גם בסרטון הוידאו.**

INTENT

אפליקציה ב Android-יכולה להתחיל ולהפעיל אפליקציה אחרת, או שירות מערכת.

באופן א-פורמאלי, - Intent כנובע משמו - מייצג את ה"כוונה" של אפליקציה 'לעשות משהו'. למעשה, מדובר בלא יותר מאשר הודעה של אפליקציה שהיא 'עשתה משהו' או רוצה ש'משהו יקרה'. כתלות ב Intent, -אפליקציות נוספות – או מערכת ההפעלה – יאזינו לו, ויגיבו בהתאם.

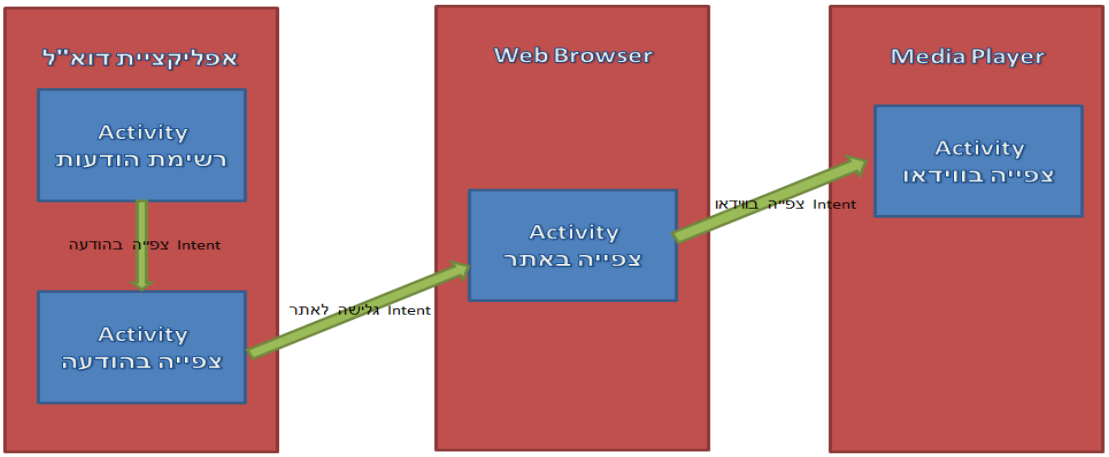
ניתן כדוגמה אפליקציית צילום שרוצה לצלם באמצעות מצלמת המכשיר. למרות שמדובר באפליקציה נפרדת, האפליקציה הראשונה תוכל לקבל את התמונה כאילו היא כתבה את קוד הצילום ויצירת התמונה בעצמה. בפועל, בעצם מדובר בריצה של שני Activities נפרדים בשני תהליכים שונים .

המעבר מתהליך לתהליך מתבצע באמצעות מערכת ההפעלה, כאשר אפליקציית הצילום פונה למערכת ההפעלה ומעבירה אליה אובייקט מסוג.Intent

האובייקט המועבר מכיל את המידע הדרוש להפעלת רכיב האפליקציה האחר - כמו איזו פעולה להפעיל, וכל מה שהפעולה המופעלת חייבת לדעת בנוסף .

דוגמאות: בקשה לתמונה, כתובת אתר, WEB מחרוזת כלשהי, הודעה להפצה ועוד.

את ההקשר בין Activity לבין Intent נראה באמצעות הדוגמא הבאה: נניח שאנו משתמשים באפליקציית דואר אלקטרוני וצופים בהודעות שקיבלנו. אנו פותחים הודעה מרשימת ההודעות, ובה יש לינק לאתר שבו יש תוכן וידאו .(Streaming) אנו לוחצים על קישור זה והווידאו מוצג בנגן המדיה. להלן תרשים אבסטרקטי המייצג את הקשר בין האפליקציות השונות באמצעות:Intent



קיימים שני טיפוסים של - Intent מפורש ומרומז. ב Intent-מפורש (Explicit) השולח מציין במפורש את שם יעד ההודעה. במקרה של Intent מרומז (Implicit) השולח לא מציין את יעד ההודעה, אבל לפי הפרמטרים שבה, ההודעה מכוונת ע"י מנגנון המכונה Intent Filter ליעד המתאים. מנגנון זה מאפשר קוד גמיש בו אין צורך לדעת מראש את זהות היעד של הודעה.

משאבי האפליקציה

האבסטרקציה אותה מערכת ההפעלה Android מספקת לנו כמפתחים היא הרבה יותר מהאפשרות לתכנת ב-Java את האפליקציות שלנו ואספקת פונקציות ספריה.

כל אפליקציית Android מורכבת מ**קבצי מקור** (Source Code Files, קבצי Java) ומ**קבצי משאבים**[2], עליהם נדבר בסעיף זה.

קבצי המשאבים מספקים מודולריות משמעותית לאפליקציות אנדרואיד באופן שבו ניתן לשנות ולהחליף את המשאבים ללא צורך לשנות את הקוד – למשל Layouts שונים לגדלי מסך שונים. דוגמא נוספת: אפשר להוסיף ארבע וריאציות לסמל של האפליקציה – כל וריאציה באיכות תמונה אחרת וכתלות בכמות הפיקסלים וגודל המסך של הטלפון שמריץ את האפליקציה יוצג סמליל אחר.

לכל משאב יש מזהה ייחודי (כגון מספר או מחרוזת) הנקרא Resource ID שניתן לפנות אליו ישירות מהקוד.

קובץ ה-MANIFEST

לכל אפליקציה חייב להיות קובץ בשם **AndroidManifest.xml**.

קובץ זה מייצג את כל המידע החיוני על האפליקציה כלפי מערכת האנדרואיד, לפני שהמערכת תוכל להריץ את הקוד. הקובץ מציין את שם החבילה (package,) המשמש כמזהה ייחודי לאפליקציה.

בקובץ מוצהרים כל הרכיבים מהם מורכבת האפליקציה: ה Activities, Services, Broadcast Receivers-ו Content Providers, וכן התכונות והיכולות שלהם.

[2] משאב (Resource) בהקשר זה הינו אבסטרקציה של רכיב שאינו תלוי מוחשית בקוד. למשל קבצי תמונה ואודיו – אך לא רק - Layouts הינם משאבים, כדי שנוכל להחליף בקלות את ממשק המשתמש הגרפי או להציע מספר אופציות; גם מחרוזות טקסט הינן משאב (כדי שנוכל, למשל, לתרגם את התכנית לכמה שפות בקלות.)

תדריך

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="il.ac.technion.ee.nssl.example">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Example"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

בקובץ ה Manifest-מגדירים את ההרשאות (Permissions) של האפליקציה בגישה להתקני המערכת.
למשל: גישה לאינטרנט, גישה לאנשי קשר, מצלמה וכו '.

כמו כן, מגדירים בקובץ מספר נתוני מעטפת חשובים, כגון גרסת אנדרואיד מינימלית שדרושה, הקונפיגורציה של החומרה שנדרשת, רשימת הספריות שיש ליצור קשר אליהן, ועוד.

להלן דוגמא לקובץ Manifest של אפליקציה המכילה שני Activities, ומשתמשת במספר רב של הרשאות:

קובץ ה-R

קובץ ה R-מקשר בין עולם קוד המקור (עולם ה JAVA) לעולם המשאבים .

הוא נוצר עבורנו באופן אוטומטי (על-ידי תוכנת ה Android Studio)-ואין כל סיבה לשנות אותו: בכל פעם שמשנים משהו בספריות הרלוונטיות (למשל מוסיפים קובץ XML), קובץ ה R-מתעדכן.

כמפתחים, אין לנו כל צורך להסתכל בקובץ זה; מערכת ה Android Studio-מאפשרת להתייחס לערכים השמורים בו ונותנת לנו ממשק נוח לשינויים שבו.

ממשק המשתמש הגרפי – GUI

כאמור, האפליקציה של אנדרואיד נמצאת בשני סוגי קבצים: קבצי משאבים וקבצי קוד מקור.

בין קבצי המשאבים ישנם קבצי XML עבור הממשק הגרפי. קבצים אלו מתארים את ה Layout-(אופן הפריסה) של הצמתים המשתתפים בהיררכיה ואת תכונותיהם .

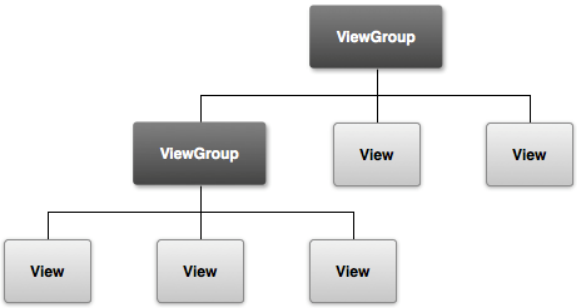
גם בקובץ קוד המקור (Java) אנו מגדירים את האלמנטים הגרפיים, אולם ההבדל בין הקבצים הוא מהותי: בקובץ ה XML-עדיף להגדיר את הפריסה הסטטית של האלמנטים (כלומר, היכן כל אלמנט מופיע על המסך), ואת ההתנהגות הדינאמית בקבצי הקוד (למשל, כיצד הכפתור יגיב כאשר ילחצו עליו – האם ישנה את הטקסט.(?

VIEW ,VIEWGROUP

כל אלמנטי ה GUI-באפליקציית Android מורכבים מאובייקטים שהם מופעים של המחלקות View ו-ViewGroup.

אובייקט View הוא אובייקט שמצייר על המסך אלמנט שהמשתמש יכול לעשות איתו אינטראקציה כלשהי (למשל, כפתור או תיבת טקסט). בתוך האובייקט יש מבנה נתונים השומר בתכונותיו את הפרמטרים של המבנה והתוכן של מלבן מסוים על המסך. הוא מטפל במידות של עצמו, בפריסה שלו על פני המסך, בציור, שינוי הפוקוס וכו' בתוך המלבן בו הוא יושב. בעזרתו נעשית האינטראקציה עם אירועי המשתמש.

ViewGroupהוא אובייקט מאגד שמכיל בתוכו מופעים של) View וגם של (ViewGroup כדי להגדיר את ה-Layoutשל הממשק (היכן יוצג איזה אובייקט), באופן הבא[3]:



קובץ ה XML-של ה Activity-הראשי של הפרויקט נמצא בקובץ : res/layout/main_activity.xml.

