

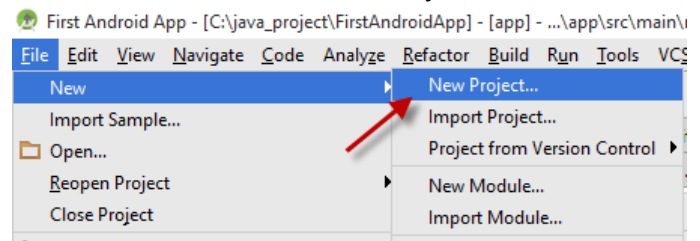
תרגיל בסיסי באנדרואיד

תוכן עניינים

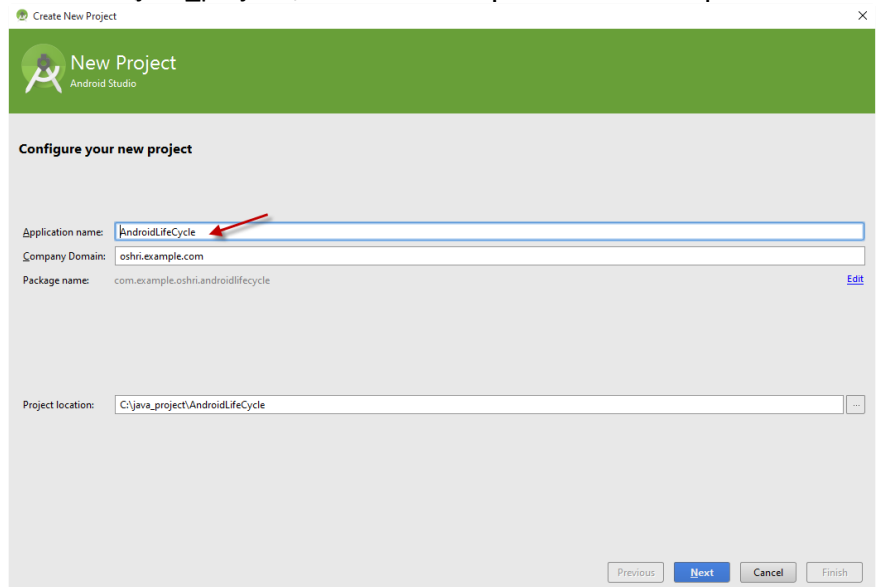
2	יצירת פרויקט אנדרואיד
6	הוספת קוד לבדיקת מחזור החיים
8	בדיקת מחזור החיים ב LogCat
8	הרצת הפרויקט
9	הכרת ה LogCat
11	אמולטור (AVD)

יצירת פרויקט אנדרואיד

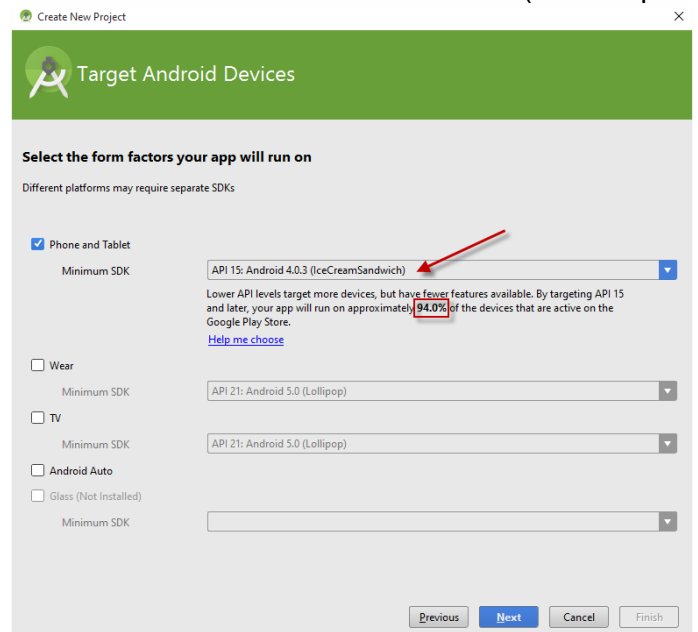
בחר בתפריט File -> New -> New Project...