כדי לגרום לאפליקציה להציג על המסך את ה Layout-הרלוונטי אנו קוראים למתודה setContentView(), ונותנים לה כפרמטר את ה **Resource ID**-של ה Layout-למשל:

```
setContentView(R.layout.main_activity);
```

(חישבו לעצמכם : מהו ה ID-במקרה זה? באיזה קובץ הוא מאוחסן? למה?)

הספריות הסטנדרטיות ב Android-מספקות אוסף של מחלקות יורשות מ View-ומ ViewGroup-שמציעות אלמנטים בסיסיים לעבוד איתם (כמו כפתורים ושדות טקסט, ואף Layouts יחסיים או אבסולוטיים – כפי שתוכלו להתרשם מצפיה בווידאו ובמהלך הניסוי.)

LAYOUT[4]

הדרך המקובלת להגדיר את פריסת ה Views,-כלומר את ה - ViewGroup-היא בעזרת קבצי XML.

ה ViewGroup-הנפוץ ביותר הוא Layout מהסוגים, LinearLayout ConstraintLayout ו Relative Layout,-כאשר, כמובן, כל אחד מהם יכול להכיל מופעים נוספים.

---

[3] למתעניינים, שיטה זו הינה וריאציה של מימוש Design Pattern שנקרא.Composite

[4] המונח Layout הוא גם מונח מבלבל בו משתמשים, לסירוגין, לתיאור שני דברים: האחד הוא קובץ ה XML-עצמו שנקרא "Layout" עבור כל .Activity השני הוא, בתוך קובץ ה XML,-מופע של ViewGroup שמכיל בתוכו GUI Widgetsואף עוד מופעי Layout, כמתואר בשרטוט בסעיף קודם.

תדריך

קובץ ה XML-מציע מבנה קריא של מערך הפריסה, ומבנהו בדומה ל.HTML-באופן זה, קל לתאר את עץ הרכיבים בדומה לשרטוט בסעיף קודם. כל שם ב XML-מייצג אלמנט ממחלקה עם שם תואם בתוכנית ה-Java.

בזמן טעינת קובץ ה) XML-כפי שהראינו לעיל), מערכת ה Android-מאתחלת את האובייקטים שיוצגו בזמן הריצה בהתאמה לאלמנטים בקובץ ה.XML-

את קובץ ה XML-ניתן לערוך באופן טקסטואלי או באמצעות ממשק GUI נוח מאוד שה Android Studio-מספק.

הקובץ הבא הינו דוגמא המתארת LinearLayout אנכי הממלא את גובה ורוחב המסך, והמכיל בתוכו תיבת טקסט (TextView) וכפתור. הכפתור והטקסט תופסים רק את הרוחב הדרוש לטקסט המוצג בהם.

```
<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" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am some Text!"
        tools:context=".MainActivity" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />

</LinearLayout>
```

INPUT EVENTS

האינטראקציה עם האובייקטים של הממשק נעשית בעזרת אירועים (Events) שגורמים לביצוע פעולות.

המחלקה View מכילה אוסף של ממשקים (Java Interfaces) הנקראים Listeners.[5]

שמו של כל ממשק X הוא: On<X>Listener, כאשר הביטוי <X> מציין את סוג האינטראקציה עם המשתמש, למשל:

- הממשק View.OnClickListener – ממשק המאפשר לאלמנט להגיב לאירוע לחיצת כפתור
- הממשק View.OnTouchListener – ממשק המאפשר לאלמנט להגיב לנגיעה במסך בשטח האלמנט

לכל ממשק שכזה יש מתודת CallBack בשם On<X>(), שאותה יש לדרוס כדי לומר לאובייקט כיצד להתנהג –

- למשל, עבור הממשק View.OnClickListener, יש להגדיר את מתודת onClick() שתפקידה לבצע פעולה בתגובה לאירוע לחיצת עכבר בשטח האלמנט.

[5] למתעניינים, זהו מימוש של Design Pattern הנקרא Observer.

לאחר שיצרנו את הממשק ודרסנו את המתודה הרלוונטית, יש לקשר את אובייקט ה-View שלנו (למשל, הכפתור) אל ה-Listener המתאים, באמצעות המתודה setOnClickListener(OnClickListener x).

בסרטון הווידאו ניתן לראות דוגמא לביצוע מתודולוגיה כזו. באחד הניסויים במעבדה תתאמנו על מימוש נוסף של המתודולוגיה.

תפריטים

תפריטים הינם חלק חשוב בממשק המשתמש. בעזרתם, ניתן לראות ולשנות את ההגדרות של האפליקציה, וליצור ממשק פונקציונלי נוח בין המשתמש לאפליקציה.

ישנם סוגים רבים של תפריטים. בתדריך זה נדון בשני סוגים מרכזיים –

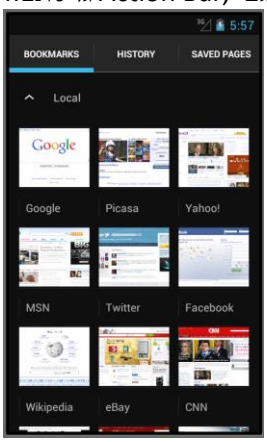
תפריט אופציות (Options Menu), וכן תפריט תכן (Context Menu).

OPTIONS MENU

תפריט האופציות הוא האוסף העיקרי של פריטי תפריט בשביל Activity ספציפי. בתפריט הזה כדאי למקם פריטים שיש להם השפעה גלובלית על האפליקציה – כגון "חיפוש" ו"הגדרות."

עבור גרסת Android 2.3 ומטה, פאנל התפריט מופיע על-ידי לחיצה על כפתור Menu. במכשירים חדשים אין כלל כפתור Menu ולכן בגרסאות Android 3.0 ומעלה הגישה לתפריט היא מה-Action Bar.

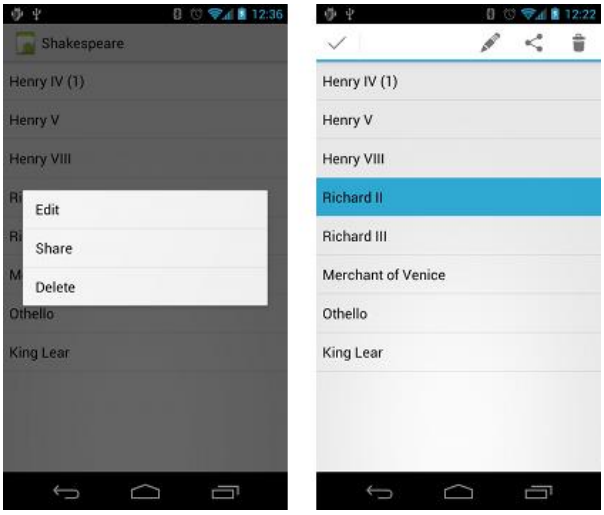
תפריט על גבי Action Bar, גרסת Android 4.1



תפריט Options Menu בגרסת Android 2.3

CONTEXT MENU

Context Menuהינו תפריט צף שמופיע כאשר המשתמש לוחץ על אלמנט כלשהו (GUI Widget, שעליו הוגדר התפריט) לחיצה ארוכה. כדאי שהתפריט יספק פעולות שניתן לייחס לאלמנט שעליו לחצנו.



תפריט Context Menu שצץ בלחיצה ארוכה

הגדרת תפריט

כדי להגדיר תפריט יש ליצור אובייקט שמייצג את התפריט (מופע של ContextMenu / Menu וכן מופעי MenuItemרלוונטיים), ולממש *לכל הפחות* מספר מתודות של המחלקה Activity ב Activity-הרלוונטי. להלן נפרט אותן ומה כל אחת עושה:

- עבור תפריט אופציות: onCreateOptionsMenu(Menu menu) – מתודה זו מאתחלת את תוכן התפריט הראשי של ה-Activity עם התפריט שלנו. כך ניתן להגדיר את הרכיבים שרוצים לכלול בתפריט. התפריט נכנס באופן אוטומטי לעץ ה-Views עם פריטי התפריט.
- עבור תפריט Context המתודה הרלוונטית היא onCreateContextMenu() והיא מקבלת מעט יותר פרמטרים.

הטיפול באירועים ורישום המאזינים לאירועי התפריט נעשה ע"י התפריט עצמו ואין צורך לטפל בו. כאשר נבחר פריט בתפריט, מתבצעת קריאה למתודת ה CallBack-הבאות, בהתאמה:

- onOptionsItemSelected()
- onContextItemSelected()

יש להגדיר מתודות אלו לכל.MenuItem

- עבור תפריט Context יש לקשר את ה-View אל התפריט באמצעות קריאה למתודה registerForContextMenu(View view).

ישנן, כמובן, מתודות נוספות שמאפשרות פעולות נוספות (כגון עדכון התפריט במהלך ריצת ה Activity-ועוד), ועליהן ניתן לקרוא ב.API-

נעיר גם כי קיימת אפשרות להגדיר את רכיבי התפריט ישירות בקובץ ה .XML-ניתן לקרוא על כך באן, אולם בניסוי זה ניצור את התפריטים דרך קוד ה.Java-

תדריך

יצירת אפליקציה ב-ANDROID STUDIO

בניסוי נפתח, נדבר ונריץ את האפליקציות בעזרת תוכנת ה Android Studio-המשמשת את רוב המפתחים שכותבים קוד ל .Android-

למערכת זו יש תמיכה רבה לפיתוח על גבי מערכת Android והיא מספקת פתרונות נוחים .

אין חובה לבצע את התהליך שמפורט וניתן להסתפק בקריאה, אך מומלץ "ללכלך" קצת את הידיים ולהתנסות לבד .

תוכלו להתקין את המערכת בבית או לעבוד על המערכות בחווה או במעבדה. מומלץ להגיע לניסוי כאשר לכל הפחות התנסיתם כבר ביצירת אפליקציה בסיסית מאוד, כפי שנדגים כאן ונפרט אודות השדות השונים.