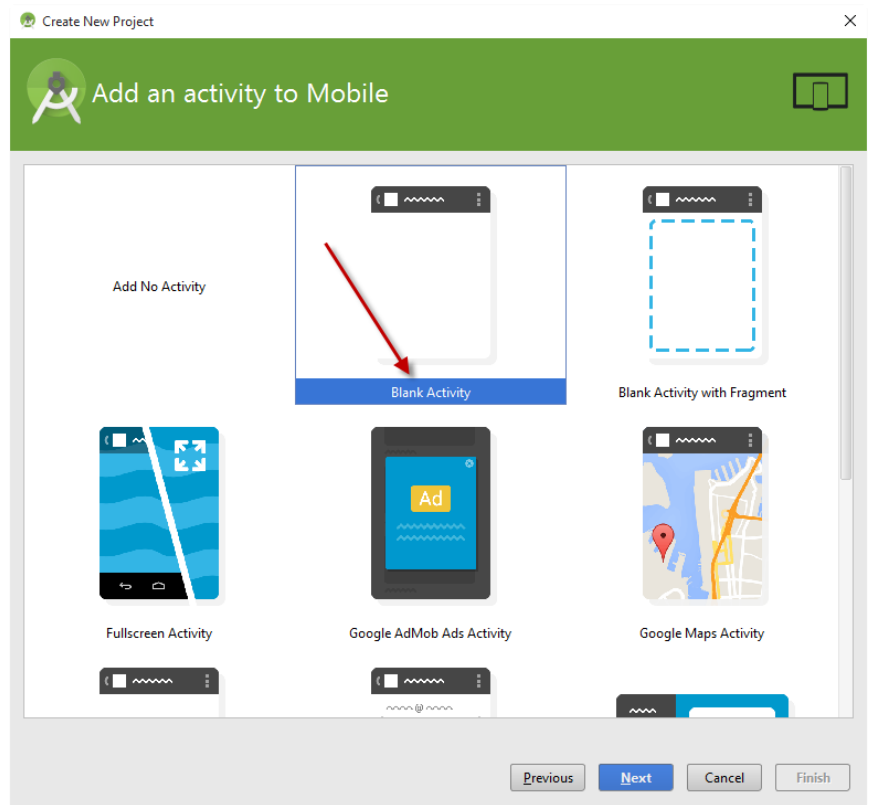
בחלון שייפתח נספק לאפליקציה את השם AndroidLifeCycle ובחר מקום לשמירת הפרויקט כדאי שיהיה מקום פשוט כמו שניתן לראות בתמונה "C:\java_project\"



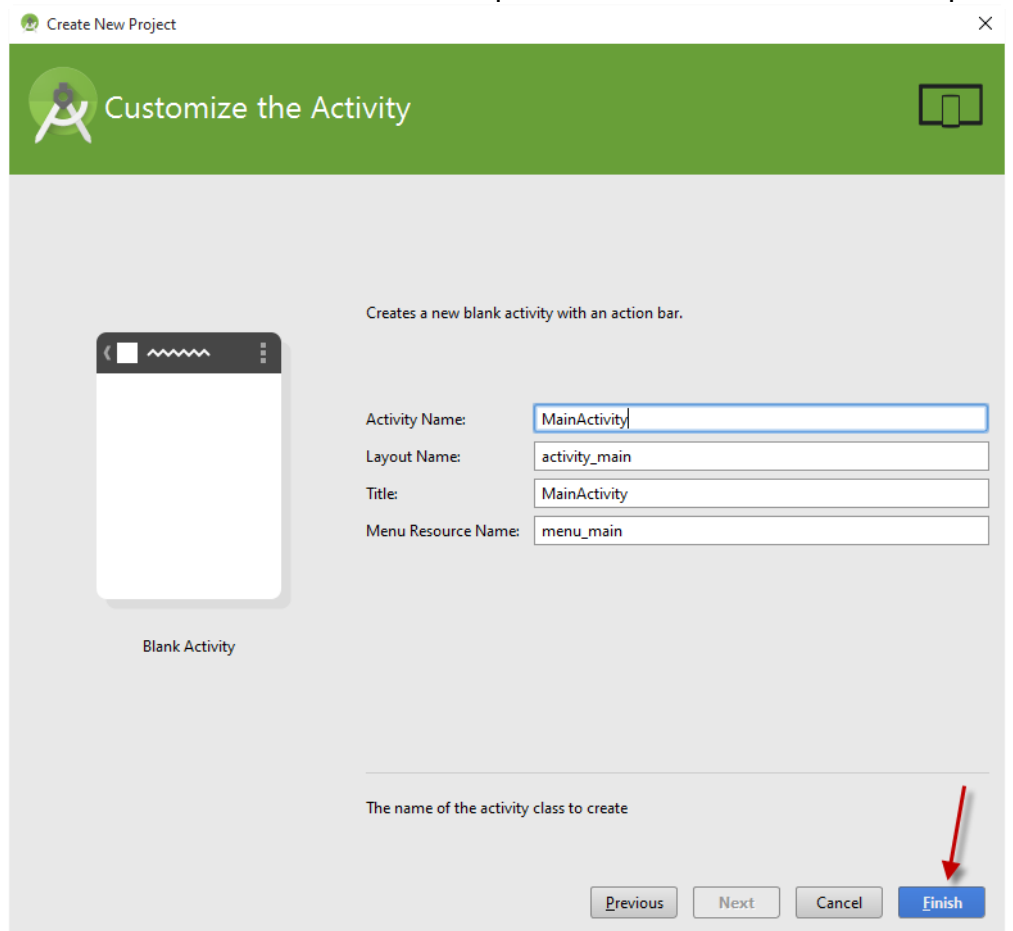
במסך הבא נוודא שמסומנת האפשרות לפיתוח עבור טלפון (Phone and Tablet) ובחר ב Minimum SDK את API 15 (ניתן לראות שגרסה זו מתאימה ל 94 אחוז מהמכשירים – כלומר ל 94 אחוז מהמכשירים יש לפחות גרסה זו של אנדרואיד או גרסה מתקדמת יותר)