בתחילת התדריך מופיע קישור להתקנת התוכנה Android Studio

תמונות המסך בתדריך זה לקוחות מגירסת ה Android Studio-הבאה:

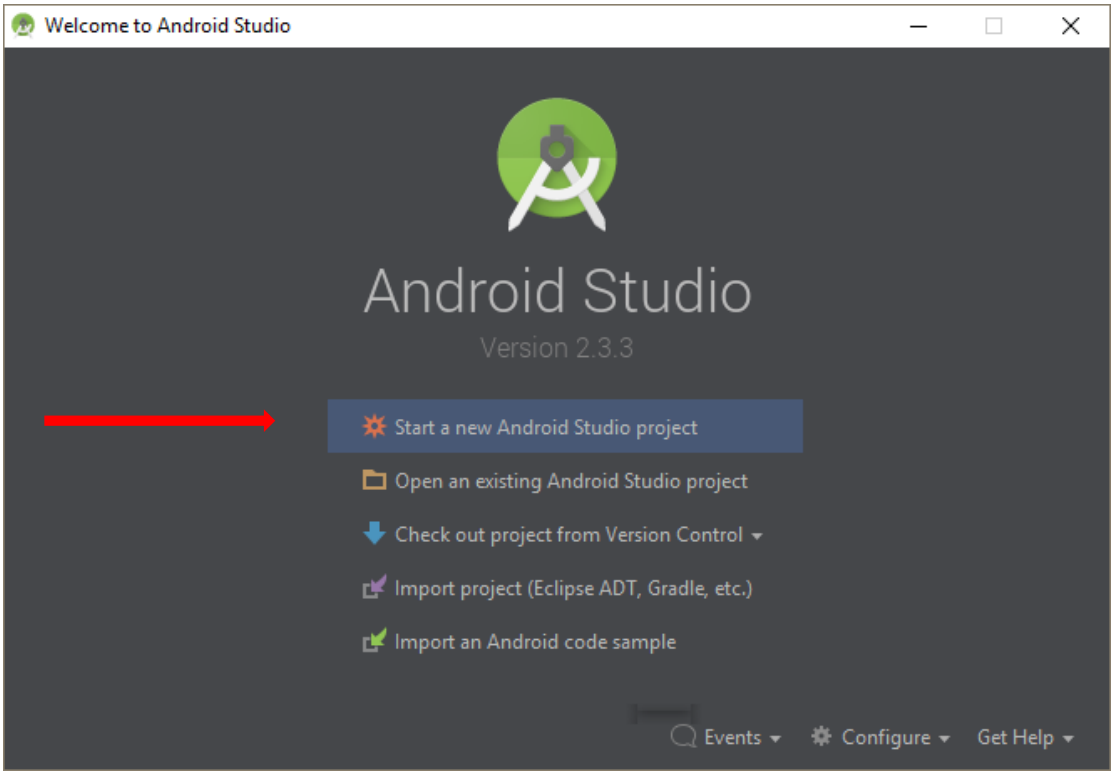
Version: 2.3.3

במידה ואתם משתמשים בגרסאות שונות, עשוי להיות שוני מינימלי בתפריטים.

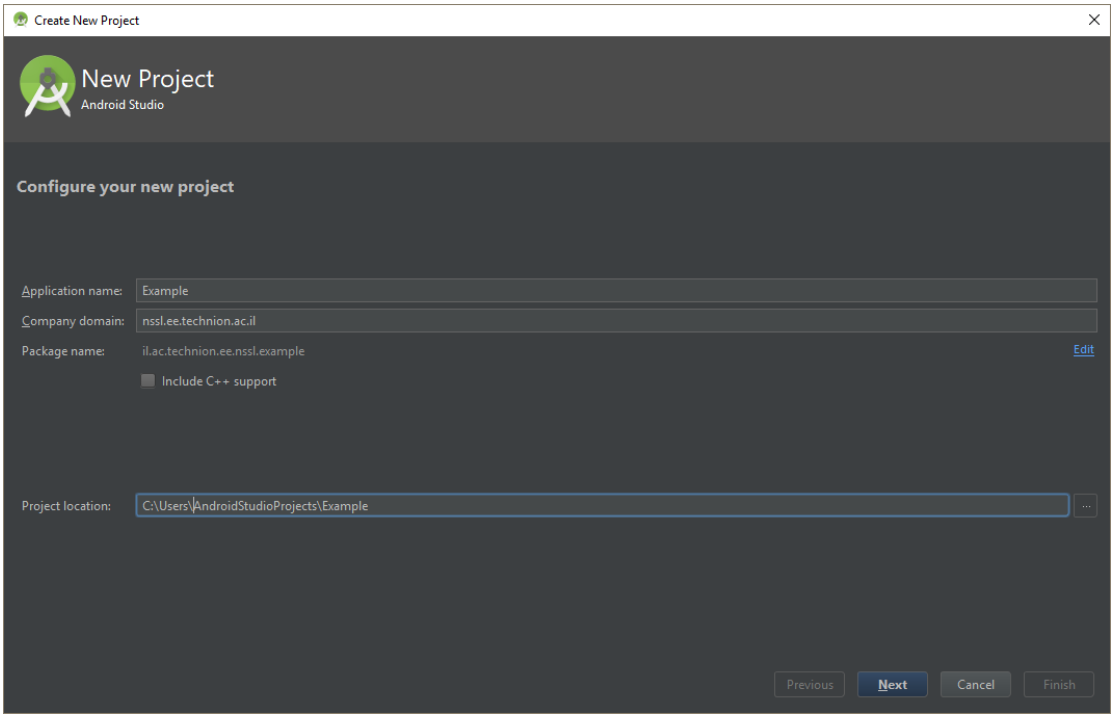
יצירת פרויקט חדש

הפעל את Android Studio ובחר:

Start a new Android Studio project



כעת יופיע החלון הבא:



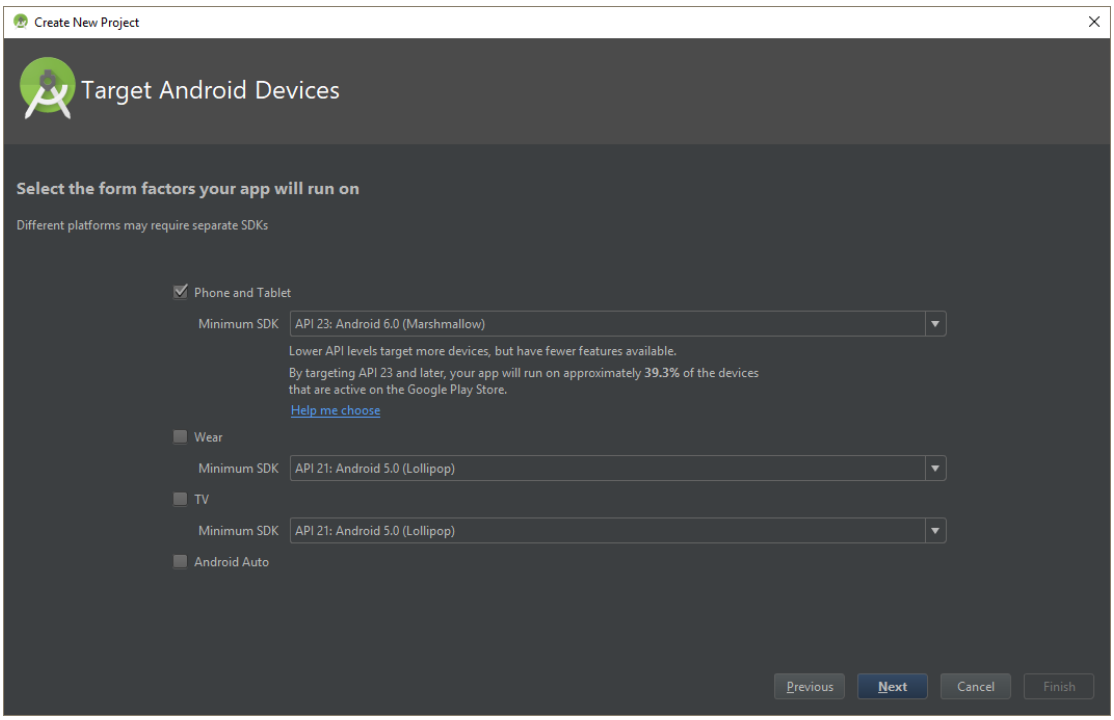
יש לבחור בשם האפליקציה ושם מייצג ; שם הפרויקט במקרה שלנו יהיה זהה ויתעדכן אוטומטית.

שם החבילה (Package Name) הוא מזהה ייחודי לכל אפליקציה. הוא אינו ויזיבילי כלפי המשתמשים, *והוא חייב להישאר זהה לאורך כל זמן החיים של האפליקציה*. דרך שם החבילה ניתן להתייחס למספר גרסאות של אותה אפליקציה בתור "אותה אפליקציה ."

המוסכמה היא ששם החבילה הוא שם ה Domain-של הארגון בצורה הפוכה, בתוספת מזהה אחד או יותר. השם חייב להיות שם תקין של) Java Package ללא מקפים וכו.('

לחצו על הכפתור:Next

תדריך



ה SDK-בו נשתמש הוא API 23 של Android 6.0.

במסך הבא נרצה ליצור Empty Activity וללחוץ על Next.