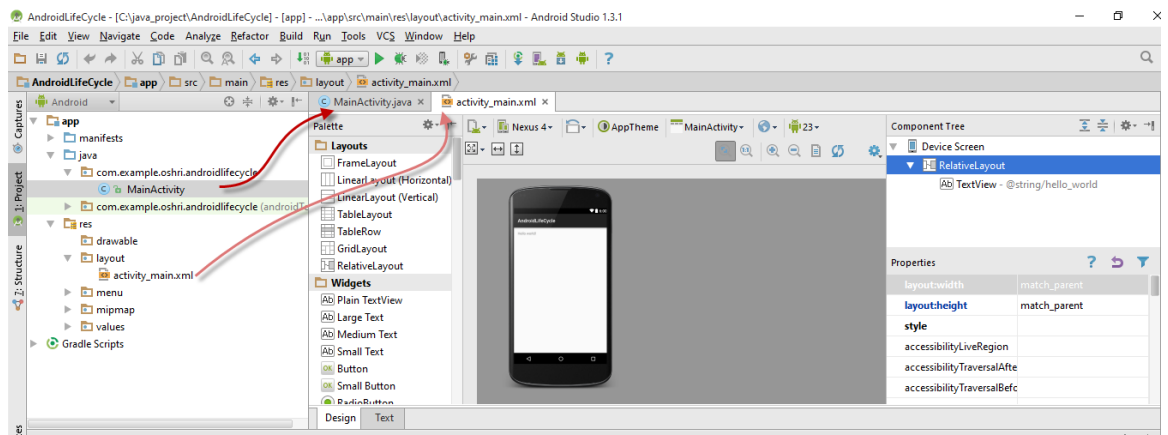
במסך הבא נבחר ב Blank Activity



במסך הבא נשאיר את השמות כפי שהם ונלחץ על Finish



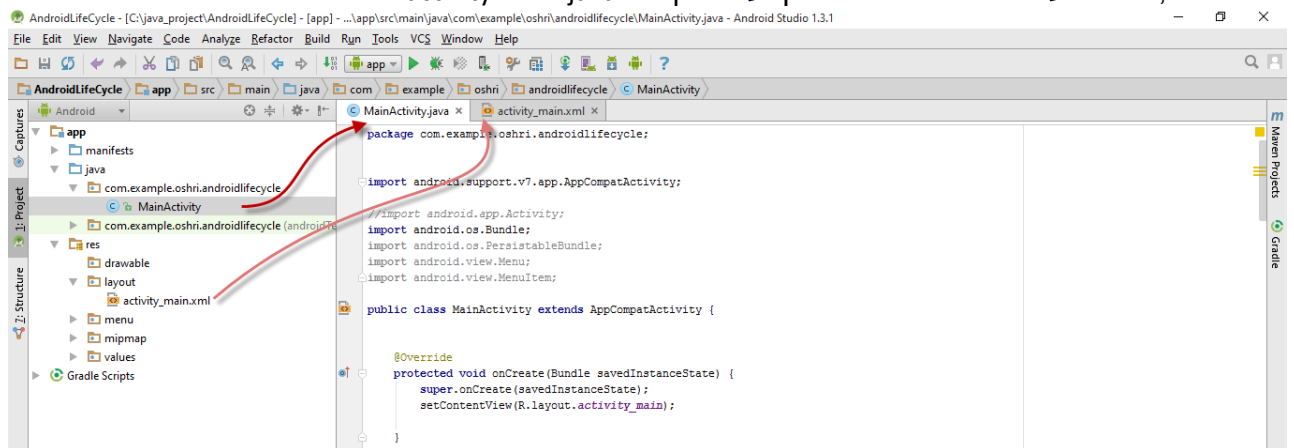
בסופו של דבר ייפתח החלון של Android studio עם פרויקט האנדרואיד שהגדרנו בתוכו כך:



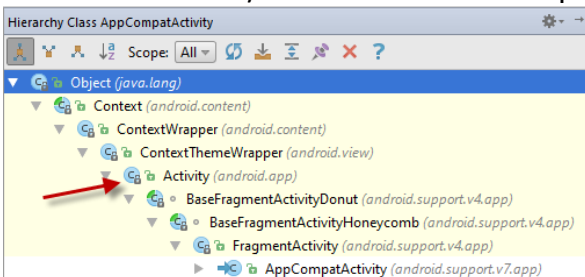
הסבר:

activity_main.xml זה מסמך שמתאר את התצוגה (חלק ה view ב MVC) המחלקה MainActivity זו מחלקה שמתארת את הקוד של המסך (חלק ה control ב MVC) בדרך כלל מסך באפליקציה תהיה תצוגה ב XML ומחלקה של Activity ב java

בתרגיל זה, פחות מעניינת אותנו התצוגה ולכן נעבור לקוד ה java של ה activity



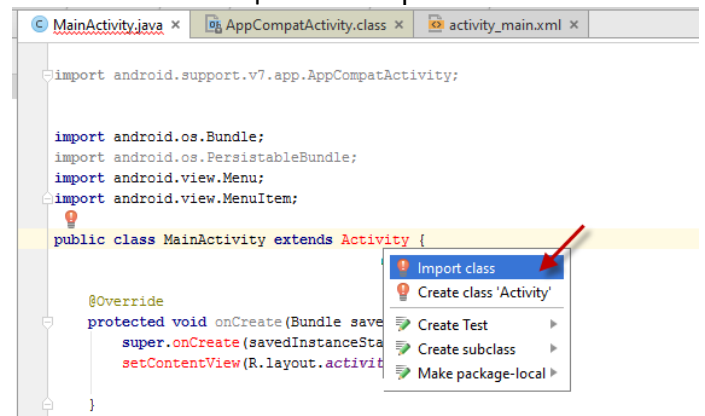
ניתן לראות שהמחלקה MainActivity יורשת ממחלקה AppCompatActivity כמובן שעל מנת שיהיה מדובר ב Activity אחד מהאבות שלה צריך להיות Activity