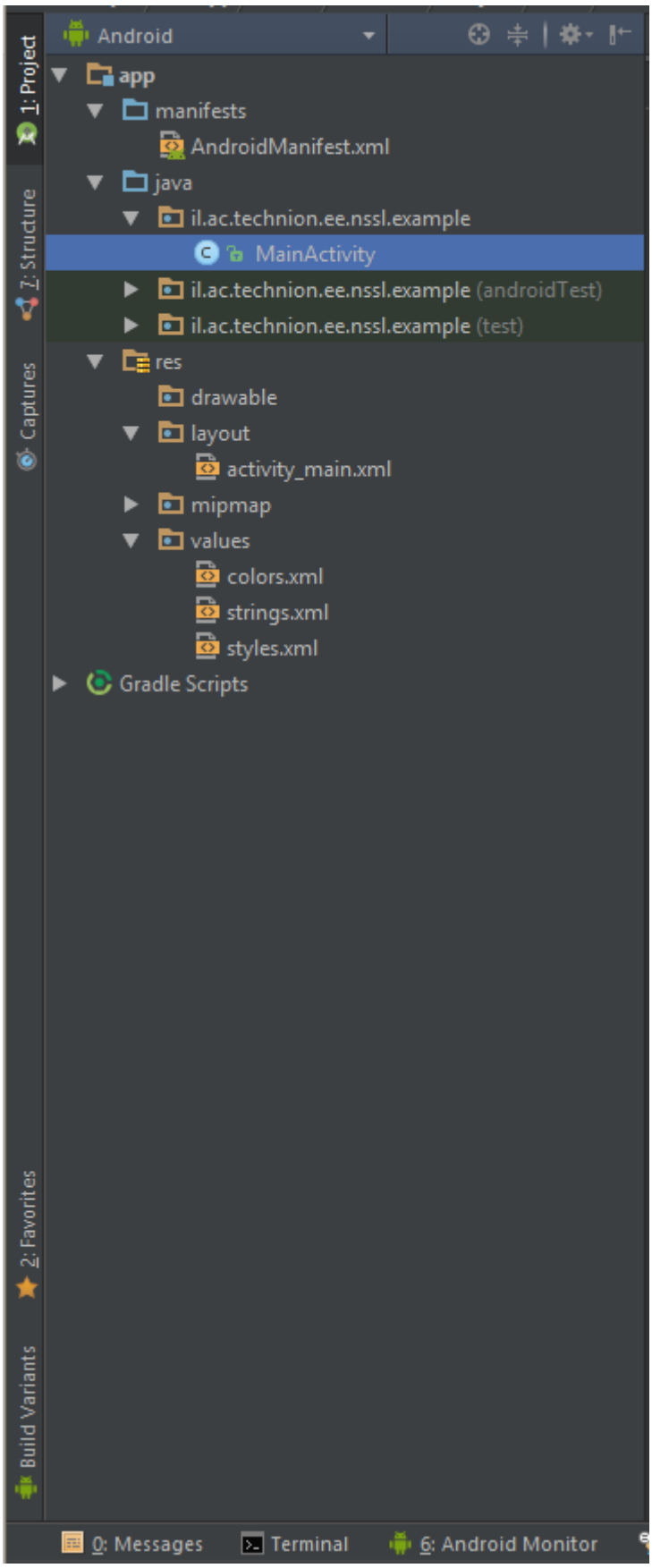
כעת תוכלו לשנות את השם של ה Activity-הראשי וה Layout-הראשי, אך אין צורך בכך. ניתן ללחוץ Finish.

נוצר לנו פרויקט Android חדש שמפעיל Activity ובו כתוב Hello World!

סביבת העבודה

כעת נעשה קצת סדר בדברים שלמדנו עד כה ונראה כיצד הם באים לידי ביטוי בסביבת העבודה:

תדריך



/java:

ספריה עם קבצי קוד המקור (Java) של האפליקציה אותם יש צורך לשנות.

/res:

מלשון .Resources כאן אנו שמים את קבצי המשאבים. המשאבים המרכזיים הם:

- משאבי layout – קבצי ה-XML שמגדירים את ממשק המשתמש. פתיחת קובץ ה-Layout תציג לנו ממשק גרפי נוח לעיבוד, וכן ניתן לגשת ישירות לקוד.
- משאבי values – כגון מחרוזות והגדרות של צבעים וגפנים.

/manifests/AndroidManifest.xml:

זהו קובץ ההצהרה כפי שפורט קודם. הקובץ הזה מפרט את החלקים השונים הקשורים באפליקציה. השינוי הידני העיקרי שנדרש בו הוא ההרשאות.

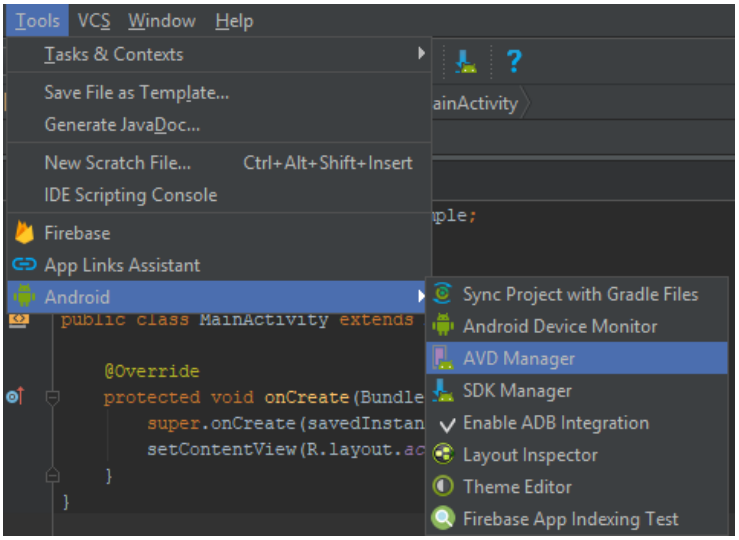
הרצת האפליקציה

בניסוי יסופקו לכם מכשירי Android עליהם תוכלו להריץ את האפליקציה.

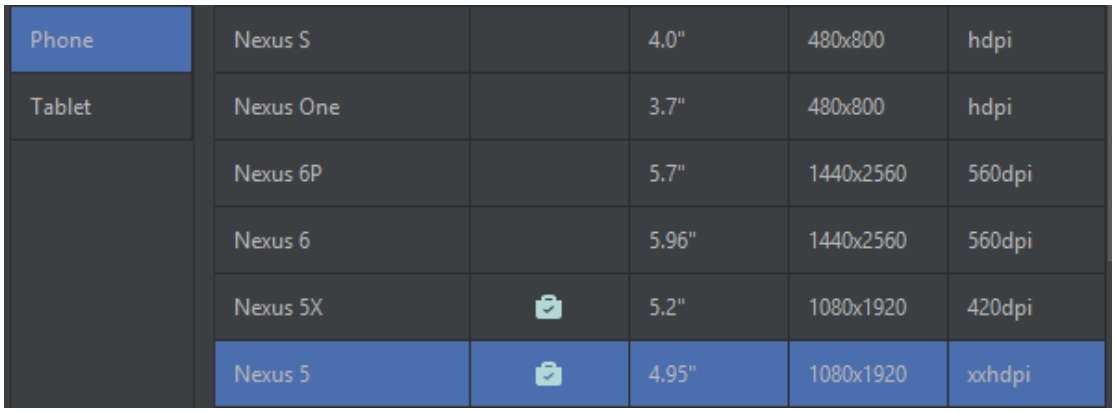
ניתן להשתמש גם בכלי אמולציה ,Emulators – שמדמים את הפעולה של מכשיר ה Android-על גבי המחשב האישי שלכם.

כעת נראה כיצד נגדיר אמולטור ונריץ עליו את התכנית הבסיסית שלנו:

לחצו על תפריט.Tools → Android →AVD[6] Manager



בחלון שעולה, לחצו .Create Virtual Device יופיע החלון הבא:

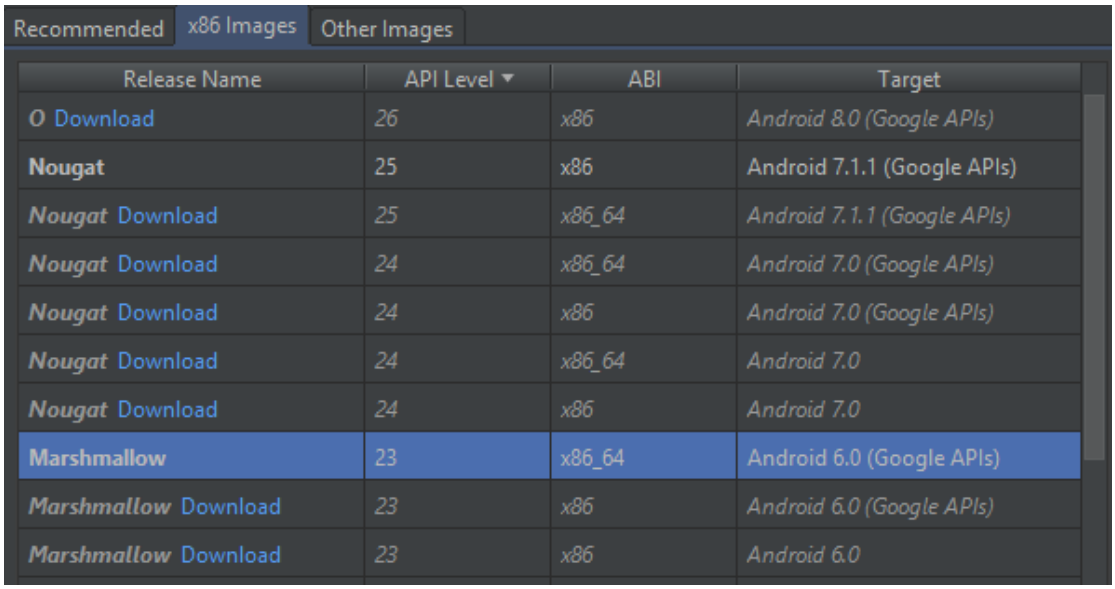


נא לבחור Nexus 5 – Phone ולחצו על הכפתור:Next

[6] AVD – Android Virtual Device

תדריך

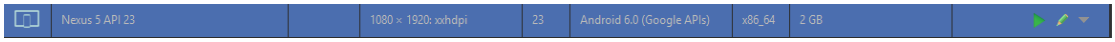
במסך הבא, בחר



לחצו על הכפתור:Next

עדכנו כרצונכם את השם ולחצו.Finish

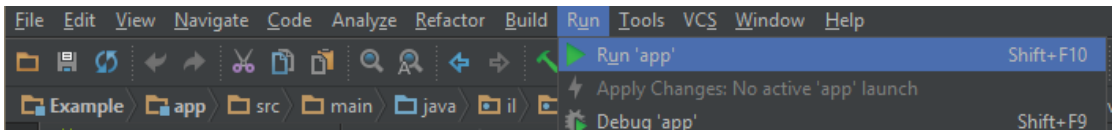
כעת יופיע בפניכם החלון המקורי רק שתוכלו לבחור במכשיר הוירטואלי שיצרתם. לחצו על המכשיר, ולאחר מכן לחצו Start ו .Launch-המתינו עד שהתפריט הראשי של המכשיר עולה.



תהליך אמולציה הוא תהליך כבד. כתלות בחוזק המחשב שלכם, התהליך עשוי להיות איטי למדי.

כעת נוכל לחזור לפרויקט שיצרנו ולהריץ אותו על האמולטור, באופן הבא:

Run → Run app



בתפריט שיעלה תוכלו לבחור את האמולטור שיצרתם, ולאחר ההרצה תוכלו לעבור לחלון האמולטור ולראות את התכנית רצה:

תדריך

מזל טוב! יצרתם תכנית!Android

כעת נבחן מעט את הקוד שנוצר לנו ונראה כיצד האלמנטים שלמדנו עד כה באים בו לידי ביטוי. לאחר מכן, נבצע בו שדרוג קל.

נתחיל בגישה למשאבים השונים, ולאחר מכן נתבונן בקוד.

משאבים

VALUES

אם ניכנס לקובץ Strings.xml נוכל לראות את המחרוזות השונות שמרכיבות את האפליקציה: החל משם האפליקציה."Example"

כפי שניתן לראות, אפשר בקלות לשנות את הערכים הללו מבלי לגשת לקוד. זהו תכנות נכון שמקל בצורה משמעותית על המודולריות וחוסר הצימוד של הקוד.

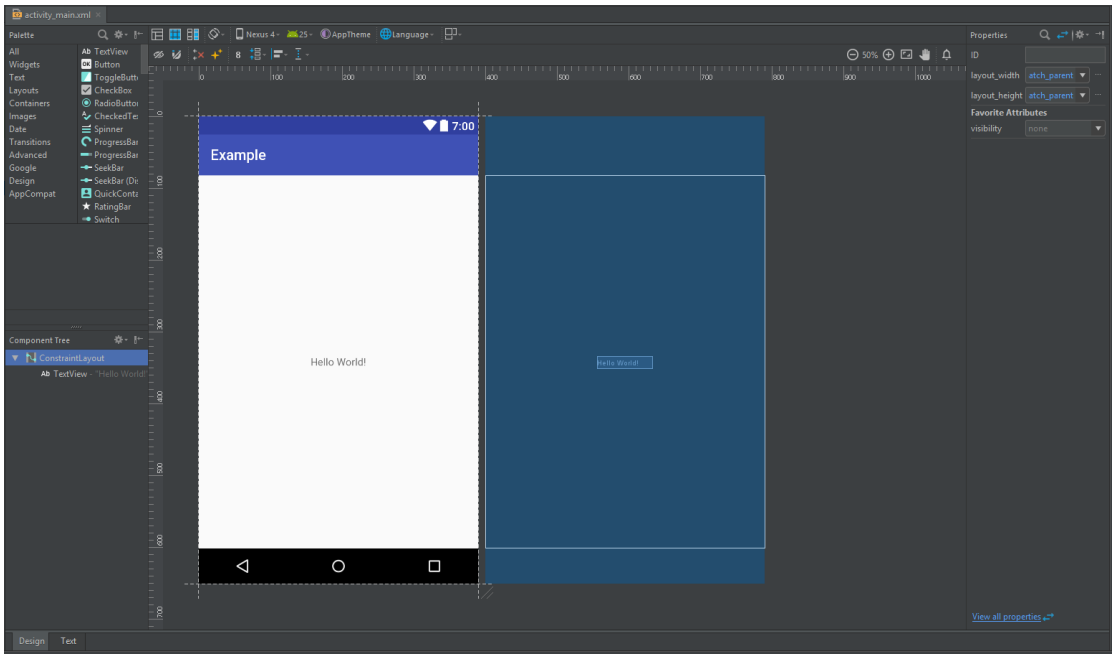
כאשר רוצים להוסיף מחרוזת לממשק, עדיף להוסיף אותה כמשאב בקובץ המשאבים. בכך מרכזים את כל המחרוזות במקום אחד, וקל יותר לעדכן את הטקסט או לספק אלטרנטיבות לשפות שונות.

את המחרוזות שמים בקובץ. res/values/strings.xml

תוכלו לעבור בין הגרסה הטקסטואלית של צפייה בקובץ לגרסה הגראפית (open editor), ולעבוד עם כל אחת – בהתאם לנוחות.

LAYOUT

כעת נעבור לחלון ה Layout-המרכזי ונראה את האפשרויות שהוא מציע:



במרכז מופיע ה Layout-כפי שמתואר.

משמאל למעלה נתן לראות את ה :Palette-מכאן ניתן לבחור אלמנטים גרפיים מהסוג View ו ViewGroup-
)כפי שהוסבר בתחילת התדריך.(

משמאל למטה מופיע ה :Outline-זהו קיבוץ כל ה ViewGroups-שקובעים את סידור ה Views-(ראו דיאגרמה בתדריך). בשלב זה יש לנו ConstraintLayout אחד שמכיל בתוכו אלמנט TextView של תצוגת טקסט.

אם נלחץ (במסך המרכזי) על **הטקסט**, נוכל לראות את השדות והערכים השונים שלו מימין. שני פרטים חשובים הם–

1) Id – זוכרים שאמרנו שלכל אלמנט יש ID ייחודי? שדה זה הוא מאוד חשוב, ועוד על כך בוידאו.
נא להוסיף id (לדוגמה txtHello)

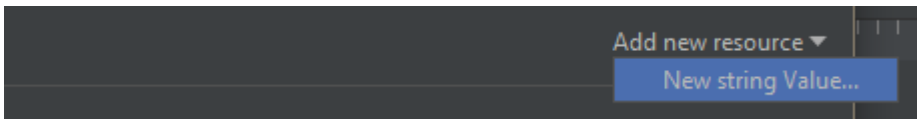
שימו לב שגם לקובץ ה XML-עצמו אפשר להוסיף שדה .ID באמצעות השדה הזה נוכל לגשת אליו בקוד המקור.

הסימן @ דרוש כאשר רוצים להתייחס <u>למשאב</u> כלשהו בקובץ ה-XML והוא מלווה בטיפוס המשאב (במקרה זה : id) ובשמו של המשאב.

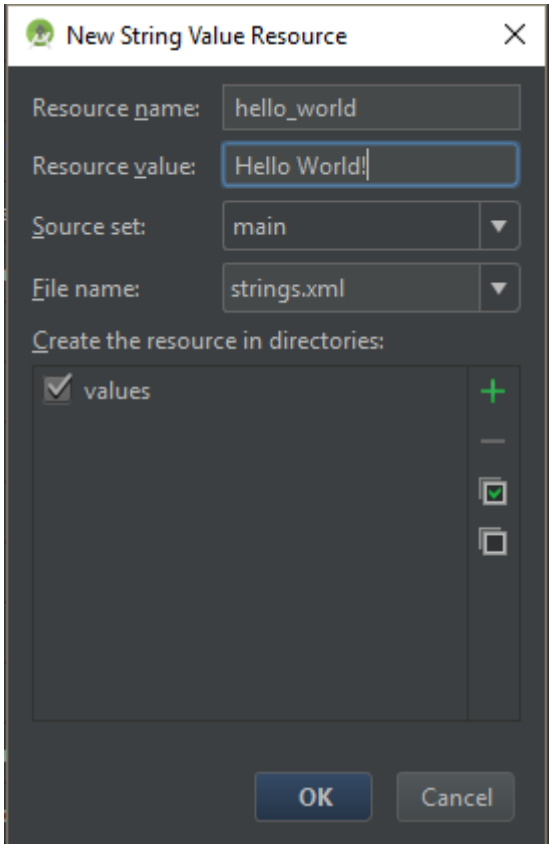
סימן ה-+ דרוש רק כאשר מגדירים את המשאב בפעם הראשונה, זוהי הודעה שיש ליצר אותו.

2) גלילה מעט למטה תראה לנו את השדה Text
נא למחוק את הטקסט Hello World! ולחצו אל שלוש נקודות מימין השדה, בחלון שנפתח לחצו על Add new resource – ו New string Value

תדריך



למלא שדות Resource name וResource Value



לאחר שתלחצו על אישור זה יקשר את ה TextView-למחרוזת, @string/hello_world שנראה גם – ב-Values! שוב, שימו לב למודולריות וחוסר התלות בין הקוד, למחרוזות, ולממשק הגרפי.

כעת נלחץ על ה Layout-מימין למעלה ונשים לב לעוד כמה שדות חשובים:

Width, Height: 1) ניתן להכניס גודל ספציפי בגדלים שונים או ערכים יחסיים כגון fill_parent, match_parent, fill_content. דוגמא לערכים אלו ניתן לראות בוידאו.

תוכלו לעבור בין הגרסה הטקסטואלית של צפיה בקובץ לגרסה הגראפית, ולעבוד עם כל אחת – בהתאם לנוחות.

קוד

כעת עיברו לקובץ הקוד.MainActivity.java

ניתן לראות את המחלקה MainActivity שיורשת ממחלקת האב Activity ודורסת את המתודה החשובה onCreate().ניתן לראות קריאה למתודה של מחלקת האב (חשוב), ולאחר מכן קריאה למתודה setContentView()עם ה ID-של קובץ ה XML Layout-שראינו קודם .activity_main – כך שכאשר התכנית עולה, המסך שמוצג לנו הוא ה Layout-שיצרנו!

שדרוגים ושינויים

כעת נתרגל מעט שינויים קטנים ו'על הדרך' נתייחס לעוד כמה שדות חשובים של משאבים.

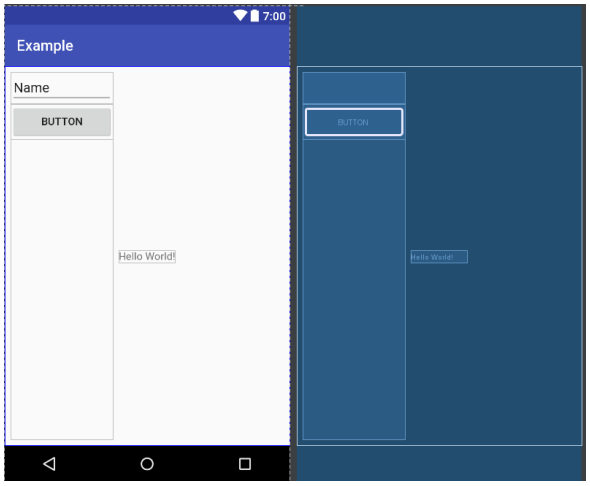
ממשק גרפי ומשאבים

נרצה להוסיף לאפליקציה שלנו שדה טקסט וכפתור.