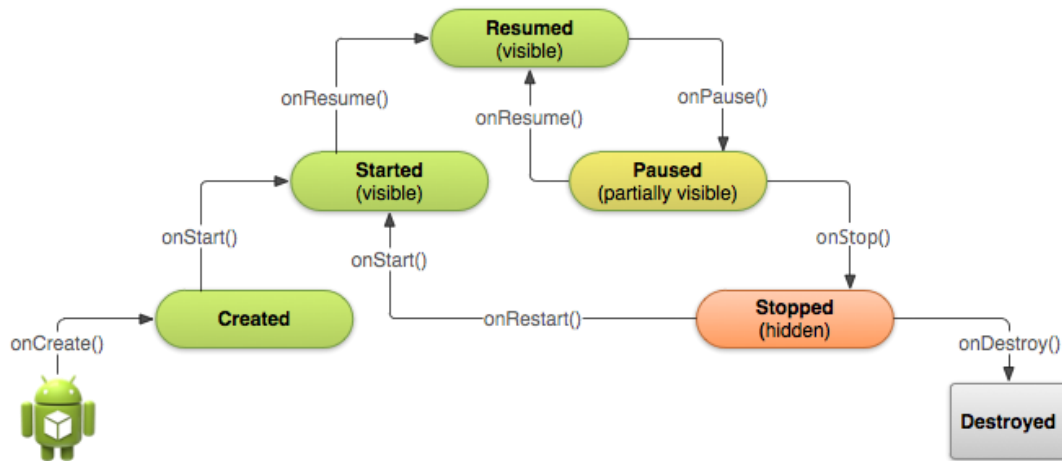


על מנת לפשט את העניין נשנה את הירושה שמוגדרת ונירש ישירות מ Activity

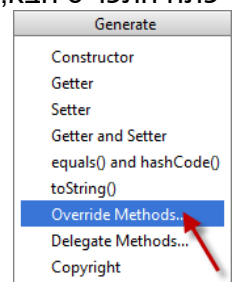
```
public class MainActivity extends Activity {
```

לאחר השינוי נקבל שגיאה כיוון שחסר לנו import של