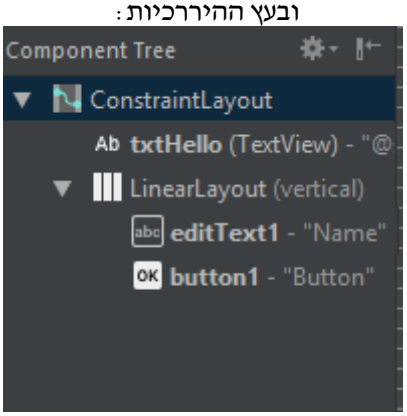
ראשית נוסיף שלוש מחרוזות לקובץ ה (string.xml)..Values-נוסיף את שלוש השורות הצבועות בצהוב ישירות לקובץ ה) XML-ניתן גם לבצע זאת באמצעות הממשק הגרפי), כך שכעת הקובץ נראה כך:

```
<resources>

    <string name="app_name">Example</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="edit_message">Enter a Message</string>
    <string name="button_send">Send!</string>

</resources>
```

נחזור לקובץ ה ,Layout-ונוסיף Linear Layout מתחת ל Layout-הראשי, ובתוכו כפתור (Button) ותיבת טקסט:.(Text Field)

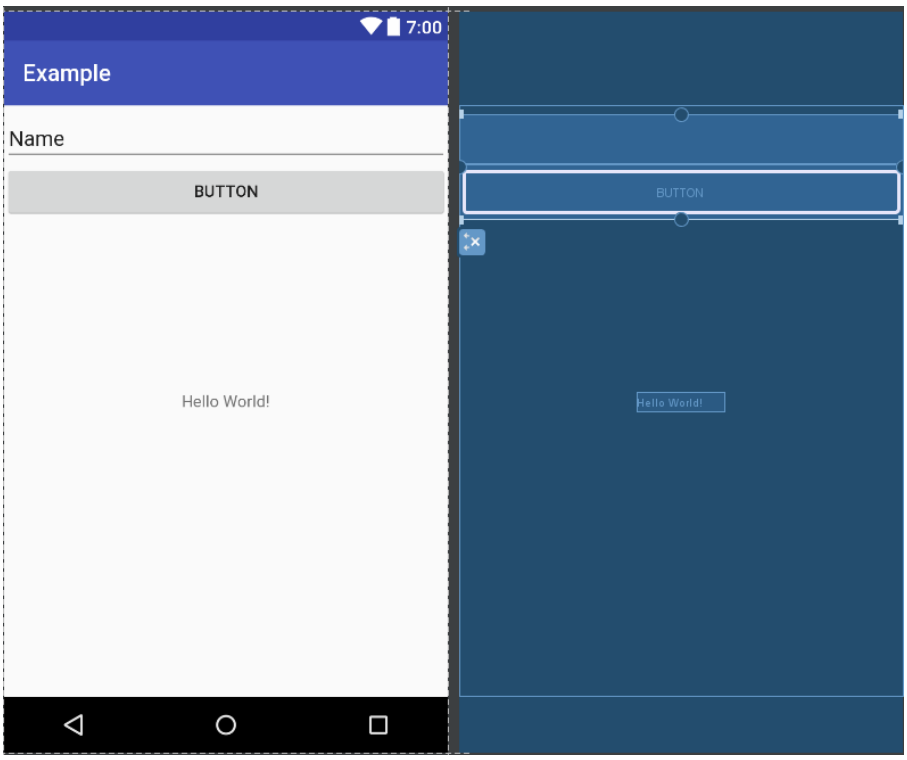


ובעץ ההיררכיות:

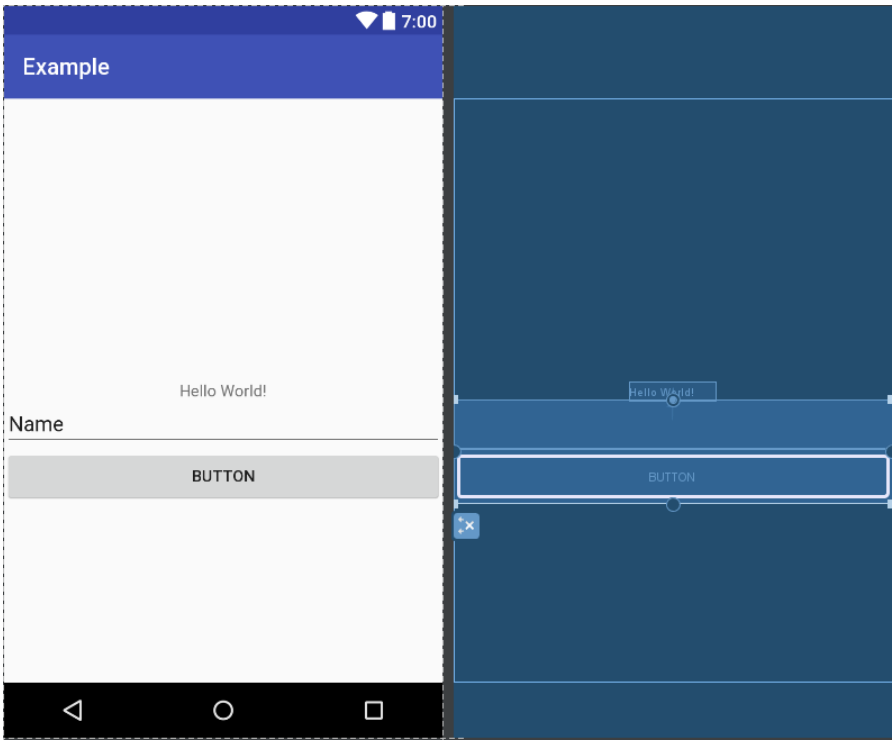


תדריך

כעת ניגש ל Linear Layout-החדש שיצרנו ונשנה את ה Width-שלו ל .fill_parent-שימו לב שכעת שדה הטקסט והכפתור "נמרחים" לאורך כל השורה:

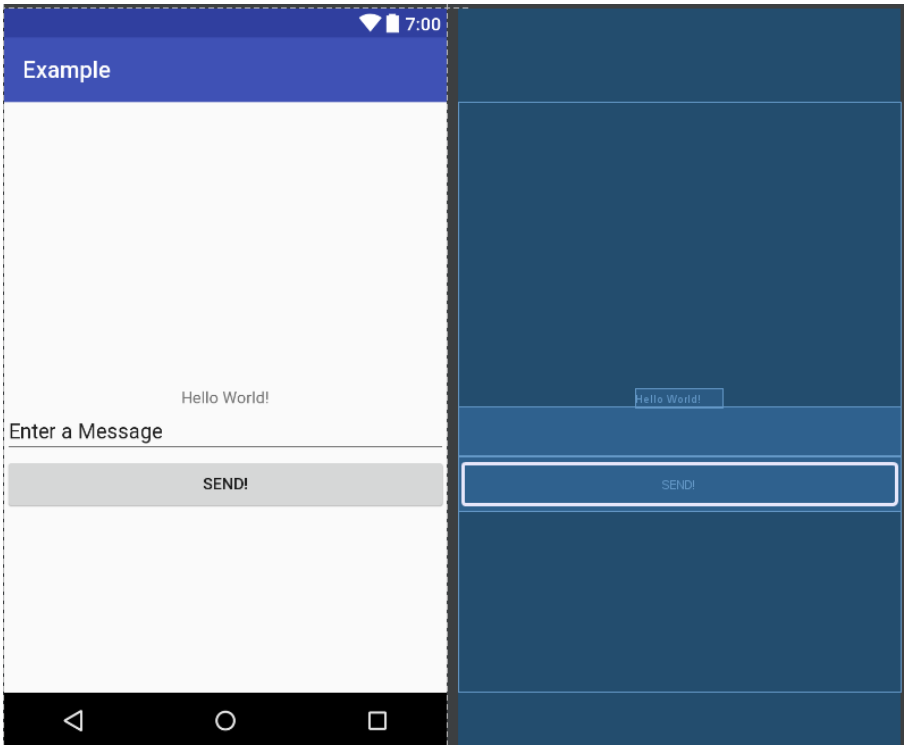


נמקם (גראפית) את ה Layout-מתחת למחרוזת ה "Hello World", TextView-כך שנקבל את התצוגה הגראפית הבאה:



תדריך

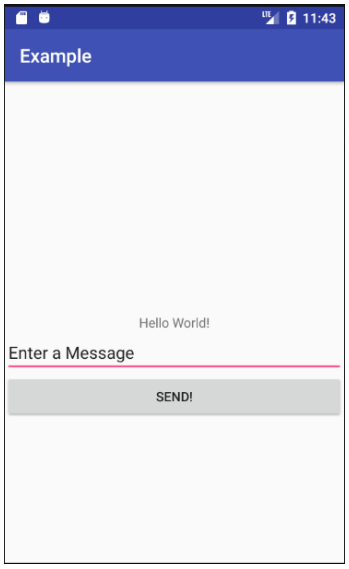
השלב הבא הוא הרצון שלנו לקשר את הכפתור ותיבת הטקסט עם המחרוזות שקבענו קודם. נלחץ על הכפתור, נלך לשדה Text ומשם נבחר את המחרוזת button_send שהגדרנו קודם – ובדומה גם בתיבת הטקסט נבחר את המחרוזת edit_message. כעת נקבל:



שימו לב שגודל תיבת הטקסט והכפתור השתנה דינאמית כאשר שינינו את הטקסט.

תכונה חשובה נוספת שנדגיש כרגע היא תכונת המשקל (layout_weight):

תכונת המשקל מאפשרת לקבוע את שארית המרחב שכל אלמנט View יתפוס, יחסית למרחב שתופסים האלמנטים השכנים שלו. בברירת המחדל, המשקל של כל אלמנט הוא 0. אם ניתן משקל לאלמנט אחד בלבד, הוא ימלא את שארית המרווח אחרי ששאר האלמנטים קיבלו את המרווח שלהם. לכן, נרצה לתת לכפתור משקל 0, ולתיבת הטקסט משקל 1. כעת התוצאה נעימה הרבה יותר:



תדריך

נסכם את קובץ ה-Layout כפי שהוא נראה כרגע:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="il.ac.technion.ee.nssl.example.MainActivity">

    <TextView
        android:id="@+id/txtHello"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintTop_toBottomOf="@+id/txtHello">

        <EditText
            android:id="@+id/editText1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="@string/edit_message"
            android:layout_weight="1"/>

        <Button
            android:id="@+id/button1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/button_send" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

(הרגישו בנוח לבצע שינויים גם דרך הממשק הגרפי וגם ישירות טקסטואלית בקובץ ה-XML – לכל אחד יתרונות משלו).