android.app.Activity ולכן נלחץ על המילה Activity שמסומנת באדום (השגיאה שלנו) ואחרי זה נלחץ על Alt + Enter ונקבל אפשרויות לפתרון הבעיה האפשרות הראשונה מספקת את ה import החסר.

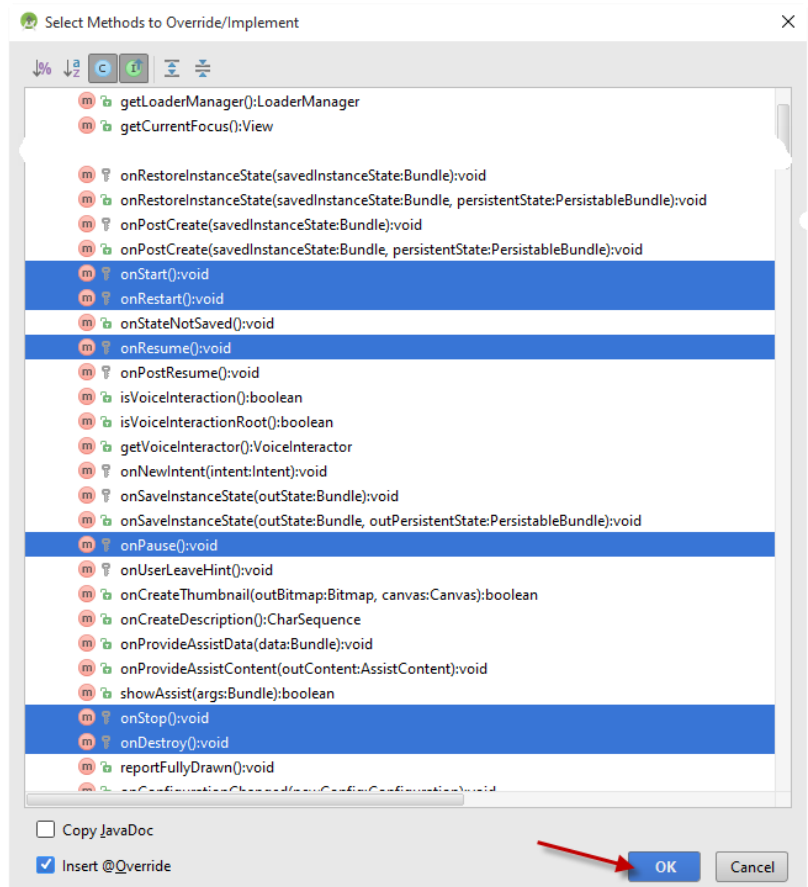
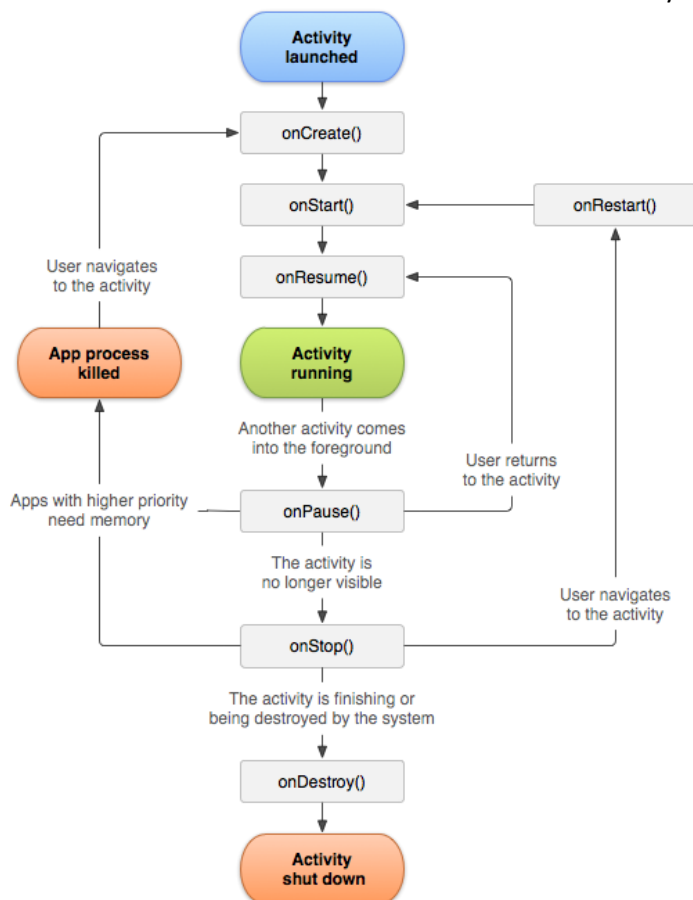