שדה נוסף שחשוב להכיר הוא השדה android:hint שמכיל את המחרוזת שתופיע בתיבת טקסט, כאשר היא ריקה.

שימו לב שאם תריצו כרגע את התכנית על גבי האמולטור, תוכלו לראות את שדה הטקסט והכפתור. כמובן שכרגע הם לא עושים כלום – ואנו נרצה לשנות זאת!

נרצה, שבעת לחיצת כפתור, נעבור ל-Activity חדש שיציג את הערך שהכנסנו בתיבת הטקסט.

תדריך

נעבור לקובץ ה-Java ונכריז על הכפתור ותיבת הטקסט[7]:

- נצהיר עליהם כמשתני מחלקה private:

```
private Button send_button;
private EditText myText;
```

- נאתחל אותם (=נקשר בינם לבין ה-Views שלהם) במהלך המתודה onCreate(), באמצעות המתודה findViewById():

```
send_button = (Button)findViewById(R.id.button1);
myText = (EditText)findViewById(R.id.editText1);
```

- נגדיר Intent שיעביר את הנתונים מה-Activity הנוכחי ל-Activity החדש.

```
final Intent intent = new Intent(this, DisplayMessageActivity.class);
```

- נגדיר לכפתור פעולת Listener, ובתוכה ניצור מופע חדש של OnClickListener בו:
- נקשר בין ה-Intent ל-Activity החדש (נקרא לו בשם DisplayMessageActivity)
- נוסיף ל-Intent את המידע הדרוש (מחרוזת שניקח משדה הטקסט)
- נאתחל את ה-Intent:

```
send_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String message = myText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
});
```

- ה-Intent מעביר איתו זוגות של מפתח-ערך הנקראים extras. המתודה putExtra() מקבלת כמפתח מחרוזת, ואת הערך כפרמטר שני. יש להגדיר את המפתח כקבוע בתוך MyFirstActivity:

```
public class MainActivity extends AppCompatActivity {
    public final static String EXTRA_MESSAGE =
"il.ac.technion.ee.nssl.example";
…
    }
```

- הקוד הסופי שאמורים לקבל הוא כדלהלן: (שימו לב שיש לבצע Import ללא מעט חבילות!)

[7] ישנן מספר דרכים לקשר ישירות בין הכפתור למתודה בממשק הגרפי, תוכלו למצוא פרטים נוספים באינטרנט.

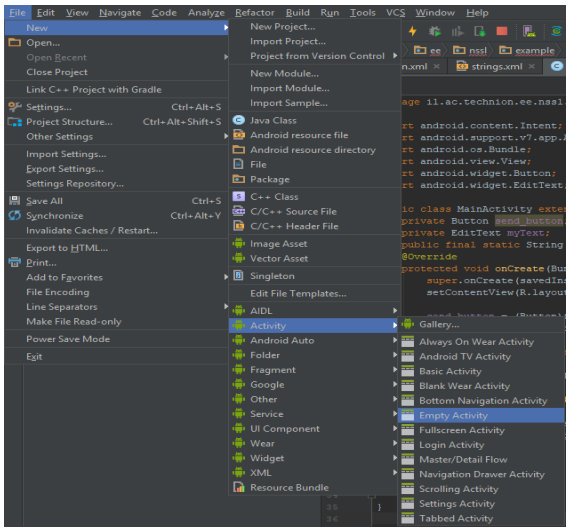
תדריך

```
package il.ac.technion.ee.nssl.example;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    private Button send_button;
    private EditText myText;
    public final static String EXTRA_MESSAGE =
"il.ac.technion.ee.nssl.example";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        send_button = (Button)findViewById(R.id.button1);
        myText = (EditText)findViewById(R.id.editText1);

        final Intent intent = new Intent(this, DisplayMessageActivity.class);

        send_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String message = myText.getText().toString();
                intent.putExtra(EXTRA_MESSAGE, message);
                startActivity(intent);
            }
        });


    }
}
```

אם אתם מעתיקים את הקוד הנ"ל ישירות ל-Android Studio, שימו לב שאתם מבינים כל שורה ושורה.

- כעת נותר ליצור את ה-Activity החדש, באופן הבא מתוך ה-Package:

נבחר File→New→Activity... ומשם ב-Empty Activity:



ניצור את המחלקה DisplayMessageActivity שיורשת מ-Activity.

תדריך

נעדכן ב Activity-את הקוד הבא:

- "נשאב" את ה-Intent שעבר אלינו:

```
Intent intent = getIntent();
String message = intent.getStringExtra("il.ac.technion.ee.nssl.example");
```

- נדרוס את המתודה OnCreate() [אין זה אמור להפתיע אתכם, שכן *כל* Activity דורס מתודה זו!] באופן שבו תציג את המחרוזת על המסך.

הקוד הסופי שאמורים לקבל הוא כדלהלן: (שימו לב שיש לבצע Import ללא מעט חבילות!)

```
package il.ac.technion.ee.nssl.example;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class DisplayMessageActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent intent = getIntent();
        String message =
intent.getStringExtra("il.ac.technion.ee.nssl.example");

        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(message);

        setContentView(textView);
    }
}
```

ובכן..

זהו!

הריצו את האמולטור ובידקו שהכל עובד כמו שצריך!

מזל טוב!

כעת נראה דוגמאות כלליות למספר דברים חשובים נוספים:

החזרת ערך מ-ACTIVITY

שליחת הערך

ניתן לאתחל Activity ולצפות לערך שיחזור כתוצאה. למשל, אפשר לאתחל אפליקציית צילום ולקבל את התמונה שצולמה כערך חוזר.

כדי לבצע זאת, יש להחליף את המתודה startActivity() במתודה startActivityForResult().

למתודה זו נוסף ארגומנט מסוג Integer שהוא "request code" שמזהה את הבקשה.

כאשר מתקבל ה-Intent של התוצאה, המתודה המטפלת מספקת את אותו הקוד כך שהאפליקציה תדע איך לטפל בו. את הערך שחזר שולפים בעזרת מתודת ה-Callback בשם onActivityResult().

דוגמא לקוד שמאתחל Activity חדש עם המתודה startActivityForResult():

```
static final int PICK_CONTACT_REQUEST = 1;  // The request code
…
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK,
Uri.parse("content://contacts"));

pickContactIntent.setType(ContactsContract.CommonDataKinds.Phone.CONTENT_TYPE);
    // Show user only contacts w/ phone numbers
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
```

קבלת התוצאה

כאשר חוזרים מה Activity-המשני, יש לקרוא למתודה onActivityResult() ולספק לה שלושה פרמטרים:

- הקוד שנשלח לזיהוי הבקשה.
- קוד תוצאה שחזר, המציין אם הייתה הצלחה או ביטול תוצאה.
- Intent הנושא עימו את מידע התוצאה.

דוגמא לקוד המקבל תוצאה מה Activity-שאותחל והסתיים:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.
            // Do something with the contact

        }
```

כעת, נראה מספר תרגילי דוגמא הדומים מאוד לתרגילים הראשונים שתראו בניסוי. נפרט את התרגיל ולאחר מכן נצרף קוד דוגמא שתוכלו לעיין בו.

תרגיל דוגמא 1

מטרה :

במשימה זו ניצור אפליקציית.Hello World

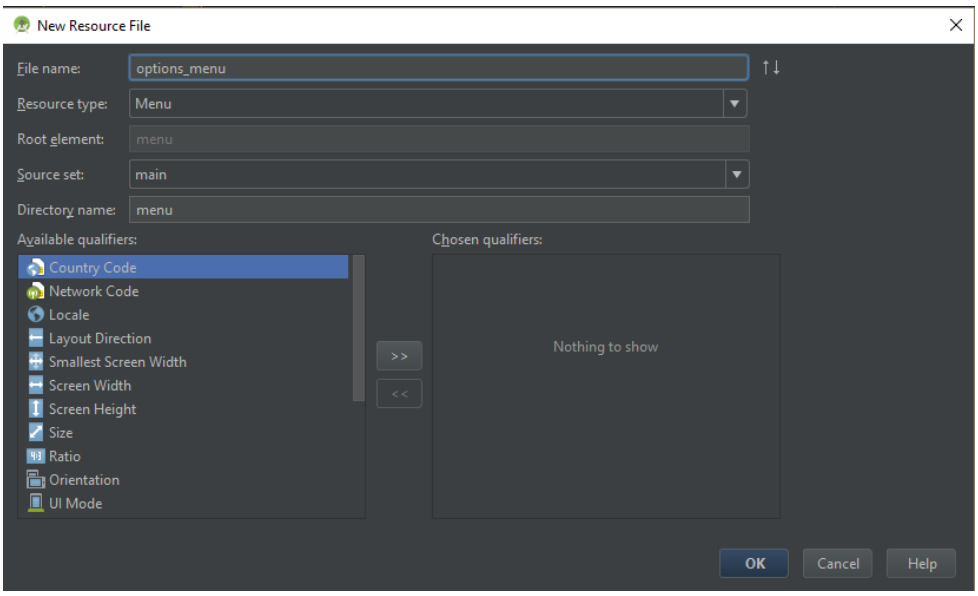
כאשר לוחצים על כפתור ה Menu-ניתן לבחור באופציה "Change String" שתשנה את הכתוב ל-Goodbye World.

שלבים לפתרון:

- יצירת פרויקט בדומה לתרגיל המודרך
- יצירת מחרוזות חדשות "Goodbye World" ו-"Change String" ב-strings.xml
- אין צורך לעדכן את ה-default layout – מלבד לתת ID ל-TextView

בקוד המקור:

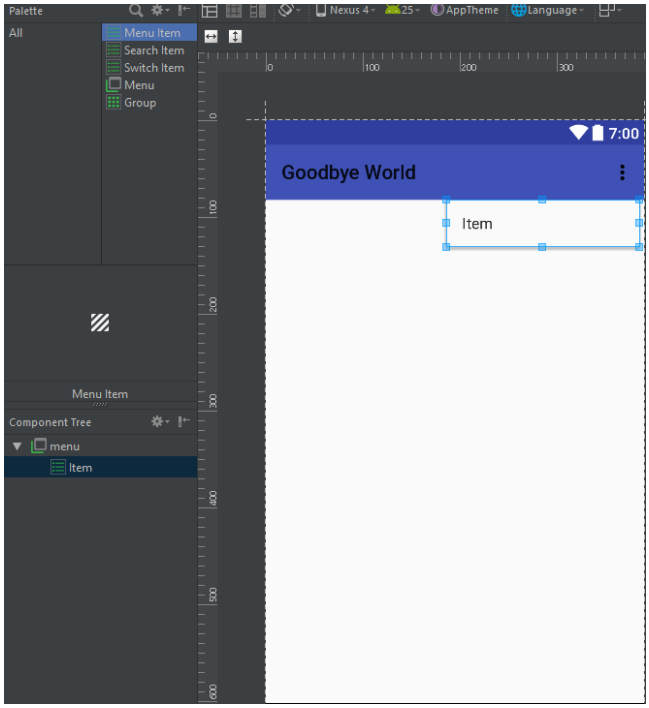
- הגדרת משתנה private מסוג TextView
- במתודה onCreate(), חיבור בין המשתנה ל-View באמצעות ה-ID
- הגדרת משתנה מחלקה סטטי קבוע MENU_CHANGE_STRING עבור הפריט בתפריט.
- הוסיפו את קובץ XML לres, File-New-Android Resource File



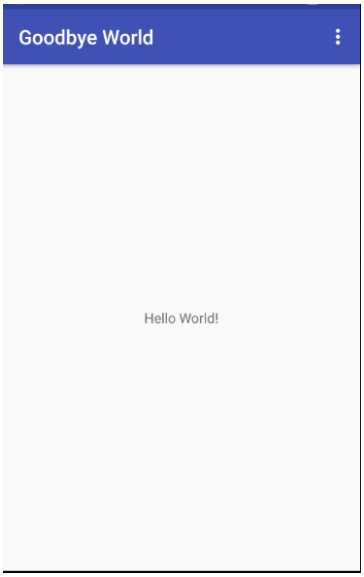
-

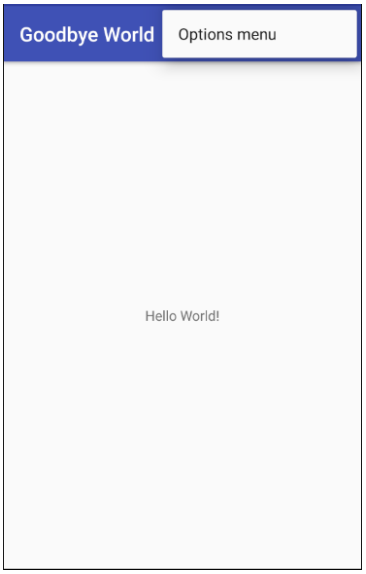
תדריך

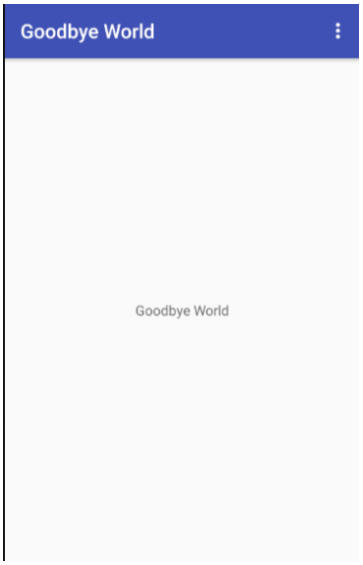
- הוסיפו את Menu item ותגדירו id וTitle



- דריסת המתודה onCreateOption sMenu() והוספת ההתנהגות שמשנה את הטקסט.







להלן התוצאה שמתקבלת :

להלן קוד ה Java-לעיונכם:

```
package il.ac.technion.ee.nssl.goodbyeworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private TextView tv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView)findViewById(R.id.txtHelloWorld);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.options_menu, menu);

        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        switch (item.getItemId()){
            case R.id.options_menu:
                tv.setText(this.getString(R.string.goodbye_world));
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }

    }
}
```

תרגיל דוגמא 2

מטרה:

בתרגיל זה נשתמש באלמנטי טקסט, כפתור, ותיבת טקסט. בעת לחיצה על הכפתור, הטקסט ישנה את הערך שלו לערך שכתוב בתיבת הטקסט.

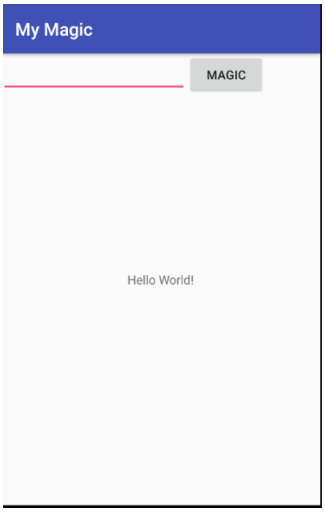
שלבים לפתרון:

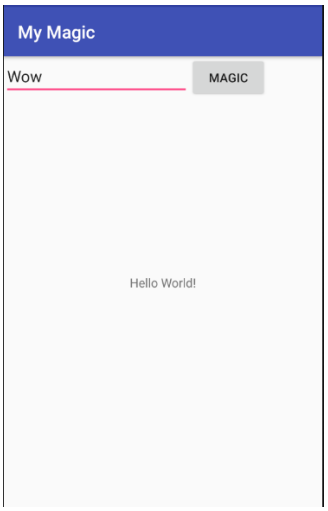
- יצירת מחרוזת רלוונטית
- הגדרת קובץ Layout XML עם TextView, EditText ו-Button מתאימים ומתן ID הולם. חיבור בין המחרוזת שניתנה לשם הכפתור, לבין הכפתור.

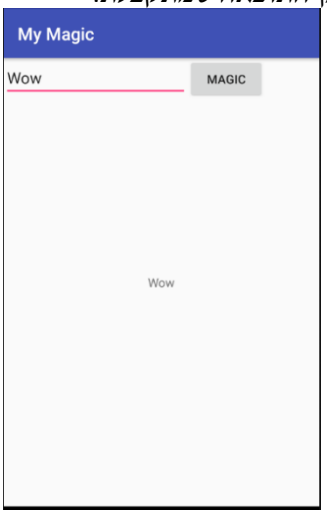
בקוד המקור:

- הגדרת משתני private מסוג TextView, EditText ו-Button.
- במתודה onCreate(), חיבור בין המשתנים לאלמנטי View באמצעות ה-ID
- הוספת מאזין לכפתור באמצעות SetOnClickListener() ויצירת OnClickListener חדש ובתוכו כתיבת המתודה onClick ().

להלן התוצאה שמתקבלת:







קובץ ה XML-של ה:Layout-

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="il.ac.technion.ee.nssl.mymagic.MainActivity">

    <TextView
        android:id="@+id/viewShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

תדריך

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="magic"
        app:layout_constraintLeft_toRightOf="@+id/editText"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

קוד המקור:

```
package il.ac.technion.ee.nssl.mymagic;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private Button button;
    private EditText editText;
    private TextView textView;

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

        button = (Button)findViewById(R.id.button);
        editText = (EditText)findViewById(R.id.editText);
        textView = (TextView)findViewById(R.id.viewShow);

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView.setText(editText.getText());
            }
        });

    }
}
```

טיפים קטנים ושימושיים

Android Studioהיא תוכנה מצויינת שנועדה למפתחים עצלנים – נצלו את יכולותיה:

- כאשר מילה מסומנת באדום, שימו עליה את הסמן. במרבית המקרים ה-Android Studio תציע לכם לעשות import ל-package הרלוונטי והבעיה תיפתר(Alt-Enter).
- שימו לב לסיים הצהרה של אובייקט חדש בנקודה-פסיק (מודגש בצהוב בדוגמא האחרונה) – את זה ה-Android Studio **לא** מוסיף באופן אוטומטי.
- לחיצה על Ctrl+Space משלימה אוטומטית מילים ומציעה לכם את האפשרויות השונות. להלן דוגמא:

תדריך

בדוגמא זו כתבנו "Button.setO" לפני שלחצנו Ctrl+Space, וה-Android Studio הציע לנו את האפשרויות שעומדות בפנינו.
https://developer.android.com/studio/intro/keyboard-shortcuts.html

דו"ח מכין – שאלות להגשה

ענה בקצרה על השאלות הבאות (לתשומת לבך, התשובות לחלק מהשאלות אינן מצוינות במפורש בדו"ח – תוכלו למצוא את המידע הרלוונטי בווידאו ובאמצעות חיפוש בגוגל:)

1. הסבר מה תפקידה של המתודה setContentView(). הדגש מה חשיבותה ונחיצותה.
2. הסבר כיצד מעבירים פרמטרים ל- Activity.
3. הסבר כיצד מוסיפים לאפליקציה הרשאת גישה לאינטרנט.
4. הסבר מה עושה המתודה setOnclickListener().
5. מה ההבדל בין Service ל-Activity?
6. הסבר מהם סוגי התפריטים הקיימים (שניים עיקריים) ומה ההבדל ביניהם.
7. מה תפקידו של קובץ ה-R וכיצד הוא נוצר?
8. הסבר כיצד משנים את שם התוכנית.
9. הסבר כיצד מוגדר ממשק המשתמש בתוך ה–Activity.
10. הסבר איך קבצי המשאבים תורמים להנדסת תוכנה נכונה.
11. תאר את מחזור החיים של ה-Activity, שרטט דיאגרמת מצבים המתארת באילו מצבים הוא יכול להימצא ואת המעברים בין המצבים.
12. הסבר אודות אובייקט מסוג Intent, מה תפקידו, אילו סוגי Intent קיימים ומה תפקידם.
13. ציין מה מציינים ראשי התיבות הבאים <u>בהקשר של תכנות Android</u>:
    - ADT
    - APK
    - API
    - AVD