כשהסמן בתוך המחלקה נלחץ על Alt + Insert (או מקש ימני ובחירה ב Generate)
יפתח התפריט הבא, ובו נבחר באפשרות Override Methods...



בחלונית שתיפתח נסמן את כל הפונקציות שמופעלות במחזור החיים של ה Activity



שימו לב, יש לסמן סה"כ 6 פונקציות (onCreate כבר נמצאת)

לאחר אישור (ok) נראה שכל הפונקציות שסימנו התווספו לקוד:

(תמונת מסך מאקליפס – אבל הרעיון זהה !!!)

```
*MainActivity.java
13 public class MainActivity extends Activity {
14
15     @Override
16     protected void onStart() {
17         // TODO Auto-generated method stub
18         super.onStart();
19     }
20
21     @Override
22     protected void onRestart() {
23         // TODO Auto-generated method stub
24         super.onRestart();
25     }
26
27     @Override
28     protected void onResume() {
29         // TODO Auto-generated method stub
30         super.onResume();
31     }
32
33     @Override
34     protected void onPause() {
35         // TODO Auto-generated method stub
36         super.onPause();
37     }
38
39     @Override
40     protected void onStop() {
41         // TODO Auto-generated method stub
42         super.onStop();
43     }
44
45     @Override
46     protected void onDestroy() {
47         // TODO Auto-generated method stub
48         super.onDestroy();
49     }
50
51     @Override
52     protected void onCreate(Bundle savedInstanceState) {
53         super.onCreate(savedInstanceState);
54         setContentView(R.layout.activity_main);
55     }
}
```

```
final String ACTIVITY_LIFE_TAG = "activity lifecycle";
```

ועבור כל פונקציה נוסיף את המימוש:

```
android.util.Log.d(ACTIVITY_LIFE_TAG , "שם הפונקציה");
```

לדוגמה עבור onStart()

```
@Override
protected void onStart() {
    super.onStart();
    android.util.Log.d(ACTIVITY_LIFE_TAG , "onStart()");
}
```

שימו לב לבצע זאת עבור כל הפונקציות, שימו לב במיוחד לא לשכוח לשנות כל פעם את התוכן בהתאם.

הסבר:

Log זוהי מחלקה שנמצאת ב package שנקרא android.util (באמת, היה ניתן לבצע פשוט import) המחלקה הזו מאפשרת דיווח הודעות מהאפליקציה למערכת הפיתוח.


הפונקציה d מדווחת הודעות debug

היא מוגדרת כך: d(String tag, String msg)

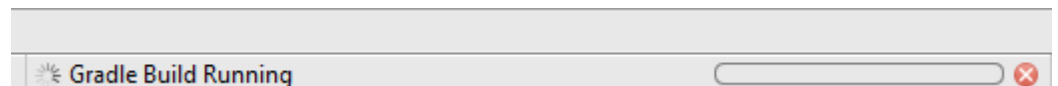
הפונקציה מקבלת:

tag – שבהמשך נראה שניתן לסנן באמצעותו ולקבל במערכת הפיתוח רק הודעות שמכילות tag זה.
msg – ההודעה שתוצג על המסך (במקרה שלנו זה יהיה שם הפונקציה).

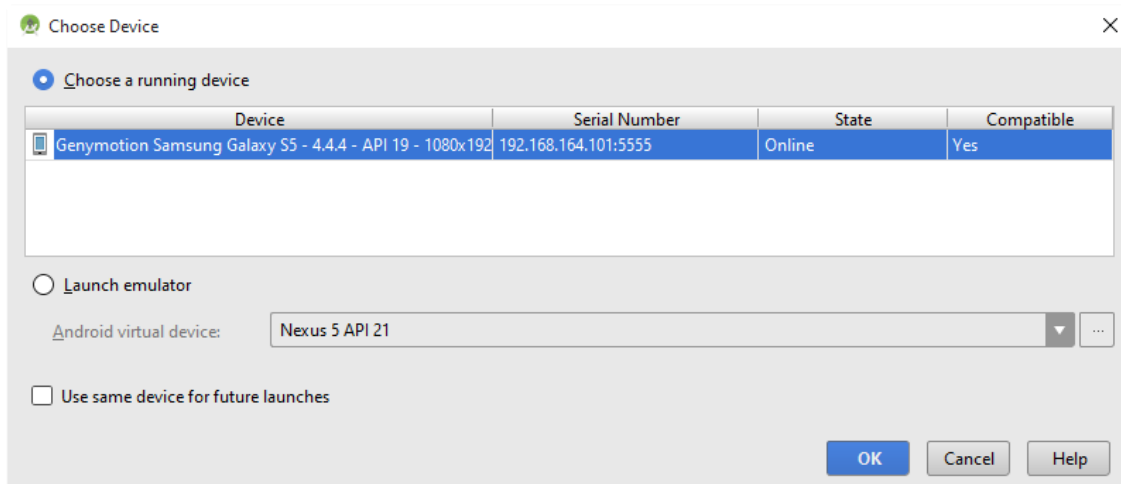
הרצת הפרויקט

נריץ את הפרויקט ע"י לחיצה על 

למטה נראה את התהליך של הבנייה והקמפול

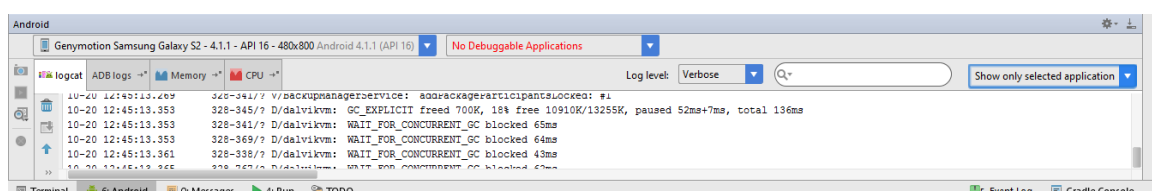


לאחר מכן, נתבקש לבחור מכשיר שעליו האפליקציה תרוץ



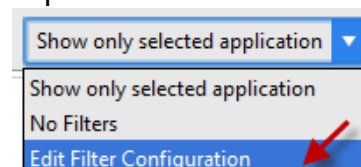
הכרת ה LogCat

logCat זה הרכיב שאחראי להדפסת הודעות שמתקבלות מהאמולטור למערכת הפיתוח

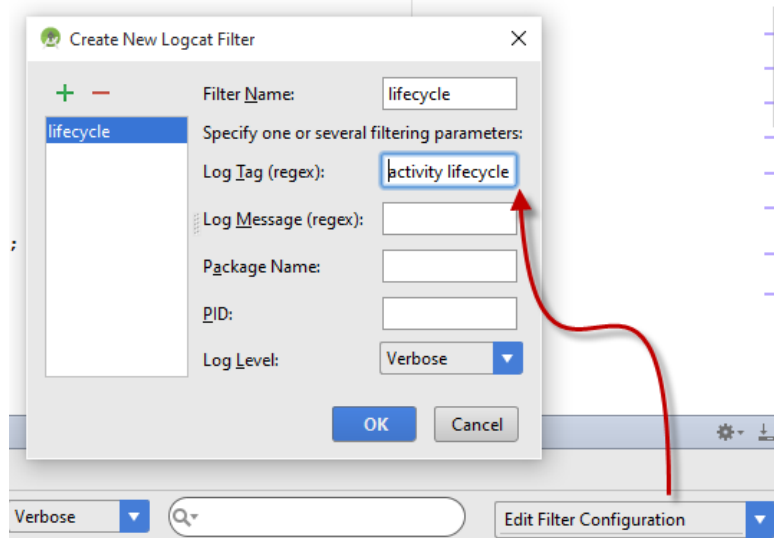


בקיצור, כאן יופיעו ההודעות שלנו שכתבנו בקוד

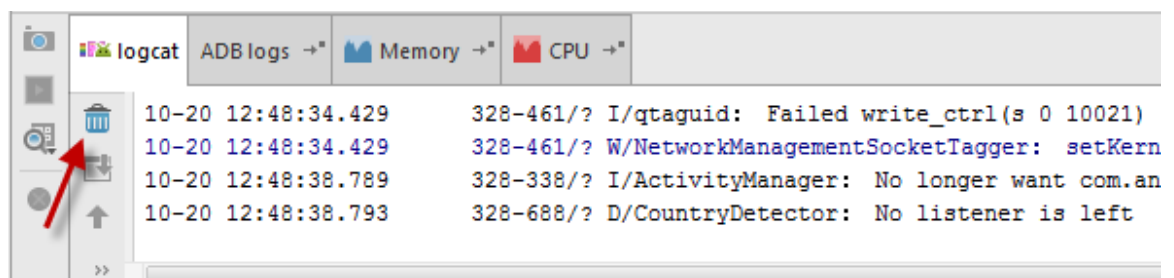
על מנת שיהיה לנו קל לזהות דווקא את ההודעות שלנו.
נבחר בתיבת בחירה בצד ימין ב Edit Filter Configuration



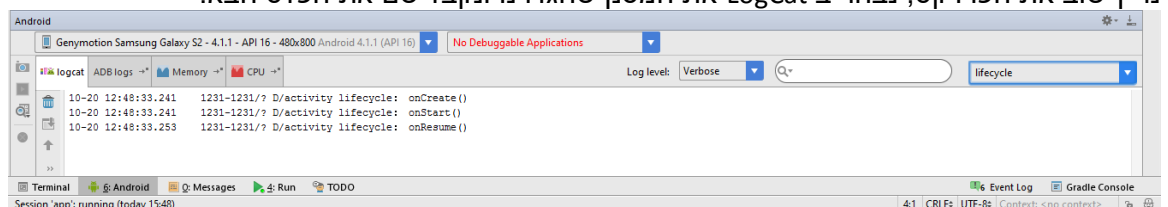
בחלונית שתיפתח ניצור מסנן חדש, השם שלו לא חשוב אבל העיקר שיכיל את אותו tag שהגדרנו



כדאי שנמחק את ההודעות שכבר קיימות:



נרץ שוב את הפרויקט, נבחר ב LogCat את המסנן שהגדרנו ונקבל שם את הפלט הבא:



כעת, יש לשחק עם האפליקציה:

לעזוב את האפליקציה באמצעות לחצן back ולראות מה ההודעות שמודפסות
להפעיל שוב את האפליקציה ולראות מה מודפס
לעזוב את האפליקציה, הפעם באמצעות כפתור home ולראות מה מודפס.
להמתין שהטלפון ינעל ולראות מה מודפס, מה מודפס שהוא משתחרר ??

תוודאו שאתם מבינים את ההתנהגות של מערכת האנדרואיד מבחינת מחזור החיים של ה activity
מה ההבדל בין יציאה מהאפליקציה באמצעות home ליציאה ממנה באמצעות back ?
מה לדעתכם יהיה הפלט אם באמצע שימוש באפליקציה תתקבל שיחת טלפון במכשיר?
מה יהיה הפלט אם תתקבל הודעה שהסוללה עומדת להיגמר?
מה יהיה הפלט לאחר שנצא מההודעה שקיבלנו על הסוללה?

כפתורים בסימולטור:

