بسمه تعالی

برنامه نویسی اندروید مقدماتی

(Introductory Android Programming)

دکتر مهدی نعمتی

drnemati.blog.ir

فصل اول : نصب برنامه های پیش نیاز

- نصب کیت توسعه جاوا (JDK)

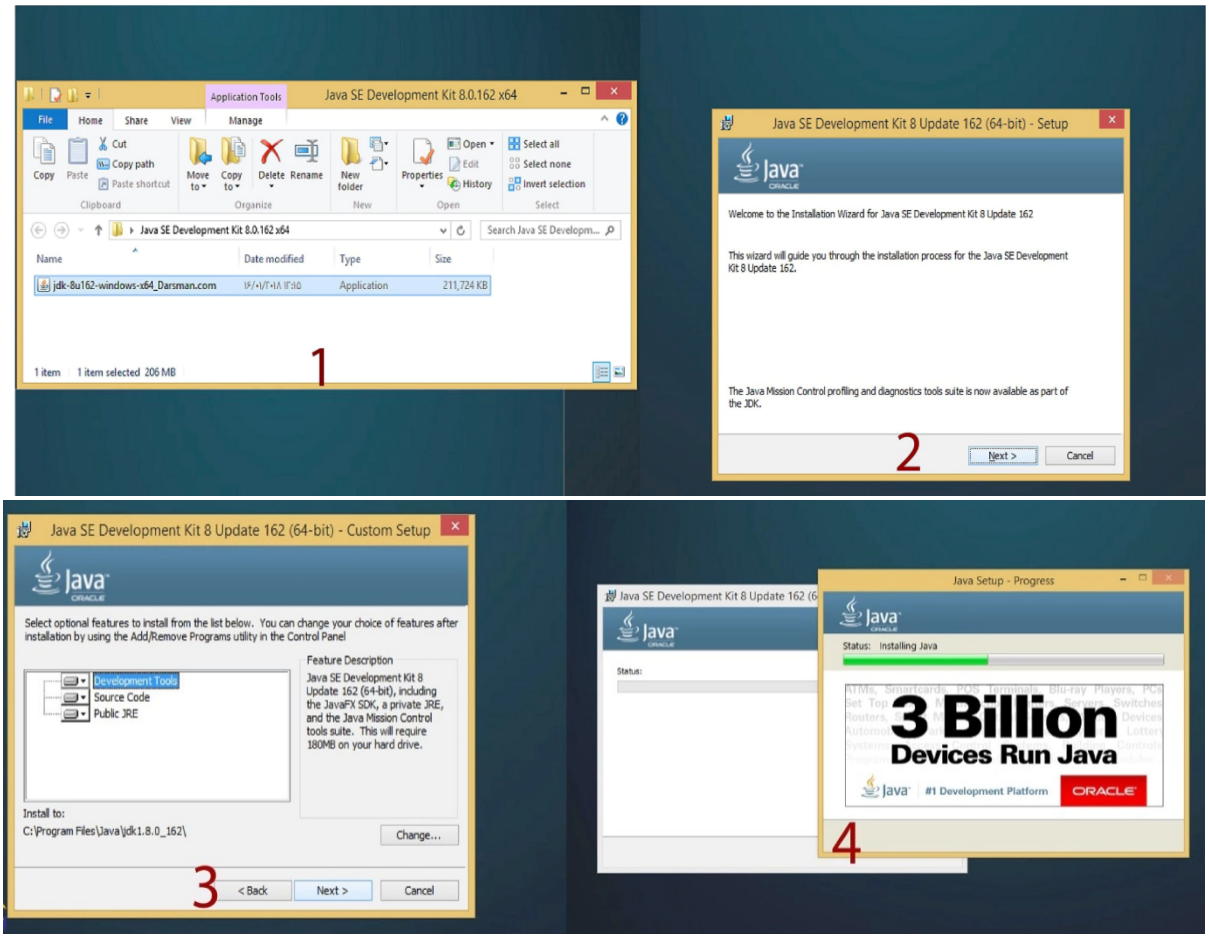
با زبان های برنامه نویسی مختلفی می توان کد برنامه های اندرویدی را نوشت که جاوا یکی از این زبانها است، که برای کد نویسی به زبان جاوا در اندروید استودیو از JDK استفاده می شود.

JDK (Java Development Kit) کیت توسعه جاوا می باشد که برای نصب و راه اندازی اندروید استودیو به آن نیاز داریم.

در فرایند ساخت برنامه اندرویدی کدهای برنامه برای اجرا به ماشین واسط (Runtime) و برای کامپایل کدهای جاوا به کامپایلر احتیاج دارند. ماشین واسط در زبان جاوا بر عهده Java Runtime Environment یا (JRE) و کامپایل کدهای جاوا بر عهده JDK می باشد. در نسخه های جدید منتشر شده JDK شامل JRE نیز می باشد. با نصب JDK هم ماشین واسط و نیز کامپایلر لازم برای زبان جاوا را نصب کرده اید.

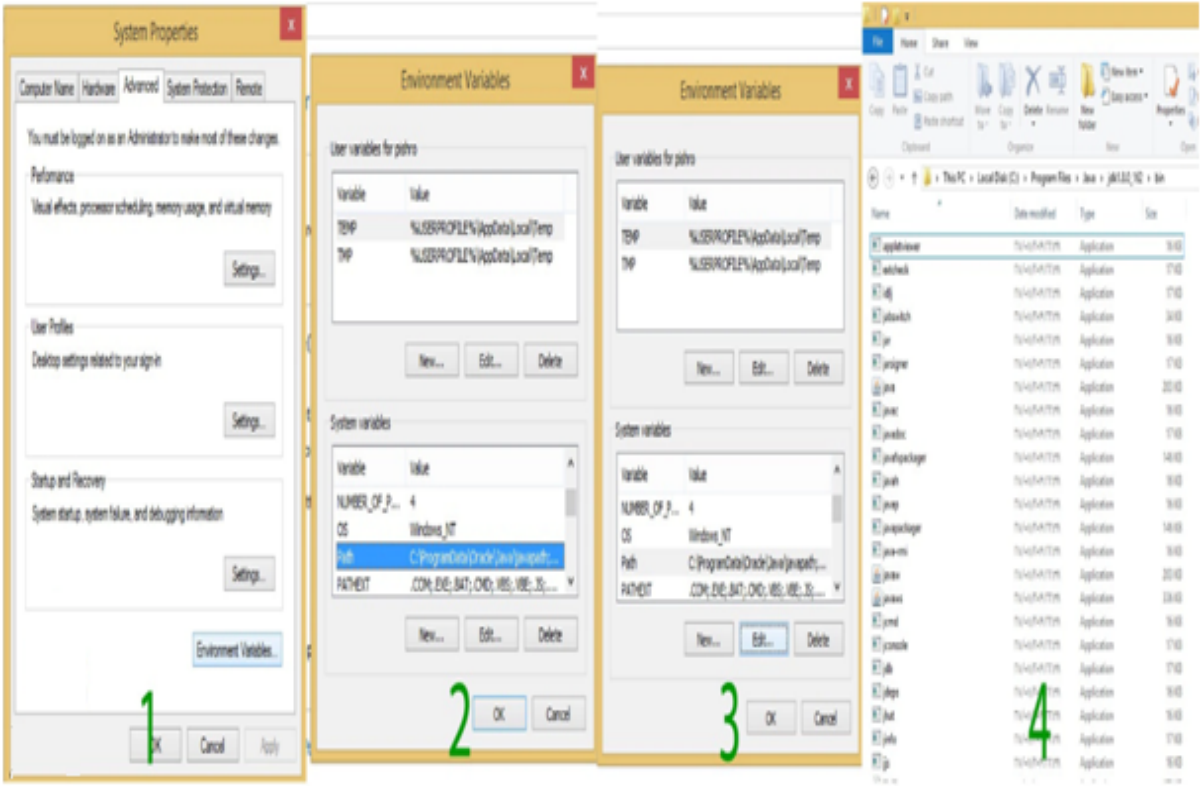
با توجه به معماری ویندوز نصب شده روی سیستم‌تان، نسخه x86 یا x64 را دانلود و نصب کنید. اکنون باید فایل دانلود شده را نصب کنید. در نصب نکته خاصی وجود ندارد و مانند اکثر نرم‌افزارهای دیگر، next را می‌زنیم تا کار تمام شود. فقط حواسمان باشد که برای استفاده در مراحل بعدی، آدرس محل نصب را به خاطر بسپاریم. (به طور پیش‌فرض آدرس C: \Program Files\Java\jdk1.8.0_172 است.)

در تصویر زیر فرایند نصب JDK را مشاهده می کنید.



همانطور که در تصویر می‌بینید فرایند نصب JRE, JDK به پایان رسیده است.

اکنون برای شناسایی این برنامه به سیستم روی آیکن PC یا My Computer راست کلیک کرده و گزینه properties را کلیک کنید، صفحه زیر باز می شود، از قسمت سمت چپ گزینه Advanced system settings را کلیک کنید.

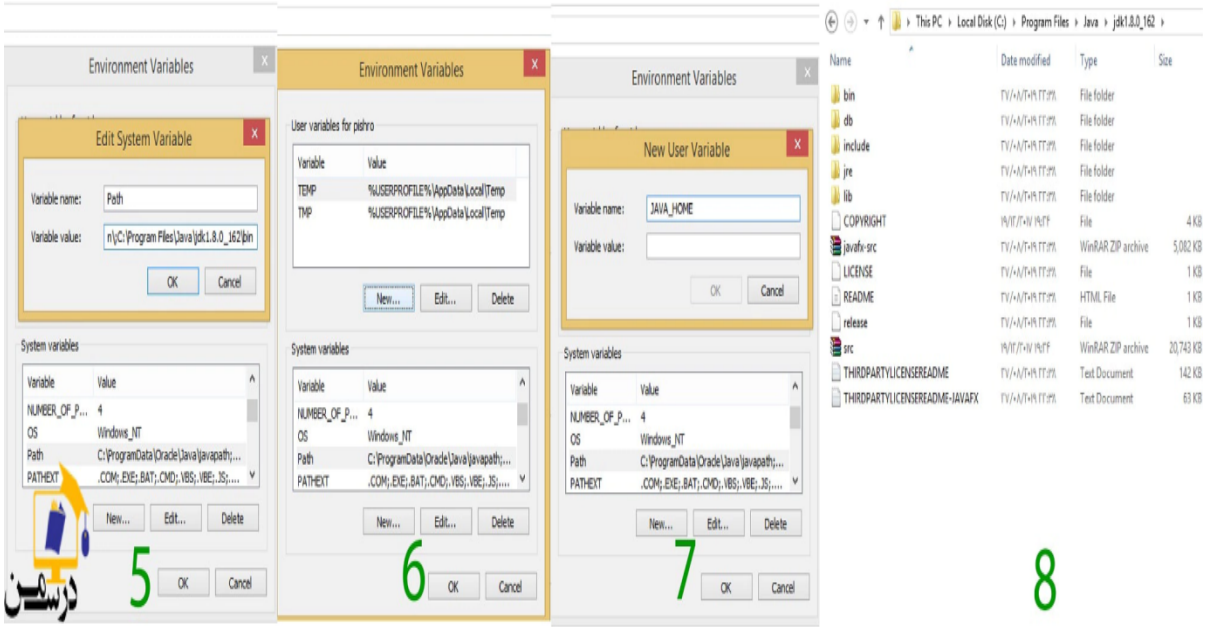


مطابق تصویر شماره ۱ بر روی گزینه Environment Variables کلیک کنید.

مطابق تصویر شماره ۲ گزینه Path را انتخاب کنید.

مطابق تصویر شماره ۳ گزینه Edit (ویرایش) را کلیک کنید.

مطابق تصویر شماره ۴ قبل از ویرایش اطلاعات در Path، به آدرس نصب فایل JDK رفته وارد پوشه bin شوید. آدرس را از نوار بالا کپی کنید.

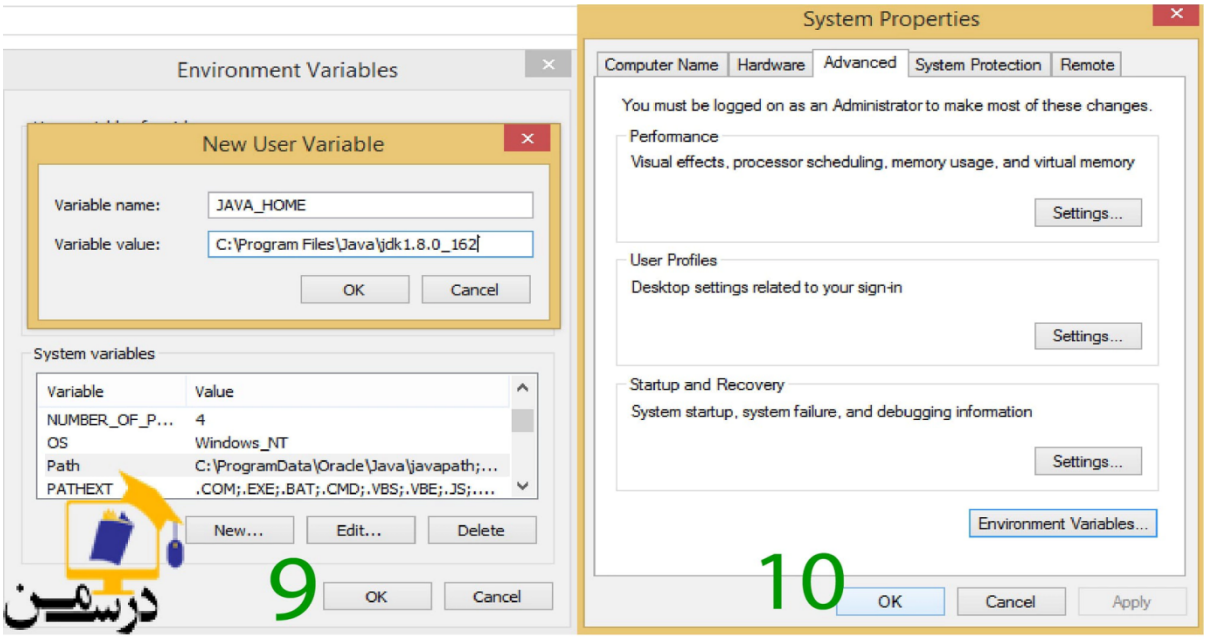


مطابق تصویر شماره ۵ آدرسی که در بالا کپی کرده اید را در قسمت Variable value وارد و Ok کنید.

مطابق تصویر شماره ۶ از قسمت User Variables دکمه New را کلیک کنید.

مطابق تصویر شماره ۷ در پنجره باز شده در قسمت Variable name عبارت JAVA_HOME را وارد کنید.

مطابق تصویر شماره ۸ آدرس محل نصب JDK را مطابق تصویر بیابید و آن را از نوار بالا کپی کنید.

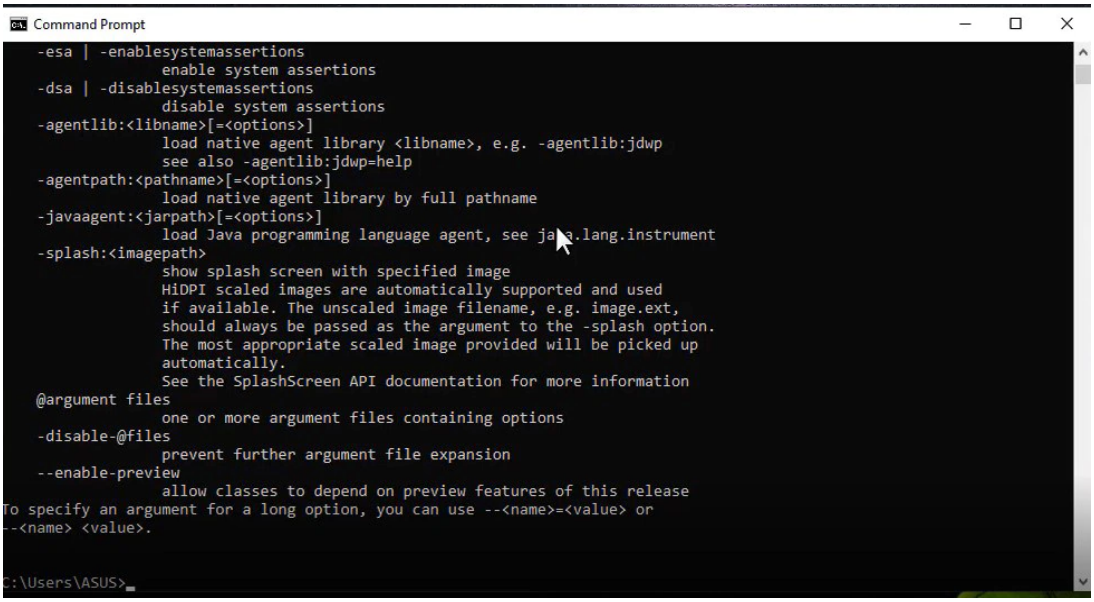


مطابق تصویر شماره ۹ آدرس کپی شده را در قسمت Variable value وارد کرده و ok کنید.

مطابق تصویر شماره ۱۰ ok را کلیک کنید.

اکنون نصب JDK که یکی از مراحل نصب و راه اندازی اندروید استودیو بود به پایان رسید.

نکته: برای اطمینان از نصب JDK در قسمت سرچ سیستم خود عبارت cmd را وارد کنید و command prompt را کلیک کنید. در پنجره باز شده عبارت جاوا را وارد کنید تا خروجی مورد نظر مشاهده شود. (مطابق تصویر زیر)



اگر پیام‌های بالا را مشاهده نکردید و خطایی با مضمون:

java is not recognized as an internal or external command, operable program or batch file.

دریافت کردید، احتمالا یکی از مراحل بالا را درست انجام نداده اید. پیشنهاد می شود آدرس پوشه‌ی bin که در environment variable وارد کرده‌اید را بررسی کنید.

- نصب کیت توسعه نرم‌افزار (SDK)

کلمه SDK مخفف کلمات Software Development Kit به معنی بسته توسعه نرم‌افزار است. همانطور که از اسم SDK مشخص است، ابزاری برای توسعه یک نرم‌افزار است که توسط شرکت سازنده در اختیار توسعه‌دهنده یا برنامه‌نویس قرار می‌گیرد تا بتواند یا استفاده از آن، برنامه خود را مطابق با پلتفرم شرکت هماهنگ کند. گوگل برای توسعه‌دهندگان اندروید یک SDK تهیه کرده که باید آن را نصب کنیم. برای دانلود SDK به این صفحه رسمی در گوگل بروید و برای سیستم عامل خود نسخه مناسب را دانلود کنید.

Command line tools only

If you do not need Android Studio, you can download the basic Android command line tools below. You can use the included `sdkmanager` to download other SDK packages.

These tools are included in Android Studio.

Platform	SDK tools package	Size	SHA-256 checksum
Windows	sdk-tools-windows-4333796.zip	148 MB	7e81d69c303e47a4f0e748a6352d85cd0c8fd90a5a95ae4e076b5e5f960d3c7a
Mac	sdk-tools-darwin-4333796.zip	98 MB	ecb29358bc0f13d7c2fa0f9290135a5b608e38434aad9bf7067d0252c160853e
Linux	sdk-tools-linux-4333796.zip	147 MB	92ffee5a1d98d856634e8b71132e8a95d96c83a63fde1099be3d86df3106def9

See the SDK tools release notes.

آن را در پوشه‌ای به جز درایو C بزیرید. این نیازی به نصب ندارد. برای مثال، یک پوشه به نام android studio در درایو D ایجاد کنید و فایل‌های دانلود شده را درون آن استخراج (extract) کنید.

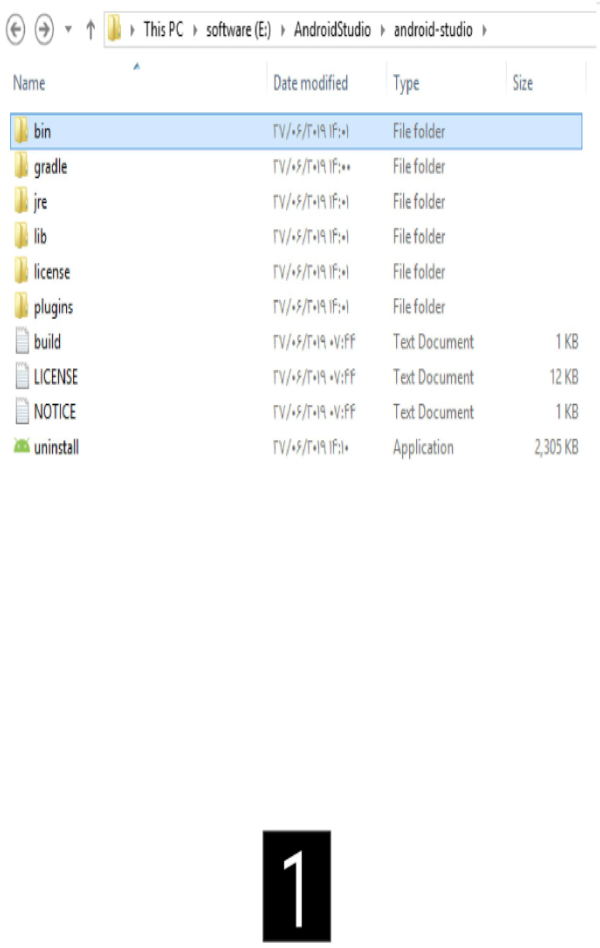
به عبارت دیگر برای کامل بودن فرایند نصب و راه اندازی اندروید استودیو بهتر است ابزارها و API ها (Application Programming Interface) که در واقع واسطی بین برنامه و امکانات ورژن های مختلف سیستم عامل اندروید است، را جداگانه دانلود و نصب کنید. که این مجموعه ابزارهای و API ها لازم برای ساخت اپلیکیشن را در قالب یه بسته نصبی به اسم SDK تهیه شده که با نصب آن می توانید یک محیط برنامه نویسی کامل داشته باشید.

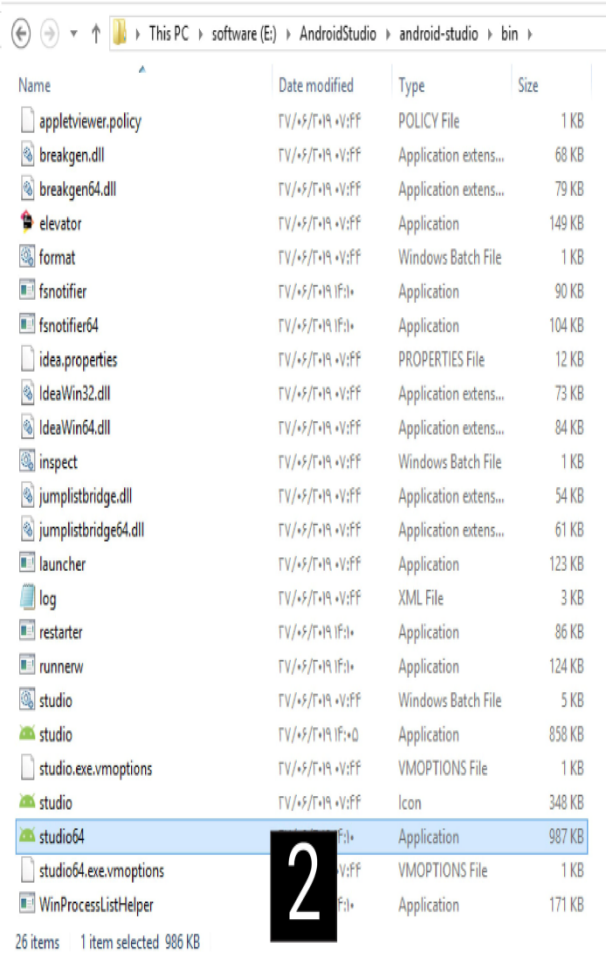
البته بهتر است نصب SDK بصورت جداگانه انجام نشود و نصب آن به همراه اندروید استودیو صورت گیرد.

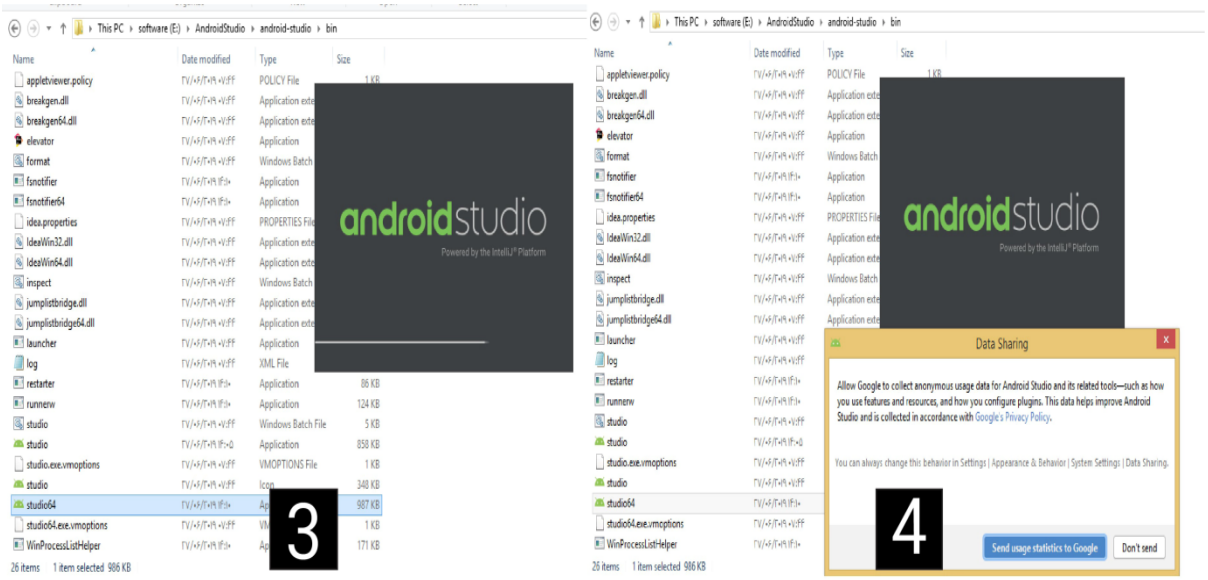
– نصب اندروید استودیو

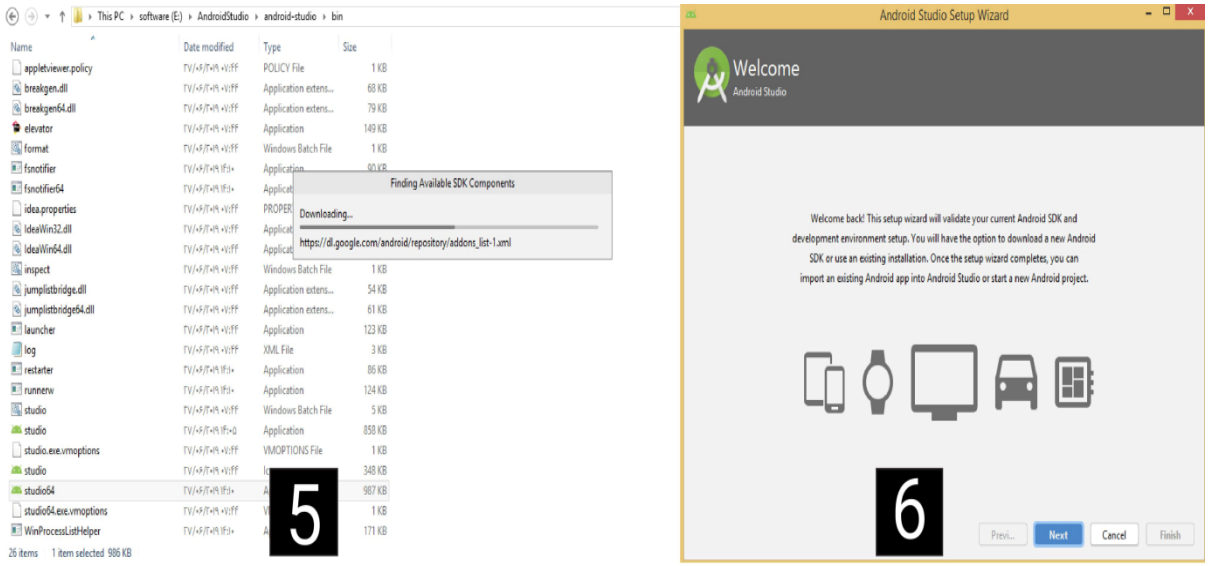
بعد از کلیک برای دانلود اندروید استودیو با صفحه ای روبه رو می شوید که از شما می خواهد قبل از دانلود، با شرایط و ضوابط گفته شده موافقت کنید. که پس از تایید آن دانلود آغاز می شود. در صحفه بعد از شما به عنوان یک توسعه دهنده از دانلود این محیط تشکر می کند و چند منبع برای مطالعه معرفی می کند.

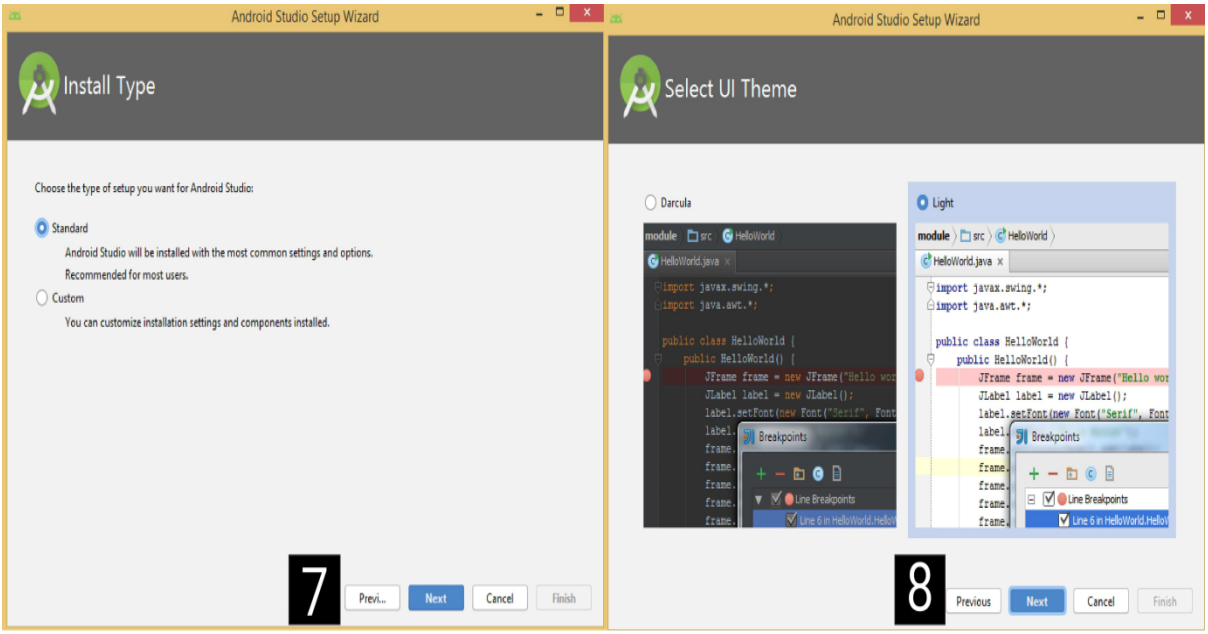
هنگامی که فرایند دانلود اندروید استودیو به پایان رسید، فایل را از حالت فشرده خارج کنید و بر روی فایل نصبی آن کلیک کنید. مراحل نصب اندروید استودیو را طبق تصاویر پشت سر بگذارید.






3
4
Data Sharing
Allow Google to collect anonymous usage data for Android Studio and its related tools—such as how you use features and resources, and how you configure plugins. This data helps improve Android Studio and is collected in accordance with Google's Privacy Policy.
You can always change this behavior in Settings | Appearance & Behavior | System Settings | Data Sharing.
Send usage statistics to Google
Don't send

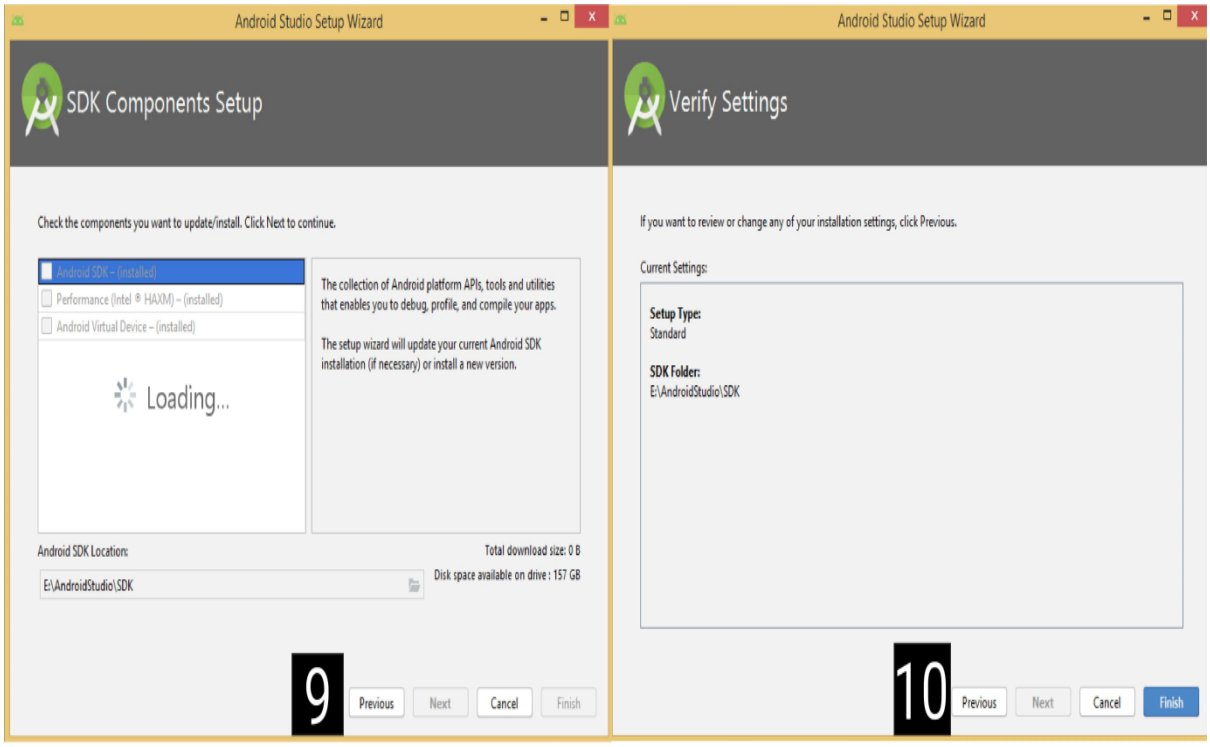

5
Finding Available SDK Components
Downloading...
https://dl.google.com/android/repository/addons_list-1.xml
6
Android Studio Setup Wizard
Welcome
Android Studio
Welcome back! This setup wizard will validate your current Android SDK and development environment setup. You will have the option to download a new Android SDK or use an existing installation. Once the setup wizard completes, you can import an existing Android app into Android Studio or start a new Android project.
Previ...
Next
Cancel
Finish


7
Android Studio Setup Wizard
Install Type
Choose the type of setup you want for Android Studio:
Standard
Android Studio will be installed with the most common settings and options.
Recommended for most users.
Custom
You can customize installation settings and components installed.
Previ...
Next
Cancel
Finish
8
Android Studio Setup Wizard
Select UI Theme
Darcula
Light
Previous
Next
Cancel
Finish

نکته: توجه داشته باشید که به اینترنت متصل باشید تا ابزارهایی را که هنگام نصب لازم دارد را دانلود کند.

در تصویر شماره ۷ شما می توانید یکی از دو گزینه های موجود را انتخاب کنید. گزینه Standard برای زمانی است که SDK را هنگام نصب دانلود می کند، اما گزینه Custom در فرایند نصب آدرس فایل SDK ای که قبلا دانلود کرده اید را می خواهد.

در تصویر شماره ۸ رنگ صفحه کد نویسی را مشخص می کنید.



در تصویر شماره ۹ مشاهده می کنید که با توجه به نوع انتخاب ما که Standard بوده فرایند دانلود SDk (در کادری که در حال Loading است) آغاز می شود که در قسمت Android SDK Location میتوانید مسیر ذخیره سازی آن را مشخص کنید. بهتر است محل ذخیره سازی SDK در درایوی جدا از محل نصب اندروید استودیو باشد.

نکته: اگر فرایند دانلود SDK کامل نشود این مرحله را میتوان پشت سر گذاشت.

اکنون فرایند نصب و راه اندازی اندروید استودیو به پایان رسیده است و میتوانید اولین پروژه خود را بسازید.

- دستگاه مجازی تست برنامه (AVD)

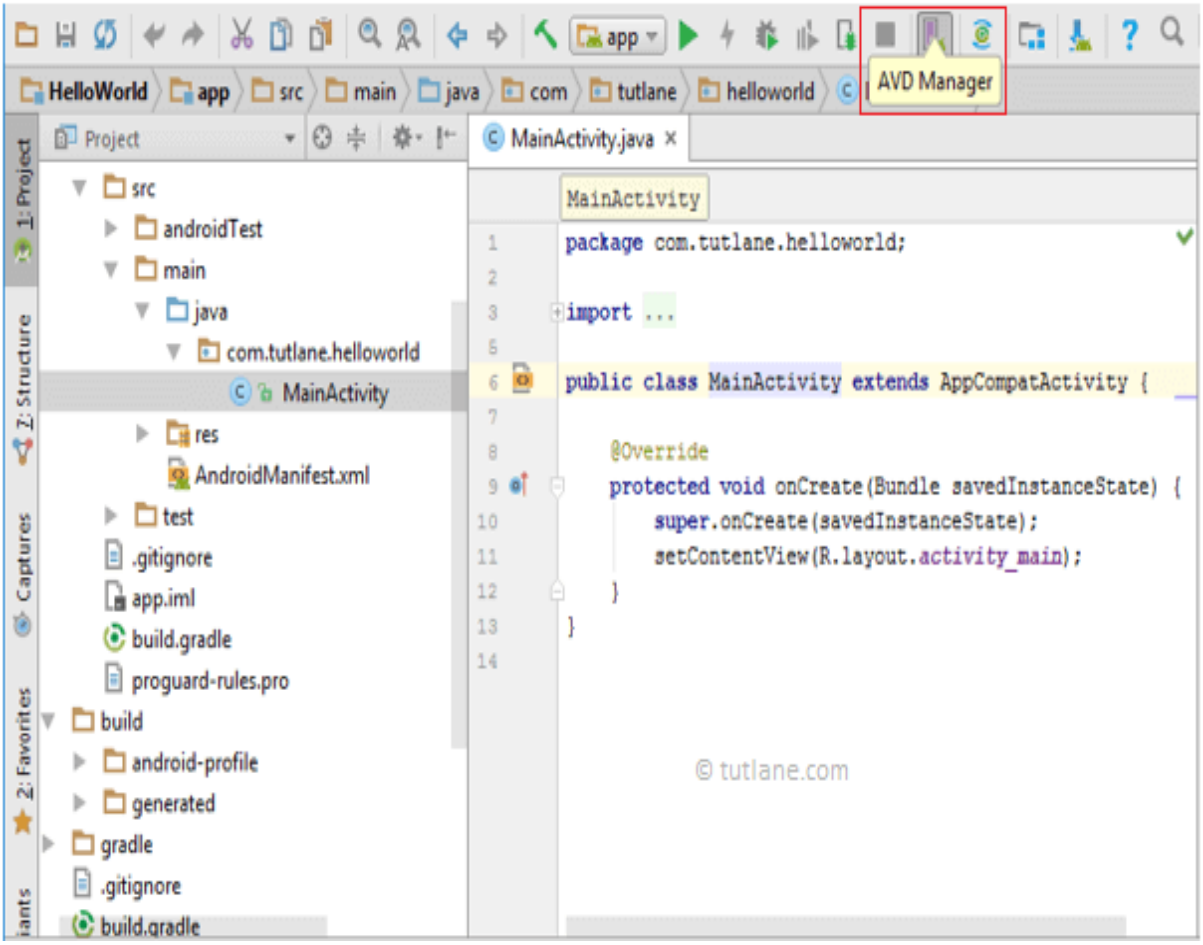
تست و رفع عیب (Debug) پروژه یکی از مهم‌ترین فرایندهای توسعه اپلیکیشن اندرویدی به شمار می‌رود. در این فرایند پروژه روی یک دستگاه حقیقی یا مجازی اندرویدی اجرا می شود تا توسعه‌دهنده بتواند کارکرد کدهای خود را بررسی کرده و نواقص احتمالی را برطرف کند.

برای تست برنامه های اندرویدی قبل از منتشر شدن در مارکت ها به یک ماشین مجازی اندرویدی جهت شبیه سازی (Emulator) در اندروید استودیو نیاز دارید تا برنامه ها را اجرا ، تست و دیباگ کنید و برای این کار باید با استفاده از رابط AVD Manager در اندروید استودیو یک شبیه ساز دستگاه مجازی اندرویدی را برای آزمایش برنامه های خود تنظیم کنید.

شبیه سازهای مختلفی برای اندروید ساخته شده که روی سیستم‌عامل‌های محبوب مانند ویندوز، لینوکس و مک قابل استفاده هستند. درحال حاضر Genymotion ، BlueStacks و MEMu جزء گزینه‌های مطرح به شمار می‌روند.

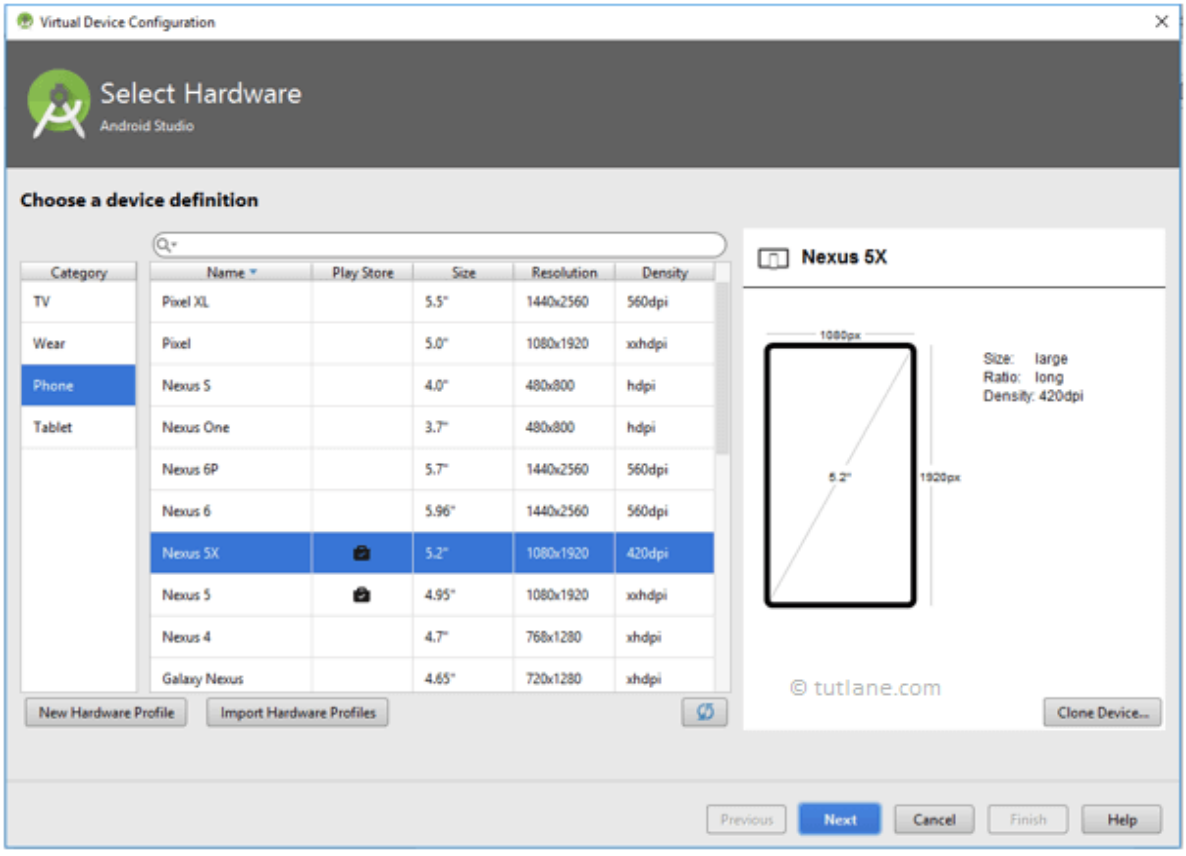
در اندروید استودیو از همان ابتدای کار یک امولاتور داخلی در اختیار توسعه‌دهندگان قرار گرفته بود که امکان مدیریت دیوایس‌ها با استفاده از ابزار Android Virtual Device یا AVD فراهم شده است.

برای اینکار با کلیک روی AVD Manager مانند تصویر زیر ، شروع به ساخت ماشین مجازی کنید.

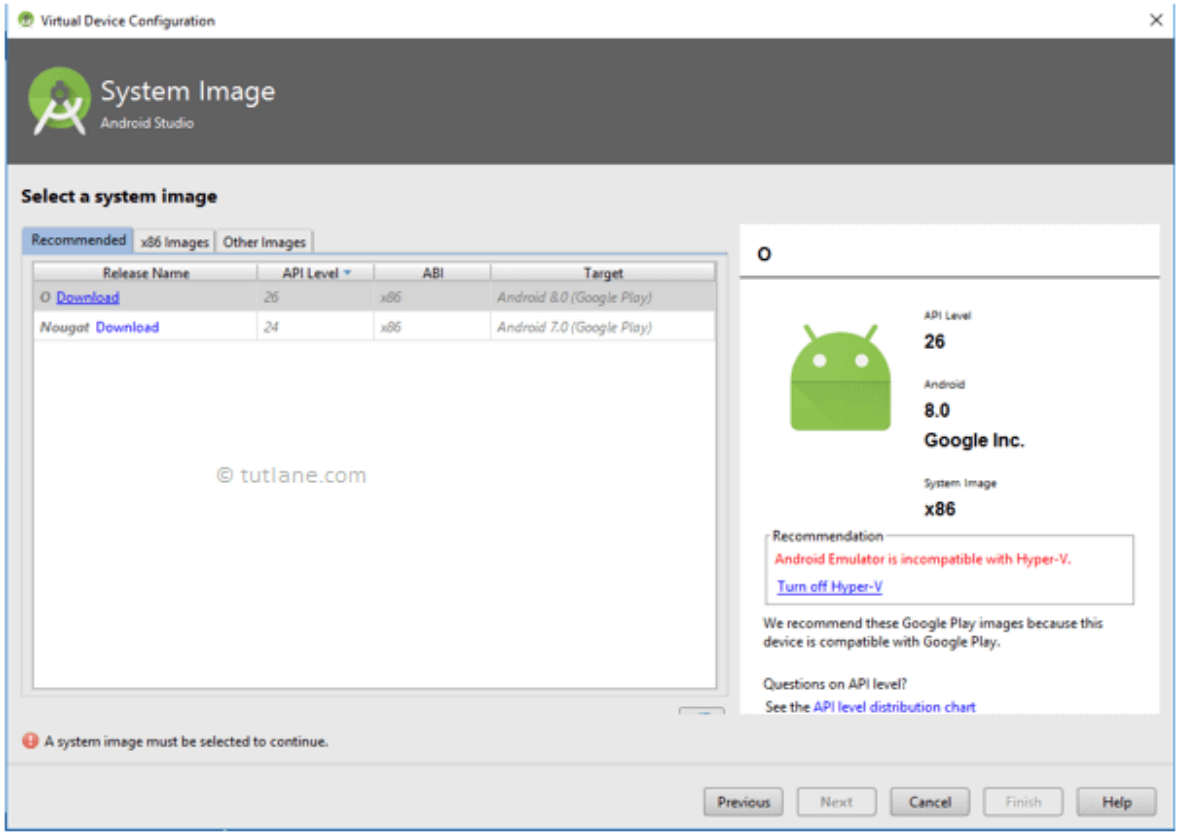


هنگامی که روی AVD Manager کلیک کردید، یک پنجره جدید باز می شود که مطابق شکل زیر بر روی ایجاد دستگاه مجازی کلیک می کنید.

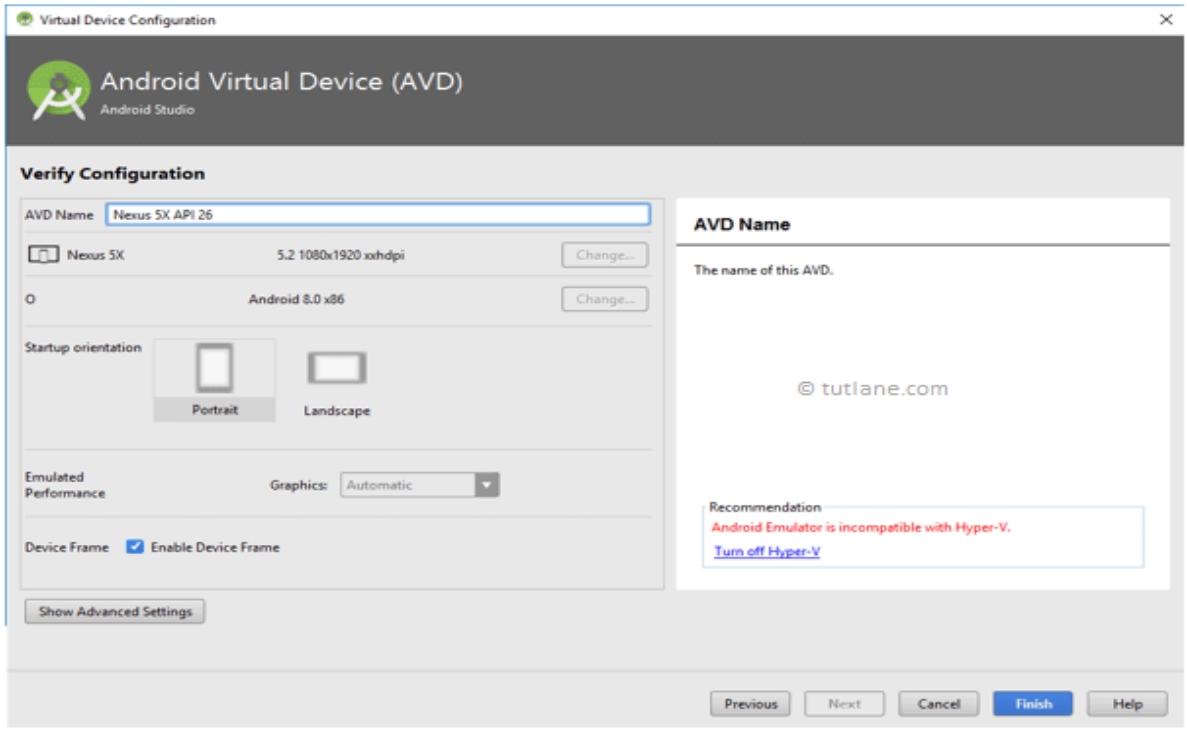
در پنجره باز شده در قسمت چپ پنل نوع دستگاه (Device) را انتخاب کنید و سپس در پنل وسط نسبت به انتخاب مدل دستگاه اقدام کنید. همانطور که می بینید به طور پیشفرض بروی Nexus تنظیم شده است و بر روی آن شبیه ساز Play Store هم نصب است. همچنین مشخصاتی نظیر اندازه (Size)، رزولیشن (Resolution) و تراکم پیکسل (Density) نیز نشان داده شده است. اکنون نوع دستگاه مورد نیاز خود را انتخاب کرده و بر روی Next کلیک کنید.



حال باید تصویر سیستم را بارگیری و انتخاب کنید و مانند تصویر زیر بر روی Next کلیک کنید.



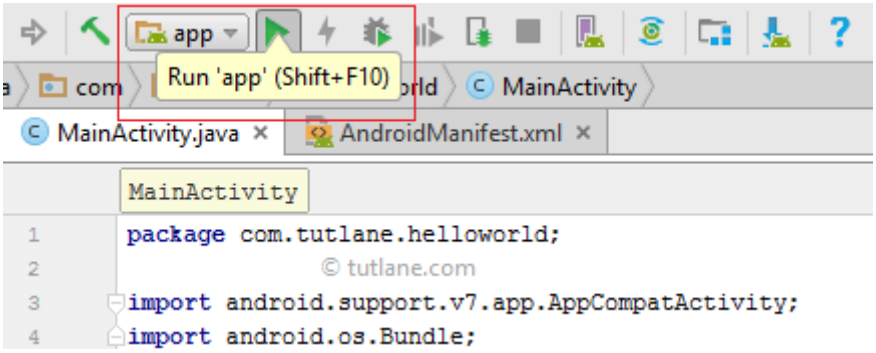
اکنون پیکربندی دستگاه مجازی اندرویدی (AVD) تمام شد و مطابق شکل زیر بر روی گزینه Finish کلیک کنید.



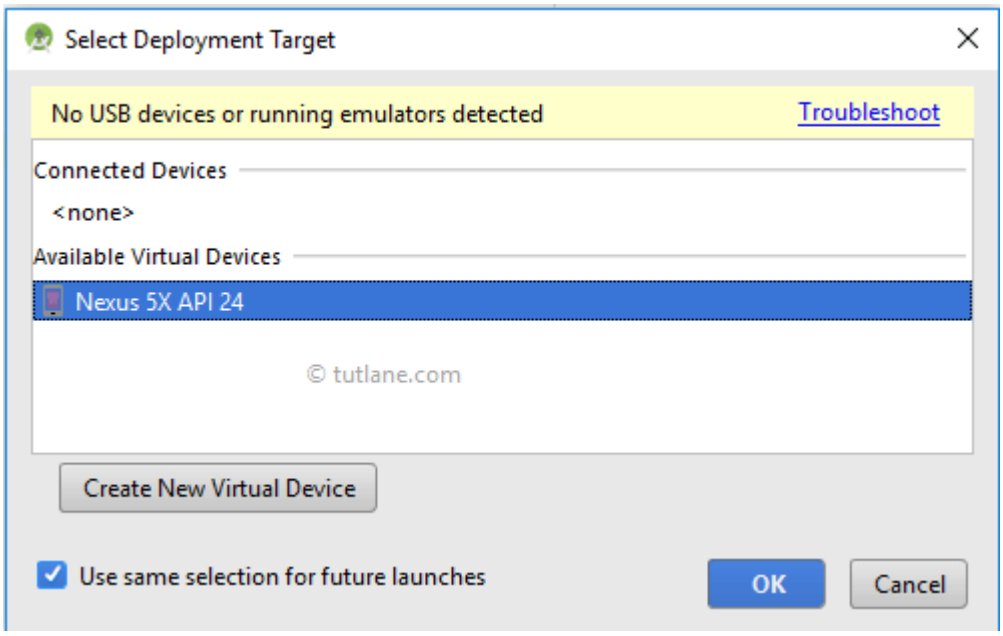
به این صورت شما توانستید برای آزمایش برنامه های کاربردی اندرویدی، ماشین مجازی اندرویدی (AVD) را در اندروید استودیو اضافه کنید.

برای اجرای ماشین مجازی اندروید نیز پس از اتمام کار نصب در اندروید استودیو، یک برنامه (Application) بسازید و برنامه را با AVD manager اجرا کنید.

برای اجرای برنامه های اندرویدی در اندروید استودیو باید روی دکمه Run کلیک کرده یا مانند شکل زیر Shift + F10 را فشار دهید.



بعد از کلیک بر روی این دکمه، پنجره جدیدی باز می شود که باید نام ماشین مجازی اندروید مورد نظر را انتخاب کنید و مانند تصویر زیر بر روی OK کلیک کنید.

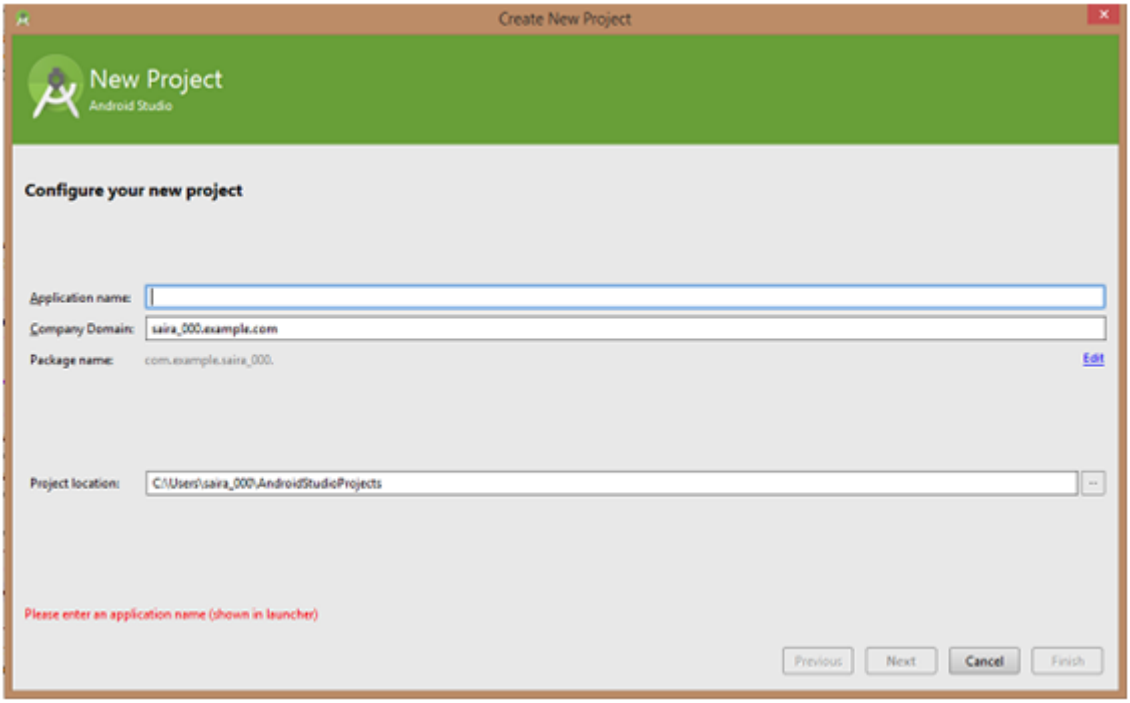


اینگونه است که می توانیم یک شبیه ساز دستگاه مجازی اندرویدی (AVD) را در اندروید استودیو تنظیم کنیم تا قابلیت های دستگاه های واقعی اندرویدی را تست و آزمایش و بررسی کنیم.

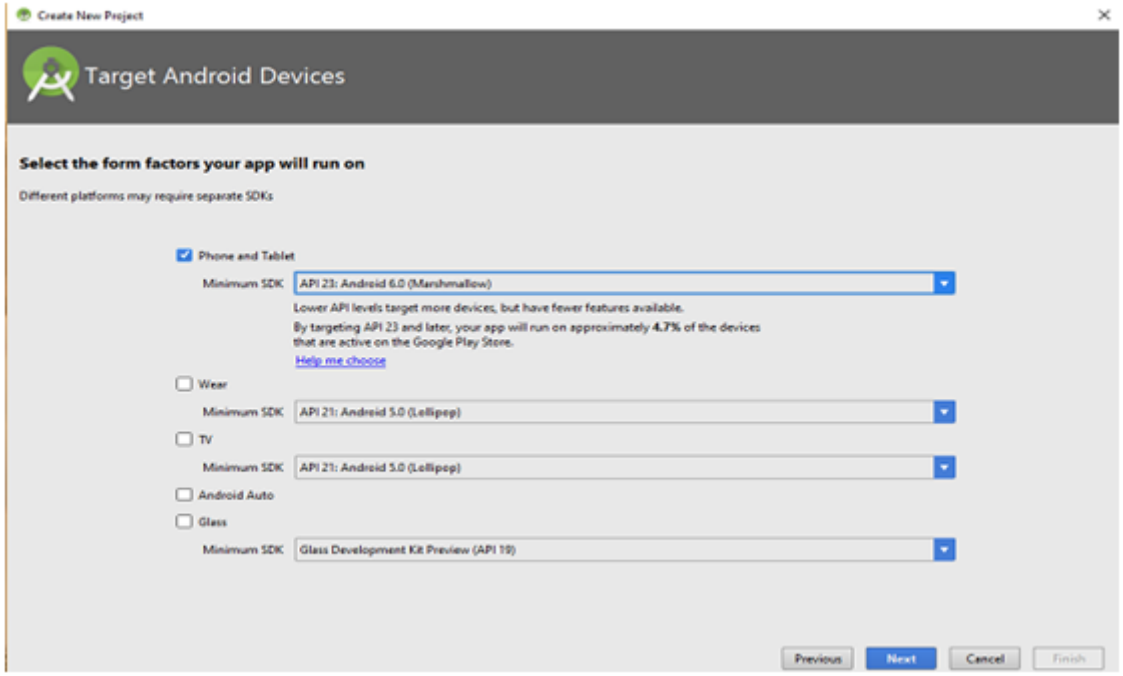
فصل دوم: شروع برنامه نویسی اندروید و آشنایی با ساختار آن

برای شروع در هر زبان برنامه نویسی اولین پروژه Hello World! است که ما هم قصد داریم یک اپلیکیشن ساده در محیط اندروید استودیو بسازیم و متن Hello World را درون آن نمایش دهیم.
برای اولین بار که محیط Android Studio که قبلا در سیستم عامل خود نصب کردید را باز می کنید با تصویری مشابه زیر مواجه می شوید:

شما میتوانید با انتخاب گزینه ی Start a new Android Studio project یک پروژه ی جدید اندرویدی بسازید و شروع کنید اپلیکیشن خود را برنامه نویسی کنید.
بعد از کلیک روی این گزینه وارد صفحه ی دیگری میشوید که اطلاعات اولیه اپلیکیشن شامل نام و نام پکیج و.. را از شما سوال می کند که مشابه تصویر زیر خواهد بود:



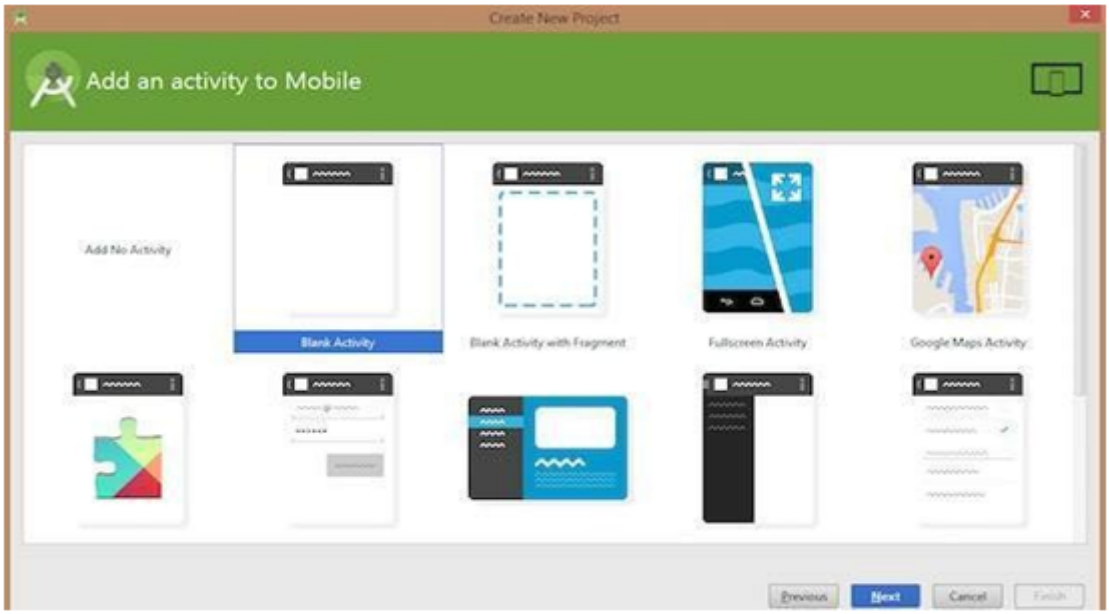
بعد از وارد کردن نام اپلیکیشن در قسمت Application name باید نام Company domain را برای ساختن پکیج نیم یا نام پکیج اپلیکیشن وارد کنید سپس محل قرار گیری پروژه که location هست را بصورت پیشفرض رها کنید و روی Next بزنید تا با صفحه ی زیر مواجه شوید:



در این بخش شما باید Minimum SDK را انتخاب کنید که به معنی حداقل نسخه ی اندرویدی است که اپلیکیشن شما روی آن میتواند نصب شود. هر چه این مقدار نسخه ی پایینتری باشد پوشش بیشتری دارد اما امکانات کمتری دارد چون در نسخه های جدید اندروید امکانات جدیدتری اضافه شده است ولی پوشش کمتری دارد چون خیلی از دستگاه های اندرویدی به نسخه های جدیدتر آپدیت نکرده اند.

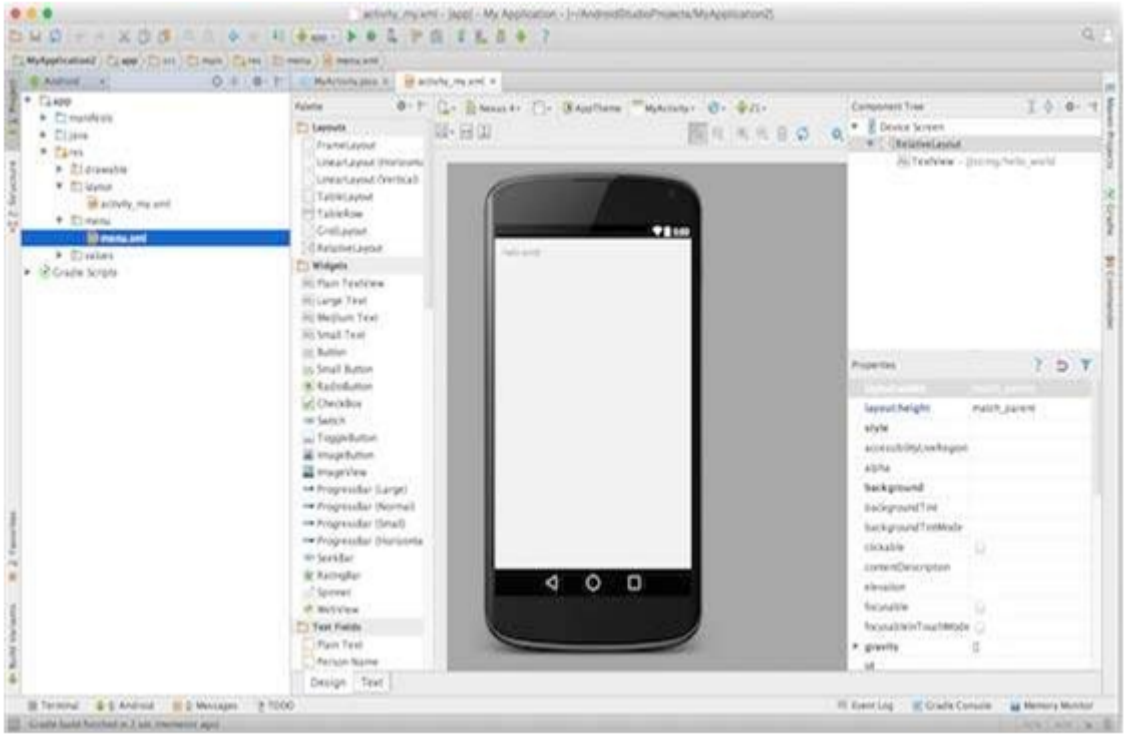
در این آموزش ما نسخه API23: Android 6.0(Mashmallow) را در نظر می گیریم.

سپس روی next کلیک کنید و در صفحه ی بعدی باید مشخص کنید که اولین اکتیویتی شما چه چیزی باشد یا چه قالب های آماده ای را نیاز دارید؟



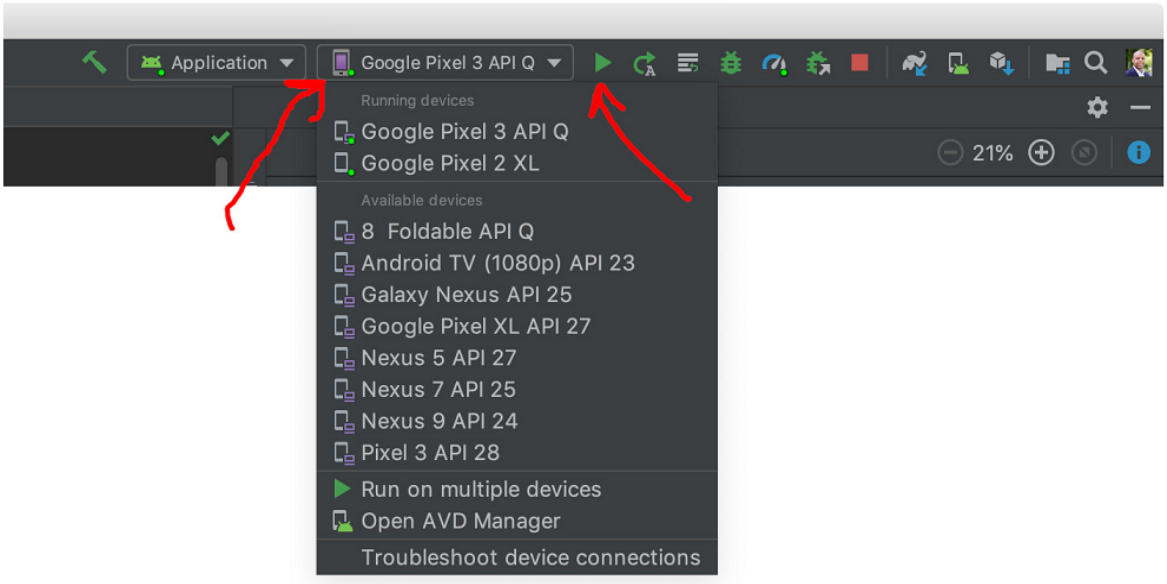
بهتر است Empty Activity را انتخاب کنید و next را بزنید.

سپس پروژه ی اندرویدی شما ساخته می شود و مطابق تصویر زیر محیط کدنویسی برای شما آماده است :



برای اجرای اپلیکیشن Hello World! ابتدا AVD را اجرا می کنیم. AVD همان ماشین مجازی اندرویدی هست که یک امولاتور یا شبیه ساز گوشی های اندرویدی برای نصب و تست اپلیکیشن ها درون کامپیوتر است.

برای اجرای یک پروژه در محیط Android Studio طبق تصویر زیر آیکون پلی سبز رنگ را بزنید تا اجرا شود که همان دکمه ی run می باشد:



اگر اپلیکیشن به درستی روی ماشین مجازی اندرویدی شما اجرا شد تبریک میگوییم شما اولین اپ اندرویدی خود را نوشتید.

قبل از اینکه پروژه جدید اندرویدی را اجرا کنید بهتر است با ساختار پوشه ها و فایل های موجود در محیط اندروید استودیو آشنا شوید.

- کامپوننت های اصلی برنامه نویسی اندروید

کامپوننت های اپلیکیشن بلاک های ضروری و اصلی یک اپلیکیشن اندرویدی هستند. در برنامه نویسی اندروید کامپوننت ها درون فایل شناسنامه ای یا همان AndroidManifest.xml تعریف و نحوه ارتباط آن با دیگر بخش های اپلیکیشن تعریف می شود.

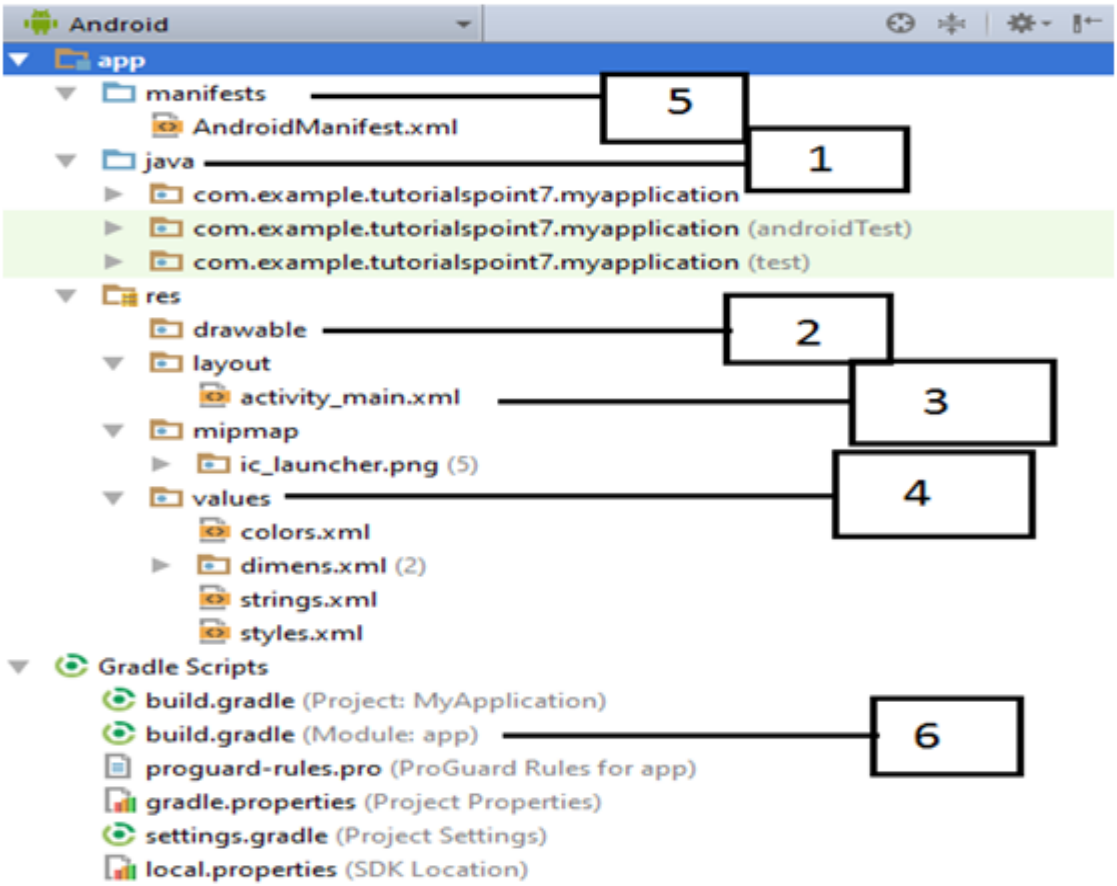
در زیر چهار نوع کامپوننت (component) معرفی شده است که اصلی ترین و پرکاربرد ترین کامپوننت ها در برنامه نویسی اپلیکیشن های اندرویدی هستند:

نام کامپوننت	توضیحات و استفاده
Activities	این کامپوننت هرچیزی که در صفحه ی نمایش به کاربر نمایش داده می شود را فراهم می کند هر چیزی که روی صفحه است یک اکتیویتی هست
Services	پردازش هایی که نیاز است پشت صفحه انجام بگیرد وظیفه ی سرویس هاست مانند پخش موسیقی در پشت صفحه یا دانلود منیجر که یک فایل دانلود می کند
Broadcast Receivers	این قسمت ارتباط بین سیستم عامل و اپلیکیشن را فراهم می کند
Content Providers	این قسمت داده های بین اپلیکیشن ها یا دیتابیس را مدیریت می کند

در ساخت صفحات می توان از اجزای زیر استفاده کرد تا بین یک دیگر در ارتباط باشند.

ردیف	اجزاء و توضیحات
۱	**Fragments** : بخشی از رابط کاربری را در یک Activity نشان می دهد.
۲	**Views** : عناصر رابط کاربری که بر روی صفحه نمایش کشیده شده اند شامل دکمه ها، لیست ها و غیره.
۳	**Layouts** : مشاهده سلسله مراتب هایی که قالب صفحه نمایش و ظاهر view ها را کنترل می کند.
۴	**Intents** : پیام هایی برای تقاضای انجام کاری از سیستم
۵	**Resources** : عناصر خارجی مانند رشته ها، ثابت ها و تصاویر قابل چاپ.
۶	**Manifest** : فایل پیکربندی برای برنامه

در تصویر زیر به آناتومی یک پروژه پرداخته شده است:



شماره بخش	توضیحات
1	Java هم اضافه شده kt. است البته جدیدا فایل های با پسوند java. این بخش دارای فایل هایی با پسوند است که دراین قسمت فایل های زبان جاوا و کاتلین قرار داده می شود برای اولین بار که پروژه ی در این قسمت ساخته می MainActivity اندرویدی میسازید بصورت پیشفرض یک فایل با نام شود که اکتیویتی اصلی اپ شماست.
2	res/drawable-hdpi به این معنی است hdpi این قسمت شامل اشیای گرافیکی برای صفحات با رزولیشین بالا هستد و.
3	res/layout اپ قرار میگیرد ui گرافیکی رابط کاربری یا xml این قسمت فایل های.
4	res/values است که مقادیر مورد نیاز پروژه مثل رشته ها ، رنگ بندی ها و xml این قسمت هم شامل فایل های .... قرار میگیرد
5	AndroidManifest.xml این فایل شناسنامه ی اپلیکیشن شماست که تمام اجرا درون آن تعریف می شود مجوزهای مورد نیاز نیز همینجا تعریف می شود و تمام اطلاعات مورد نیاز اپ اینجاست.
6	Build.gradle این قسممت مربوط به سیستم بیلدینگ است که اپلیکیشن را از روی کدها میسازد که شامل موارد compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName است

در قسمت پایین مرور کوتاهی به بخش های مختلف پروژه های اندرویدی خواهیم داشت که اهمیت بیشتری دارند.

- فایل MainActivity چیست؟

این فایل یک کلاس با زبان برنامه نویسی Java یا زبان برنامه نویسی Kotlin است که بصوت MainActivity.java یا MainActivity.kt درون فایل های پروژه وجود دارد.
معمولا این فایل در ابتدای ساخته شدن اپلیکیشن بعنوان اولین اکتیویتی یا جایی که اپ شروع می شود ساخته می شود و کدی مشابه کد زیر را دارد:

```
package com.example.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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

در کدهای بالا عبارت R.layout.activity_main به فایل activity_main.xml اشاره می کند که در مسیر res/layout قرار دارد. متود onCreate متودی است که وقتی برای اولین بار اپلیکیشن اجرا می شود صدا زده می شود.

- فایل Android Manifest در پروژه های اندرویدی

این فایل یک فایل xml است که شناسنامه و مجوز ها و هر اجرایی که در اپلیکیشن دارید باید اینجا تعریف شود. این فایل توسط سیستم عامل اندروید خوانده می شود و تمام کامپوننت های اندرویدی که درون اپ شما وجود داشته باشد باید درون این فایل نیز تعریف شده باشند.
برای مثال کد زیر یک نمونه ی ساده از محتویات یک فایل AndroidManifest.xml است:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns: android="http: //schemas.android.com/apk/res/android"
   package="ir.avasam.listviewexample">

   <application
      android: allowBackup="true"
```

```
        android: icon="@mipmap/ic_launcher"
        android: label="@string/app_name"
        android: roundIcon="@mipmap/ic_launcher_round"
        android: supportsRtl="true"
        android: theme="@style/Theme.ListViewExample">
        <activity
            android: name=".UserInformationActivity"
            android: exported="false" />
        <activity
            android: name=".RecyclerViewActivity"
            android: exported="true">
            <intent-filter>
                <action android: name="android.intent.action.MAIN" />

                <category android:
name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android: name=".MainActivity"
            android: exported="true">
            <intent-filter>
                <action android: name="android.intent.action.MAIN" />

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

</manifest>
```

در فایل manifest یک تگ به نام <application>...</application> وجود دارد که تگ اصلی هست و تمام تگ های دیگر را درون خود جای می دهد.

ویژگی android: icon به آیکون اصلی برنامه اشاره می کند که در مسیر res/drawable-hdpi وجود دارد. اپلیکیشن از یک فایل تصویری در مسیر drawable ها به نام ic_launcher.png استفاده می کند.

تگ <activity> برای معرفی یک اکتیویتی جدید استفاده می شود که درون آن ویژگی android: name به نام کلاس اکتیویتی اشاره می کند و android: label به عنوان یا لیبل اکتیویتی اشاره دارد. با استفاده از تگ <activity> می توانید هر چند تعداد اکتیویتی را درون مانیفست تعریف کنید.

تگ action که درون intent-filter قرار دارد و نام آن android.intent.action.MAIN می‌باشد برای مشخص کردن اکتیویتی اصلی اپلیکیشن می باشد این اکتیویتی نقطه ورود به اپلیکیشن می باشد و برنامه از اینجا شروع می شود.

تگ category که درون intent-filter می باشد و نام آن android.intent.category.LAUNCHER می باشد مشخص می کند که برنامه از قسمت Luncher ایکون قابل اجرا شدن می باشد.

@string به فایل strings.xml اشاره می کند از این رو ، @string/app_name به یک متغیر رشته ای اشاره می کند که نام آن app_name می باشد و درون فایل strings.xml تعریف شده است که مقدار آن "HelloWorld" می باشد ، از این روش در داخل کدنویسی اپلیکیشن هم برای پر کردن جاهای خالی رشته ها استفاده می کنیم.

در لیست زیر تگ هایی که برای تعریف کامپوننت های اصلی اندروید درون AndroidManifest.xml استفاده می شود را مشاهده می کنید:

<activity> برای تعریف اکتیویتی ها

<service> برای تعریف سرویس ها

<receiver> برای تعریف ریسیورها

<provider> برای تعریف provider ها

- فایل Strings.xml در برنامه نویسی اندروید

فایل strings.xml در مسیر res/values در پروژه های اندروید استودیو می باشد که شامل تمامی رشته های مورد استفاده در قسمت های مختلف اپلیکیشن است. برای مثال عنوان دکمه ها ، labelها ، متون پیشفرض و. .. درون این فایل تعریف می شوند. در زیر یک مثالی از محتوای strings.xml را مشاهده می کنید:

```
<resources>
  <string name="app_name">HelloWorld</string>
  <string name="hello_world">Hello world!</string>
  <string name="menu_settings">Settings</string>
  <string name="title_activity_main">MainActivity</string>
</resources>
```

- فایل های Layout در برنامه نویسی اندروید

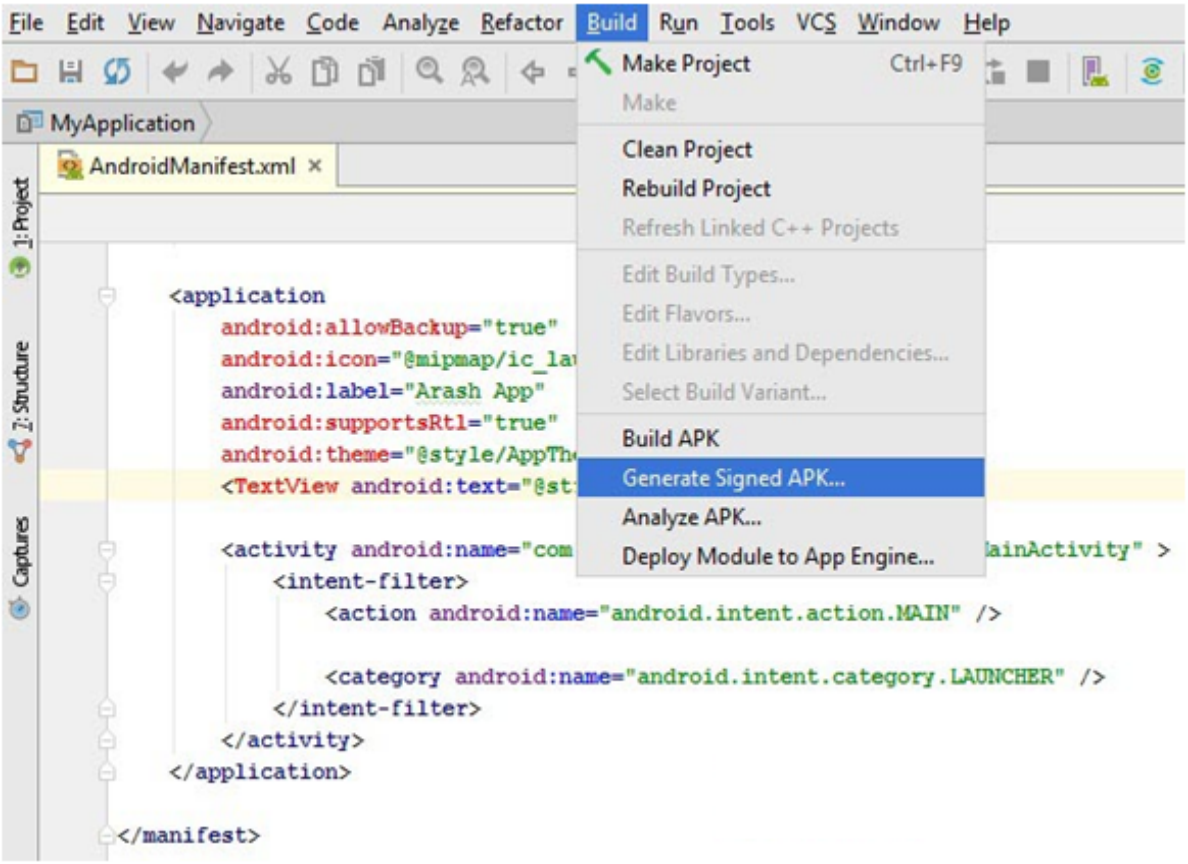
فایلی به نام activity_main.xml در مسیر res/layout پروژه قرار دارد که مسئولیت ساختن رابط کاربری یا UI یا همان ظاهر اپلیکیشن را برعهده دارد. فایل هایی که در این مسیر قرار دارند و xml هستند همگی برای تغییر ظاهر و گرافیک اپلیکیشن مورد استفاده ی برنامه نویسان اندرویدی قرار میگیرند.اینجا در اپلیکیشن نمونه ای که ساختیم محتویات درون activity_main.xml به صورت زیر خواهد بود:

```
<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" >

   <TextView
      android: layout_width="wrap_content"
      android: layout_height="wrap_content"
      android: layout_centerHorizontal="true"
      android: layout_centerVertical="true"
      android: padding="@dimen/padding_medium"
      android: text="@string/hello_world"
      tools: context=".MainActivity" />

</RelativeLayout>
```

این یک مثال ساده از RelativeLayout می باشد.
در کدهای بالا TextView یک ویجت گرافیکی برای نمایش متن هست که انواع ویژگی های خاص خود را دارد ویژگی های android: layout_width و android: layout_height برای تنظیم کردن طول و عرض ویجت استفاده می شود و بقیه ویژگی ها هم برای تغییر ظاهر استفاده می شود.
همانطور که میبینید ویژگی text این ویجت به مسیر @string/hello_world اشاره دارد که یک متغیر به نام hello_world درون فایل strings.xml هست و مقدار آن هر چه باشد در این قسمت قرار داده می شود.

- ساخت فایل نصبی اندروید (APK)

همان طور که می دانید فایل های نصبی ویندوزی exe هستند. فایل های نصبی اندرویدی نیز فرمت APK که خلاصه شده android application package است.
وقتی بخواهیم یک برنامه اندرویدی بسازیم و آن را صادر کنیم، باید آن برنامه را امضا کنیم! امضا کردن برنامه یعنی نوشتن اسم و برخی از مشخصات خود به عنوان سازنده برنامه.
همان طور که در عکس زیر نشان داده شده است، در قسمت Build با انتخاب گزینه Generate Signed APK می توانیم یک برنامه با خروجی APK تولید کنیم.



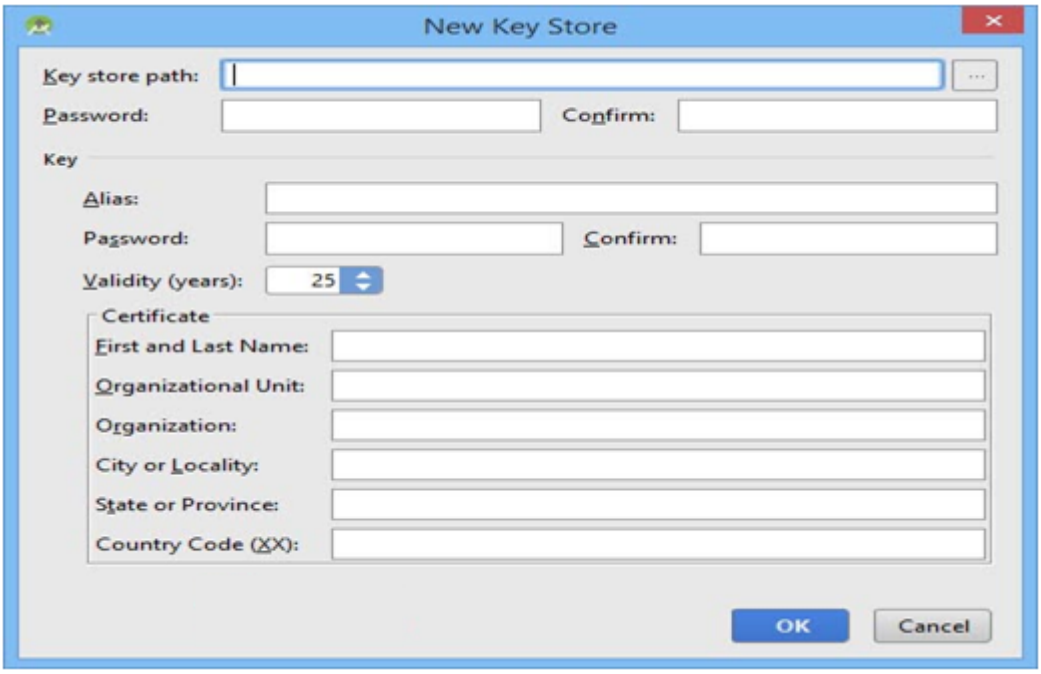
با کلیک بر روی Generate Signed APK ، پنجره ی زیر باز می شود.

صبر کنید کمی در رابطه با گزینه Key Store Path ، توضیح دهیم. همان طور که گفتیم برای هر برنامه باید امضاهایی بسازیم. این امضاها که مشخص کننده نام مالک و ... می باشد می تواند بر روی تمامی برنامه هایی که می سازیم قرار گیرد. زیرا می توان یک امضای واحد ساخت و محل قرار گیری آن را در این قسمت مشخص نمود. سپس در دفعات بعدی از همان امضا استفاده کرد. اما اگر یک امضای یکسان برای دو برنامه با دو ورژن متفاوت نداشته باشیم، برنامه بر روی نسخه قبلی نصب نمی شود.

پس در این قسمت سعی می کنیم که یک امضای واحد ساخته و سپس در محلی آن را ذخیره نماییم.

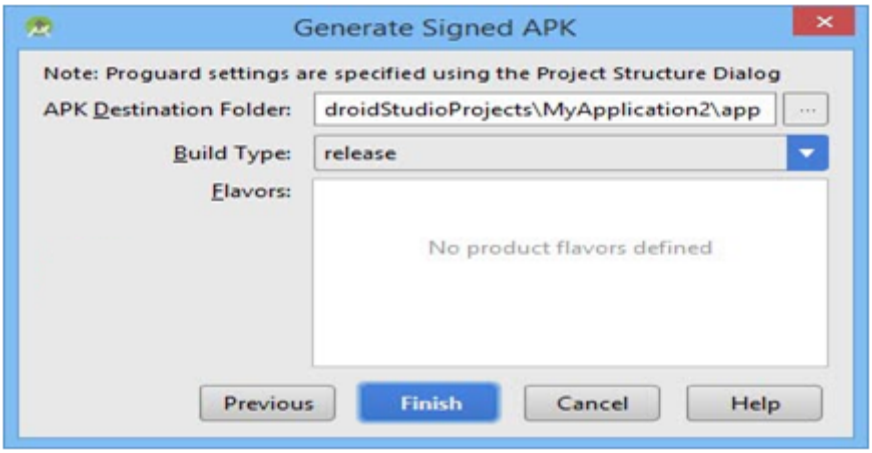
اگر از قبل امضای ساخته شده خود را داریم بر روی گزینه Choose existing و اگر نداشتیم بر روی گزینه Create New کلیک می کنیم.

یک پنجره با عنوان New Key Store باز می شود. این پنجره از ما اطلاعاتی در مورد امضا خواستار است.



قسمت Key Store Path مربوط به محل ذخیره امضا می باشد که می توانیم محل آن را در این قسمت انتخاب نماییم. سپس یک رمز عبور (پسورد) برای امضای خود انتخاب می کنیم. این رمز عبور در دفعات بعدی که بخواهیم از امضا استفاده نماییم لازم است پس آن را فراموش نکنید. رمز عبور شما حداقل باید دارای ۶ کاراکتر باشد. فیلد بعدی همان طور که از نام آن پیداست مربوط به نام مستعار می باشد. در این قسمت یک نام مستعار برای خود انتخاب می کنیم. فیلد بعدی آن نیز یک رمز عبور برای این نام مستعار می باشد که این رمز را هم انتخاب و آن را بخاطر می سپاریم. قسمت بعدی مربوط به مدت زمان ماندگاری امضای ما می باشد. این قسمت به صورت پیش فرض ۲۵ تنظیم شده است که به معنای ۲۵ سال می باشد. اطلاعات بعدی نیز مشخصات شخصی نظیر نام و نام خانوادگی و نام کمپانی و .. می باشد.

حال که امضای خود را ایجاد نمودیم، بر روی گزینه Next کلیک می کنیم و سپس با انتخاب محل ذخیره فایل APK برنامه خود، در حالی که Build Type بر روی Release است، گزینه ی Finish را میز را انتخاب می نماییم تا فایل ما صادر شود.



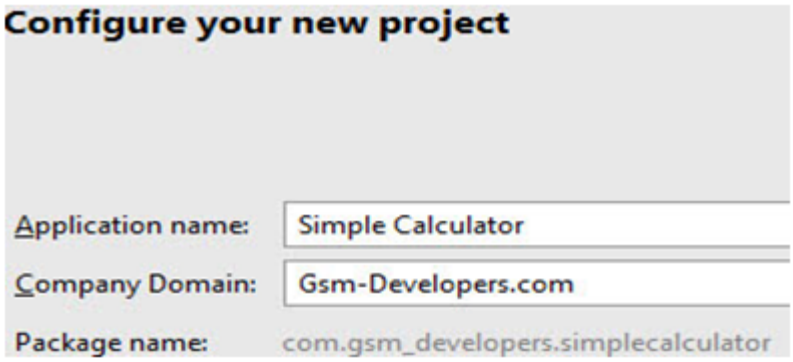
با زدن دکمه Finish ، برنامه شروع به ساخت فایل APK از روی کدهای نوشته شده می کند. اگر همه چیز خوب پیش برود یک پیغام به صورت زیر نشان داده می شود که حاکی از موفقیت آمیز بودن صدور فایل APK است.

حال می توانید این فایل را به دستگاه اندرویدی خود منتقل کرده و آن را نصب نمایید.

فصل سوم: ساخت ماشین حساب ساده

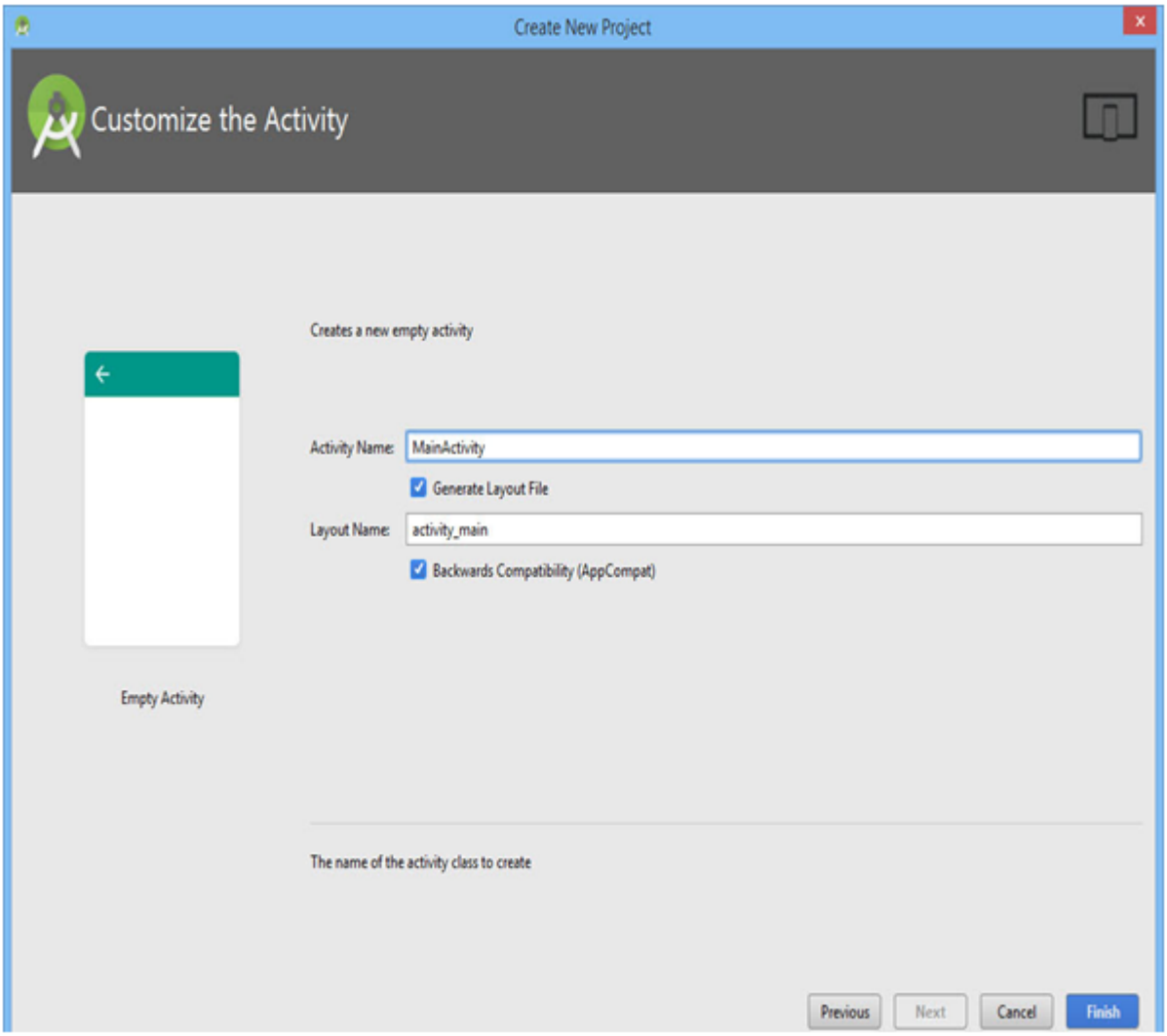
در این پروژه ما به سه شی نیاز داریم که عبارت اند از دکمه ها (Button)، تکست باکس‌ها (TextPlain) و تکست ویو (Text View).

یک پروژه جدید در اندروید استادیو می سازیم. از تب New گزینه New Project را انتخاب می کنیم. برای ایجاد پروژه جدید از ما نام پروژه و کمپانی خواسته می شود. پروژه را با نام Simple Calculator یعنی (Application Name : Simple Calculator) ایجاد می نماییم. همچنین نام کمپانی را مثلا Gsm-Developers.com می گذاریم.



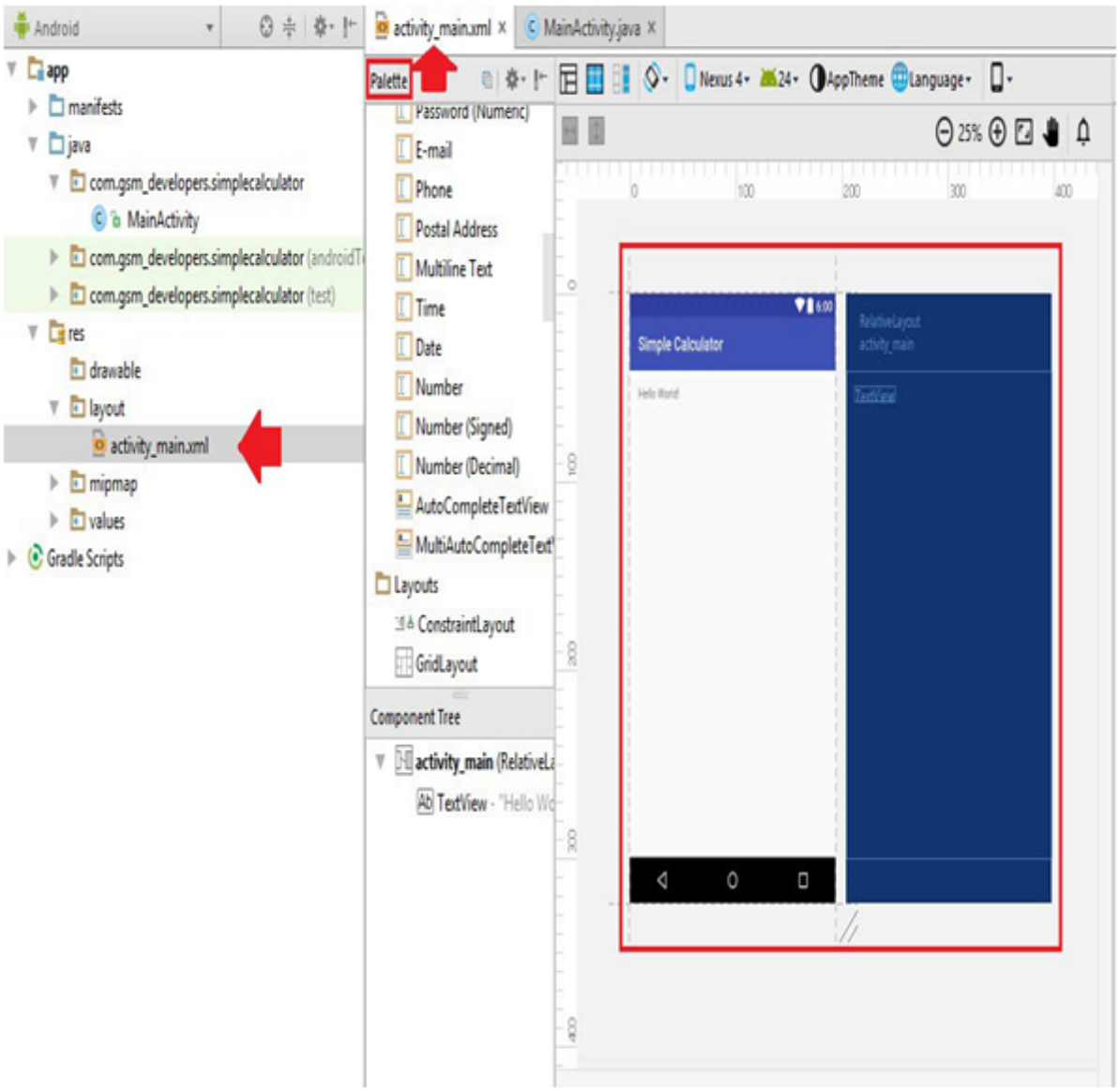
می توان محل ذخیره پروژه را نیز تغییر داد. بعد از زدن دکمه Next ، در مرحله بعد نسخه سیستم عامل اندروید را در قسمت Phone And Tablet انتخاب می کنیم. ما در اینجا نسخه اندروید ۵ (Lolipop) را انتخاب می کنیم. به یاد داشته باشید که نسخه های پایینتر اندروید، دستگاههای اندرویدی بیشتری را شامل می شوند ولی از امکانات پایین تری برخوردارند. در حالی که نسخه های بالاتر اندروید، دستگاههای اندرویدی کمتری را شامل می شوند ولی از امکانات بالاتری نیز برخوردارند. همچنین به این نکته توجه داشته باشید که اگر می خواهید این پروژه را در دستگاه اندرویدی خود استفاده نمایید از نسخه اندروید مطابق با سیستم عامل اندرویدی دستگاه خود استفاده نمایید.

دوباره دکمه Next را انتخاب می کنیم و یک Empty Activity (اکتیویتی خالی) برای پروژه انتخاب می کنیم. در قسمت نام اکتیویتی اصلی نیز می توانید همان نام پیش فرض را انتخاب نمایید.

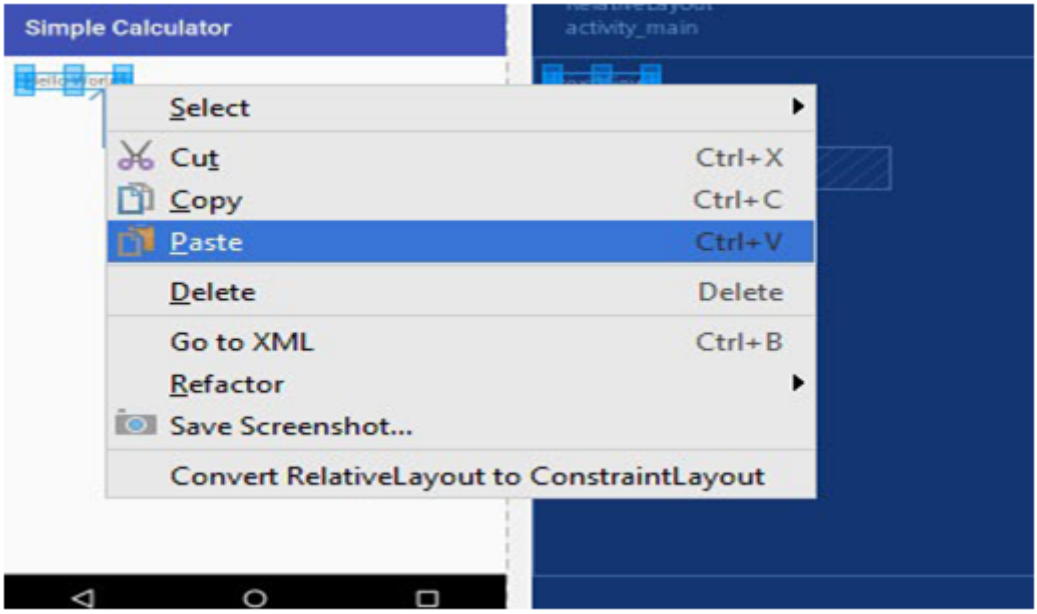


بر روی Finish کلیک کرده تا پروژه ما ایجاد شود.

حالا که پروژه ایجاد شده است، از قسمت سمت راست، به سراغ پوشه Res رفته و پس از باز کردن آن پوشه Layout را نیز باز می کنیم. حال فایل Activity_Main.xml را انتخاب نموده تا در قسمت سمت راست پنجره ای مانند عکس زیر باز شود.

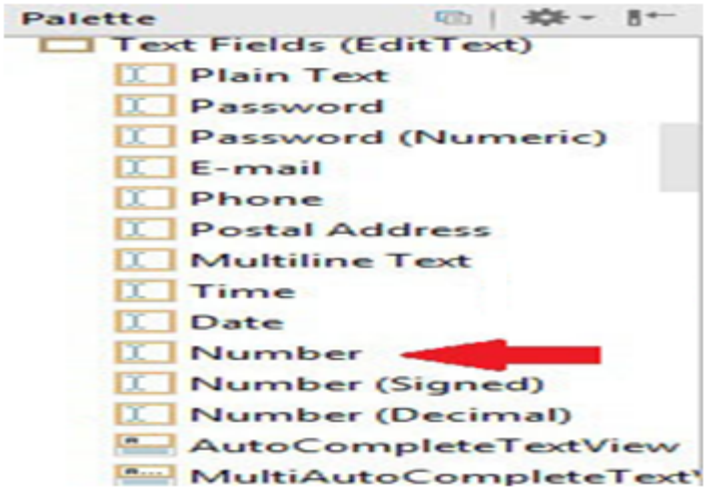


به پوسته برنامه رفته و نوشته "Hello world" را حذف می کنیم.

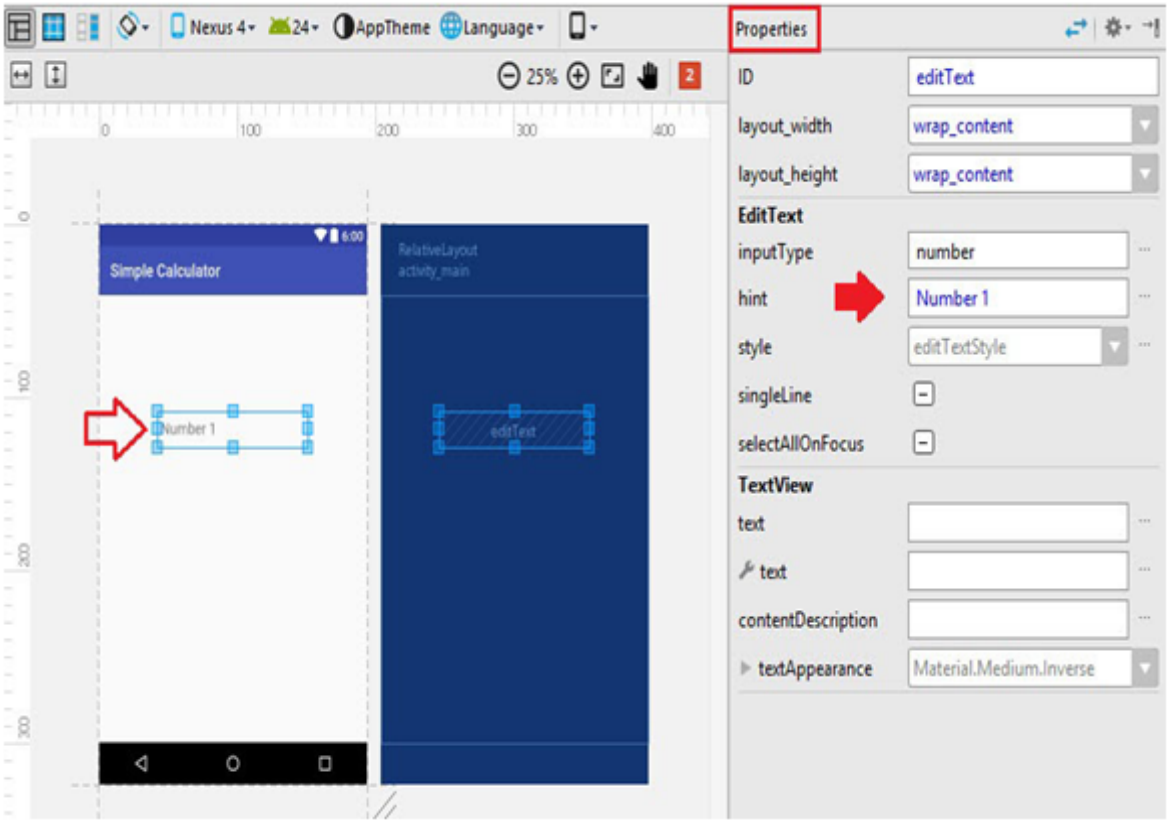


از قسمت Palette ، می توانیم دکمه هایی دلخواه را در برنامه بیاریم. در این قسمت ما می خواهیم که جایی برای وارد کردن عددهایمان داشته باشیم.

پس از قسمت Palette ، به سراغ Text Fileds ها می رویم Number را انتخاب می کنیم و به صورت Drag & Drop به پوسته برنامه می کشیم. یک Number دیگر را هم انتخاب کرده و به همین صورت به داخل پوسته می کشیم و در پایین Number قبلی قرار می دهیم.



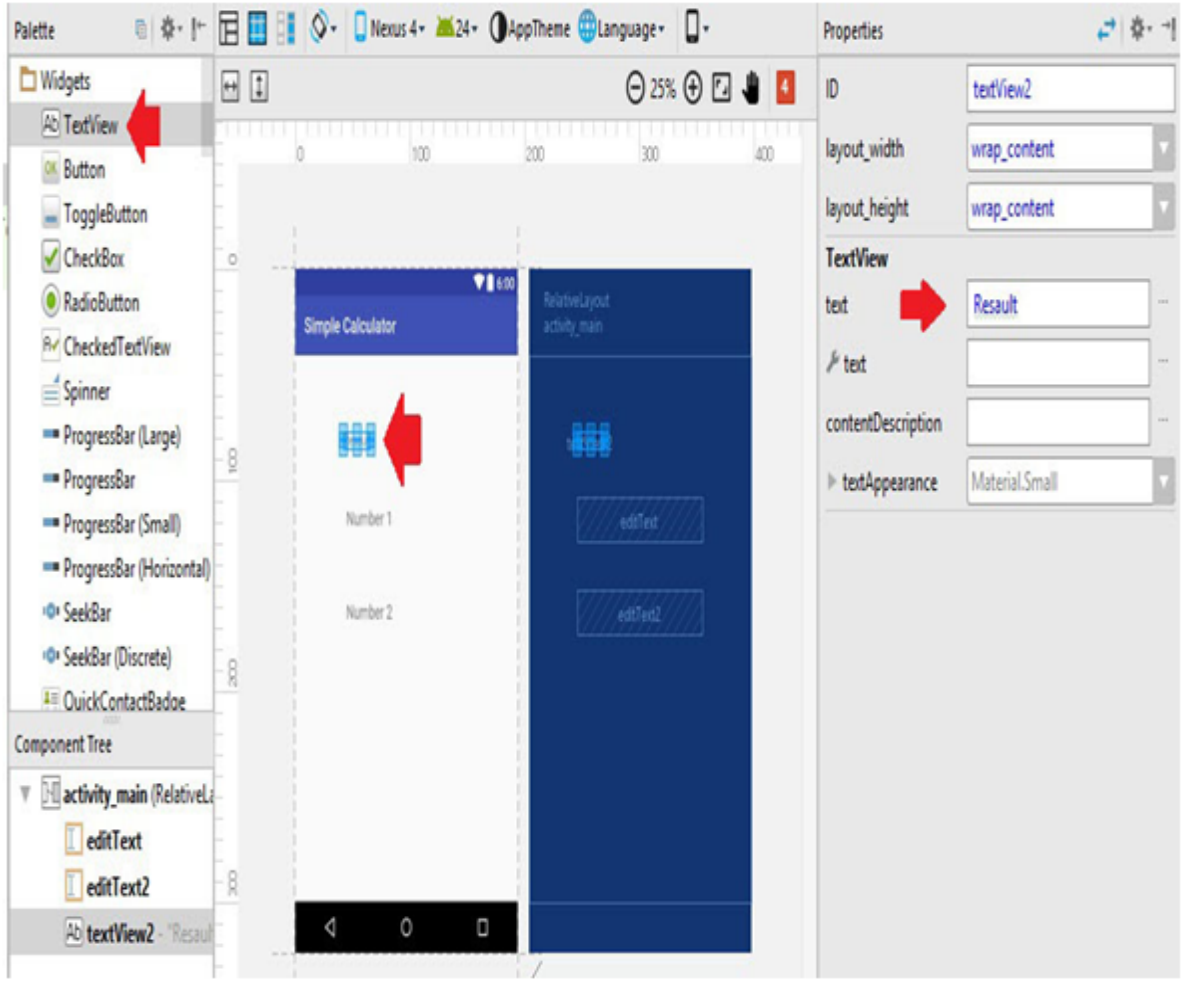
روی Number اولی که به درون پوسته آوردیم، دوبار کلیک کرده تا قسمت Properties در سمت راست برای آن باز شود. حال در قسمت Properties ، به سراغ Hint می رویم و مقدار Hint را برای آن، Number 1 قرار می دهیم.



برای Number دوم هم، Number 2 قرار می دهیم.

برای این که نتیجه را مشاهده کنیم احتیاج به یک Textview داریم. در قسمت Palette ، یک Widget از نوع TextView را به داخل پوسته برنامه Drag & Drop می کنیم.

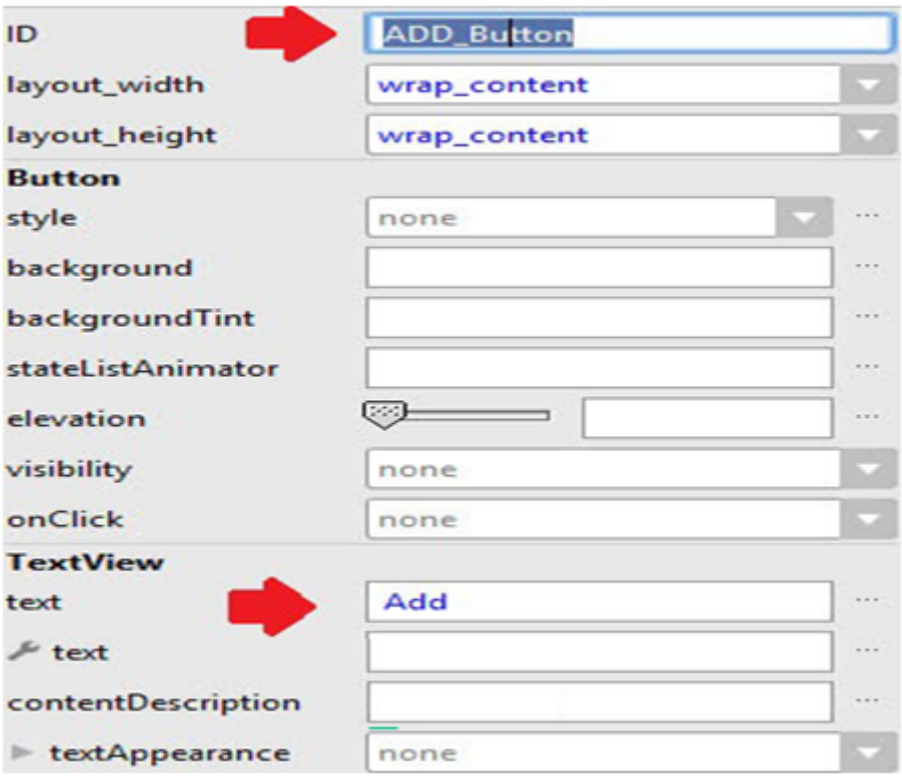
نام این TextView را در قسمت Properties و توسط Text به "Resualt" تغییر می دهیم.



خب به نظر می رسد به یک دکمه برای انجام شدن عملیات محاسبه نیاز داریم. یک Button از قسمت Widget ها انتخاب کرده و به داخل پوسته برنامه می کشیم.

طبق تصویر زیر نام این Button را Add می گذاریم تا با فشردن آن عملیات جمع را برای ما انجام دهد.

همچنین آیدی این Button را ADD_Button می دهیم.



به سراغ کد جاوای برنامه یعنی MainActivity.java رفته و در کلاس MainActivity ، کد زیر را می نویسیم.

کد زیر را در تابع onCreate اضافه کنید. دقت کنید که در پرانتز دوم این خط، یعنی آرگومان تابع findViewById ، حتما باید آی دی که برای سه ابزار قبلی بودند را انتخاب کنید.

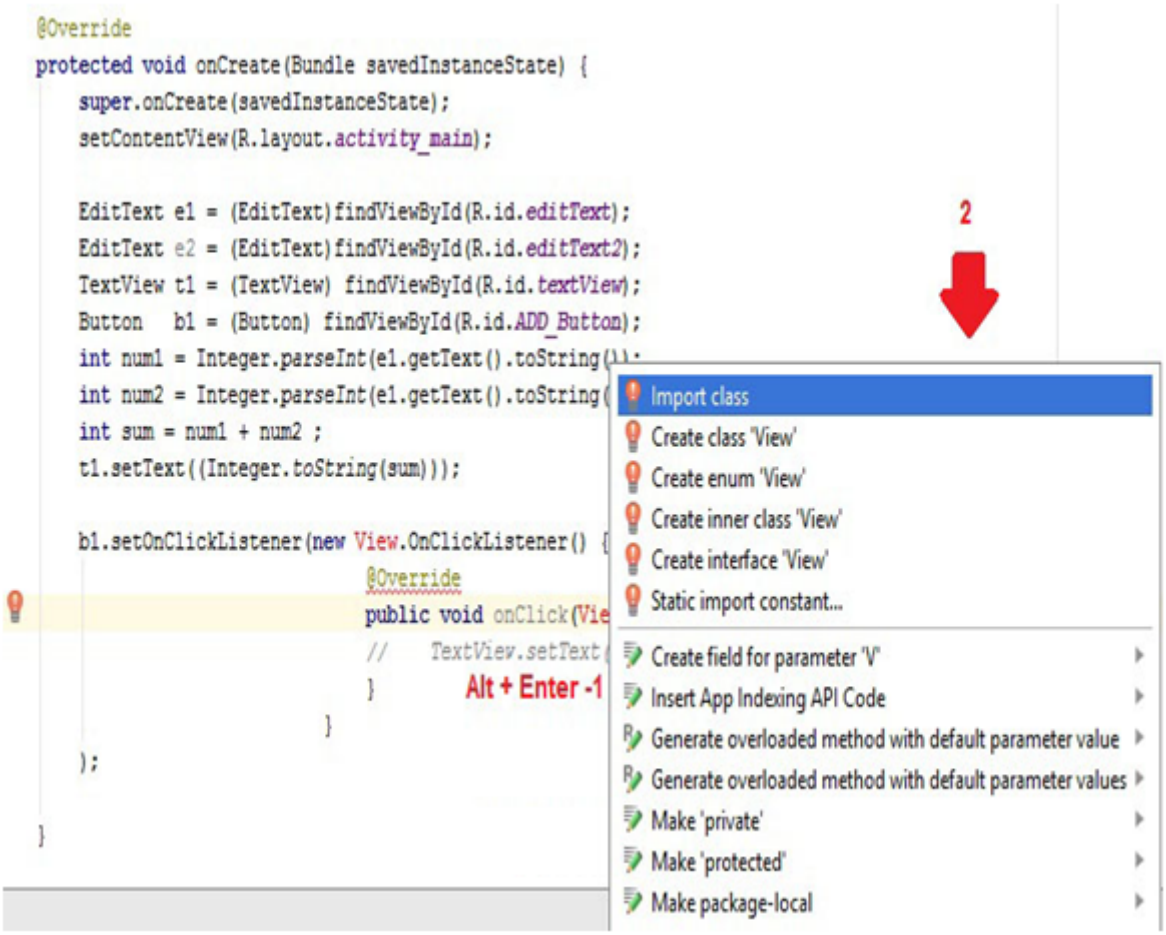
در اینجا ما یک متغیر b1 از نوع Button تعریف کردیم که در واقع همان دکمه با آیدی ADD_Button که قبلا اضافه نمودیم است.

```
Button   b1 = (Button) findViewById(R.id.ADD_Button);
```

سپس کد زیر را نیز برای این که بتوانیم با زدن دکمه Add بعدا عملیاتی انجام دهیم، پس از کد قبلی اضافه می کنیم. عملیاتی که بعدا اضافه خواهند شد و با زدن دکمه Button اجرا می شوند، در تابع onClick باید تعریف شوند.

```
b1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View V) {
    }
}
      );
```

نکته مهم) مشاهده می کنید که View در این قسمت، قرمز می باشد. این یعنی کتابخانه ما (در اینجا View) ، در کد جاوای ما وجود ندارد. برای اضافه شدن این کتابخانه وقتی که نشانگر موس بر روی View است، کلیدهای ترکیبی Alt + Enter را میزنیم تا به شکل زیر بتوانیم عمل Import Class را برای View انجام دهیم. با زدن Import class ، کتابخانه مورد نظر به صورت خودکار به پروژه اندرویدی ما و در کد کلاس جاوای ما اضافه خواهد شد.



همچنین می توانید به جای اینکار با اضافه کردن کد زیر در قسمت کتابخانه ها، این کتابخانه را اضافه نمایید.

```
import android.view.View;
```

این کد را در تابع onClick اضافه نمایید.

```
EditText e1 = (EditText)findViewById(R.id.editText);
EditText e2 = (EditText)findViewById(R.id.editText2);
int num1 = Integer.parseInt(e1.getText().toString());
int num2 = Integer.parseInt(e2.getText().toString());
int sum = num1 + num2 ;
TextView t1 = (TextView) findViewById(R.id.textView);
t1.setText((Integer.toString(sum)));
```

در این جا ما دو تکست باکس برای عدد اضافه کردیم که با آیدی های editText و editText2 شناخته می شود. همچنین آی دی تکست ویو ما textView می باشد.

خط اول و دوم این کد، متغیر هایی از نوع e1 و e2 برای تکست باکس هایی که ایجاد کردیم، به وجود می‌آورد.

خط سوم و چهارم، متغیر های num1 و num2 را از نوع int معرفی می کند، سپس این متغیرها را از تکست باکس های مربوطه توسط دو متغیر e1 و e2 می گیرد. برای تبدیل فرمت تکست باکس بهint ، ابتدا آن را به رشته تبدیل کرده تا با استفاده از تابع Integer.parseInt تبدیل به متغیر Int شود. سپس نتیجه جمع این دو متغیر، در متغیر sum ریخته می شود.

خطوط آخر نیز، برای نمایش نتیجه در تکست ویو مربوطه هستند.

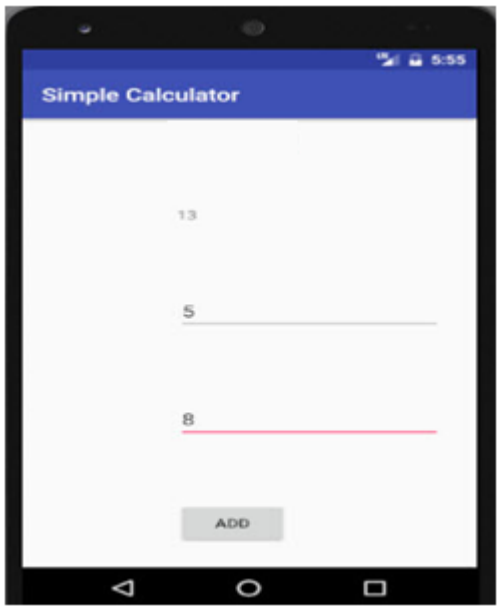
در اینجا نمایی کلی از کدهایی که در کلاس MainActivity تا به اینجا نوشتیم را مشاهده می کنید.

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

        Button   b1 = (Button) findViewById(R.id.ADD_Button);
        b1.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View V) {
                    EditText e1 =
(EditText)findViewById(R.id.editText);
                    EditText e2 =
(EditText)findViewById(R.id.editText2);
                    int num1 =
Integer.parseInt(e1.getText().toString());
                    int num2 =
Integer.parseInt(e2.getText().toString());
                    int sum = num1 + num2 ;
                    TextView t1 = (TextView)
findViewById(R.id.textView);
                    t1.setText((Integer.toString(sum)));
                }
            }
        );
    }
}
```

حال از قسمت Run ، با انتخاب Run app و سپس انتخاب شبیه ساز مناسب، برنامه را اجرا می کنیم.

خروجی باید مشابه نتیجه زیر باشد.



در نهایت با اضافه کردن دکمه های تفریق، ضرب و تقسیم می توانید سه عمل ریاضی دیگر را نیز به صفحه اضافه کرده و با نوشتن کدهای مربوط به آنها یک ماشین حساب ساده را ایجاد نمود.

فصل چهارم: ابزارهای تکمیلی در اندروید

- پیغام هشدار (Alert Dialog) در اندروید

پیغام هشدار (Alert Dialog) برای هشدار به کاربر به کار می رود. گاهی اوقات می خواهیم مثلا به کاربری که یک دکمه را فشار داده، هشداری بدهیم. این هشدار می تواند به دلایل متعددی باشد. مثلا بگوید "آیا مطمئن هستید که می خواهید ادامه دهید؟"

در این گونه موارد از Alert Dialog به عنوان هشدار و یا دادن اطلاعات به کاربر استفاده می شود.

AlertDialog در واقع یک کلاس است که این کلاس را در کدنویسی قسمت جاوای برنامه، فراخوانی می‌کنیم.

یک پروژه جدید با اسم GSM_AlertDialog ایجاد می کنم.

از قسمت پالت ها یک دکمه (Button) به لایه اپلیکیشن، اضافه می کنم.

کاری که می خواهیم انجام دهیم اینست که با زدن دکمه در برنامه، دیالوگ مورد نظر توسط AlertDialog، فراخوانی شود.

به کد جاوای برنامه می رویم، سپس کد زیر را به آن اضافه می کنیم.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
//کلاس اصلی جاوا
public class MainActivity extends AppCompatActivity {
//onCreate تابع
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
//تا اینجای کد را شما هم داشتید
//از اینجا به بعد کدی است که شما باید به کد اصلی خود اضافه کنید
//معرفی دکمه با استفاده از آیدی
        final Button button1 =  (Button)findViewById(R.id.button);
//این دستور در هنگام زدن دکمه اجرا شده و هر کدی که در آن بنویسیم اجرا می شود
        button1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
//کدی که می خواهیم وقتی دکمه را می زنیم اجرا شود
```

//این قسمت بعدا پر می شود

        }
    });
  }
}

در قسمت بالا در قسمتی که گفته شده است: "این قسمت بعدا پر می شود" کد زیر را اضافه کنید. کد زیر در هنگام زدن دکمه Button ، اجرا می شود.

```
AlertDialog.Builder dialog = new AlertDialog.Builder(MainActivity.this);
dialog.setTitle("Gsm Developers");
dialog.setMessage("Be Amoozesh Khoosh Amadid");
dialog.show();
```

کد بالا در قسمتی از برنامه اجرا می شود که هر چیزی که در آن بنویسیم، آن قسمت با زدن دکمه Button اجرا خواهد شد.

پس ابتدا AlertDialog مورد نظر را می سازیم. سپس اسمی برای دیالوگ (چیزی که می خواهیم نمایش داده شود) انتخاب می کنیم را dialog می گذاریم. در مراحل بعدی با نوشتن کلمه dialog و سپس زدن یک نقطه، می توان پیکربندی AlertDialog اصلی برنامه را تنظیم نمود. مثلا با کد dialog.setTitle آرگومانی به صورت رشته به این تابع می فرستیم که این آرگومان تعیین کننده عنوانی است که در AlertDialog نوشته خواهد شد. با نوشتن کد dialog.setMessage پیامی که می خواهیم در AlertDialog نمایش داده شود، را می نویسیم.

در آخر باید AlertDialog مورد نظر نمایش داده شود پس با کد dialog.show می توان AlertDialog را به اجرا در آورد.

تصویری از کد برنامه:

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

    final Button button1 =  (Button)findViewById(R.id.button);

    button1.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            AlertDialog.Builder dialog = new AlertDialog.Builder(MainActivity.this);
            dialog.setTitle("Gsm Developers");
            dialog.setMessage("به آموزش برنامه نویسی اندروید جلسه بیستم خوش آمدید");
            dialog.show();

        }
    });
}
```

برنامه را اجرا می کنیم.

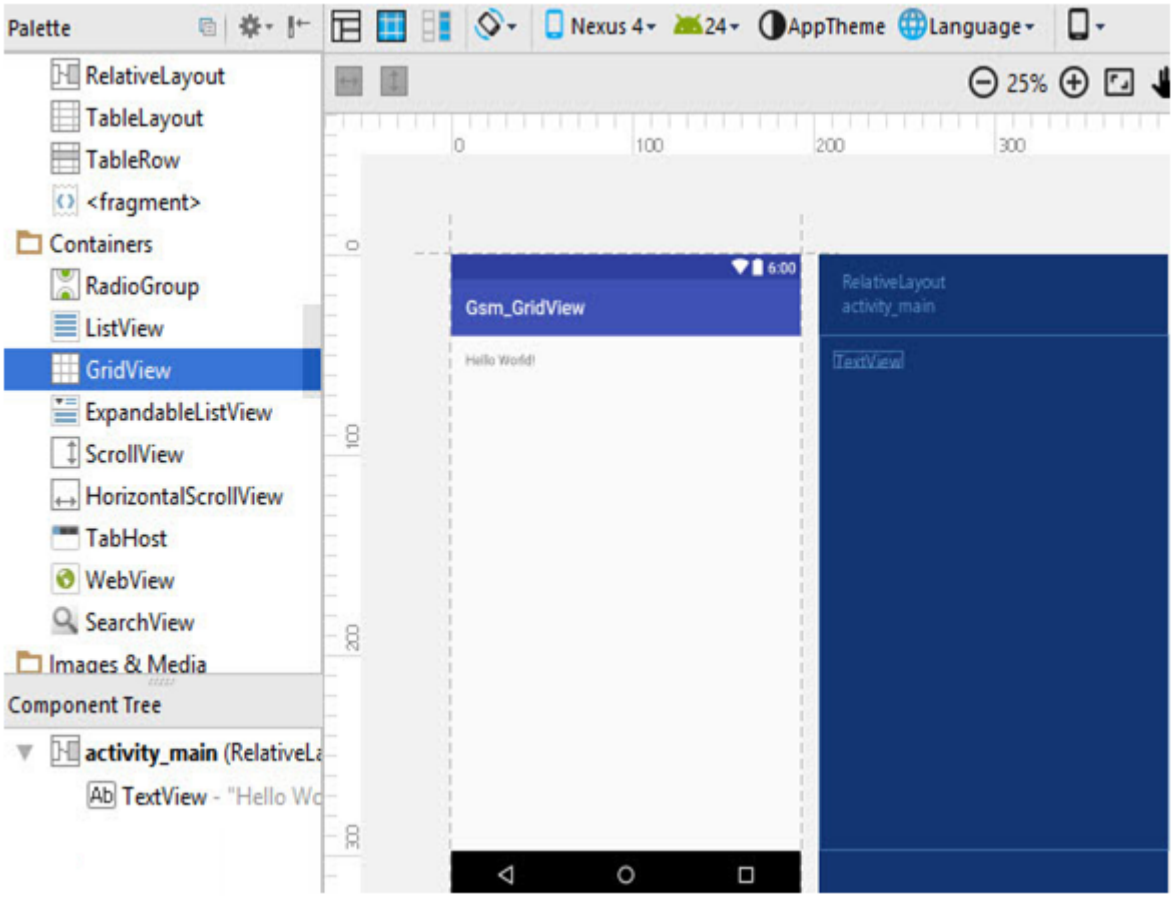
خروجی شما، پس از زدن دکمه Button به صورت زیر خواهد بود:

- گرید ویو (Grid View) در اندروید

گرید ویو ابزاری است که به لایه اضافه می شود و می توان در آن به صورت دو بعدی مطالب و اطلاعات را ارائه کرد. در لیست ویو، ما فقط می توانیم به صوت یک بعدی اطلاعات را نمایش دهیم.

گرید ویو در اندروید را می توان به یک قاب که در آن اطلاعات را نمایش می دهیم و یا مطالبی به آن اضافه می کنیم توصیف کرد.

از قسمت پالت ها به پوشه Composite می رویم. یک GridView را انتخاب کرده و به لایه برنامه می‌کشیم.



حال می خواهیم به ساختار و خواص گرید ویو در اندروید بپردازیم.

```
<GridView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="129dp" />
```

با دو خاصیت اول، قبلا آشنا شده اید. در اینجا نیز توضیحات را دوباره تکرار می کنیم.

خاصیت android:layout_width عرض گرید ویو را در لایه مشخص می کند.

مشاهده می نمایید که گرید ویو تشکیل شده است از عرض و ارتفاع که مقادیر Wrap_Content برای این گرید ویو تعریف شده است.

به جای این مقدار می توان از "fill_parent" و "match_parent" نیز استفاده نمود "fill_parent" عرض یا طول گرید ویو را به اندازه ی عرض یا طول برنامه تغییر می دهد.

خاصیت android_centerHorizontal که در اینجا True شده است، مشخص می کند که جایگاه گرید ویو ما در کجا واقع شده است. با True شدن این مقدار، نشان می دهد که جایگاه آن در مرکز می‌باشد. لازم به ذکر است که این مقادیر برای گرید ویو تعریف شده است و با تغییر جایگاه گرید ویو، مقادیر دیگری جای آن را خواهند گرفت.

خاصیت android:layout_marginTop ، فاصله این ابزار را قسمت بالایی لایه اپلیکیشن ما، تعیین می کند.

می خواهیم چند خاصیت جدید را به گرید ویو اضافه نماییم.

```
<GridView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="129dp"
//خاصیت های اضافه شده
    android:numColumns="2"
    android:columnWidth="70dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
    />
```

خاصیت android:numColumns تعداد ستون های گرید ویو را مشخص می کند. این خاصیت می‌تواند به صورت سفارشی باشد و یا به صورت Auto_fit تفاوت این دو حالت در اینست که می توانیم در حالت سفارشی، خودمان تعداد ستون ها را با عدد بدهیم که در اینجا دو ستون معرفی شده است. در حالت Auto_fit تعداد ستون ها را به صورت اتوماتیک خودش تنظیم می‌کند. خاصیت android:columnWidth نیز، فاصله ستون ها (پهنای هر ستون) از همدیگر را مشخص می کند. این فاصله می تواند با مقدار dp اندازه گیری می شود. مثلا در اینجا اگر ۷۰ dp به آن بدهیم، پهنای هر ستون به اندازه ۷۰ dp می شود. مقادیر دیگری نیز می تواند برای این خاصیت به کار رود مثلا in , mm , px

خاصیت android:stretchMode ، حالت کششی به گرید ویو ما می دهد. این حالت باعث می شود که گرید ویو خودش را با ستون ها هماهنگ بکند و مقدارش را از android:columnWidth می‌گیرد. خاصیت بعدی خاصیتی است که در همه ابزارها می توان از آن بهره گرفت. خاصیت android:gravitiy محل قرار گیری ابزار بر روی لایه را مشخص می کند. این خاصیت می تواند مقادیر Center,Top,Right,Left را داشته باشد.

حال به سراغ کد جاوای اصلی برنامه می رویم تا کدهای گرید ویو را در اندروید بنویسیم. می خواهیم مقادیری را برای نمایش در گرید ویو به کار ببریم. ابتدا یک رشته (String) با نام Gsm ایجاد کرده و مقادیری را در آن تعریف می کنیم. بعد از معرفی رشته، باید گرید ویویی که در ابتدای آموزش ایجاد کرده بودیم را در کد (MainActivity اکتیویتی اصلی برنامه) معرفی کنیم.

سپس یک آرایه آداپتر ایجاد نموده تا مقادیر را از رشته بگیرد و به گرید ویو بفرستد. اگر گرید ویو شما آیدی نداشت می توانید با اضافه کردن کد "android:id="@+id/gridView به آن آیدی بدهید. آیدی هر ابزار در واقع برای استفاده از آن در کد جاوا به کار می رود.

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

کد زیر تمام کارهایی که در بالا گفته شد را برای شما انجام می دهد. سه خط پایانی کد مربوط به آداپتر است.

```
public class MainActivity extends AppCompatActivity {
   String [] gsm = {"android" , "google" , "rom" , "mobile" , "windows
phone" ,  "apple" , "IOS"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

         GridView grid = (GridView) findViewById(R.id.gridView);

         ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
               android.R.layout.simple_list_item_1, gsm);
         grid.setAdapter(adapter);
   }
}
```

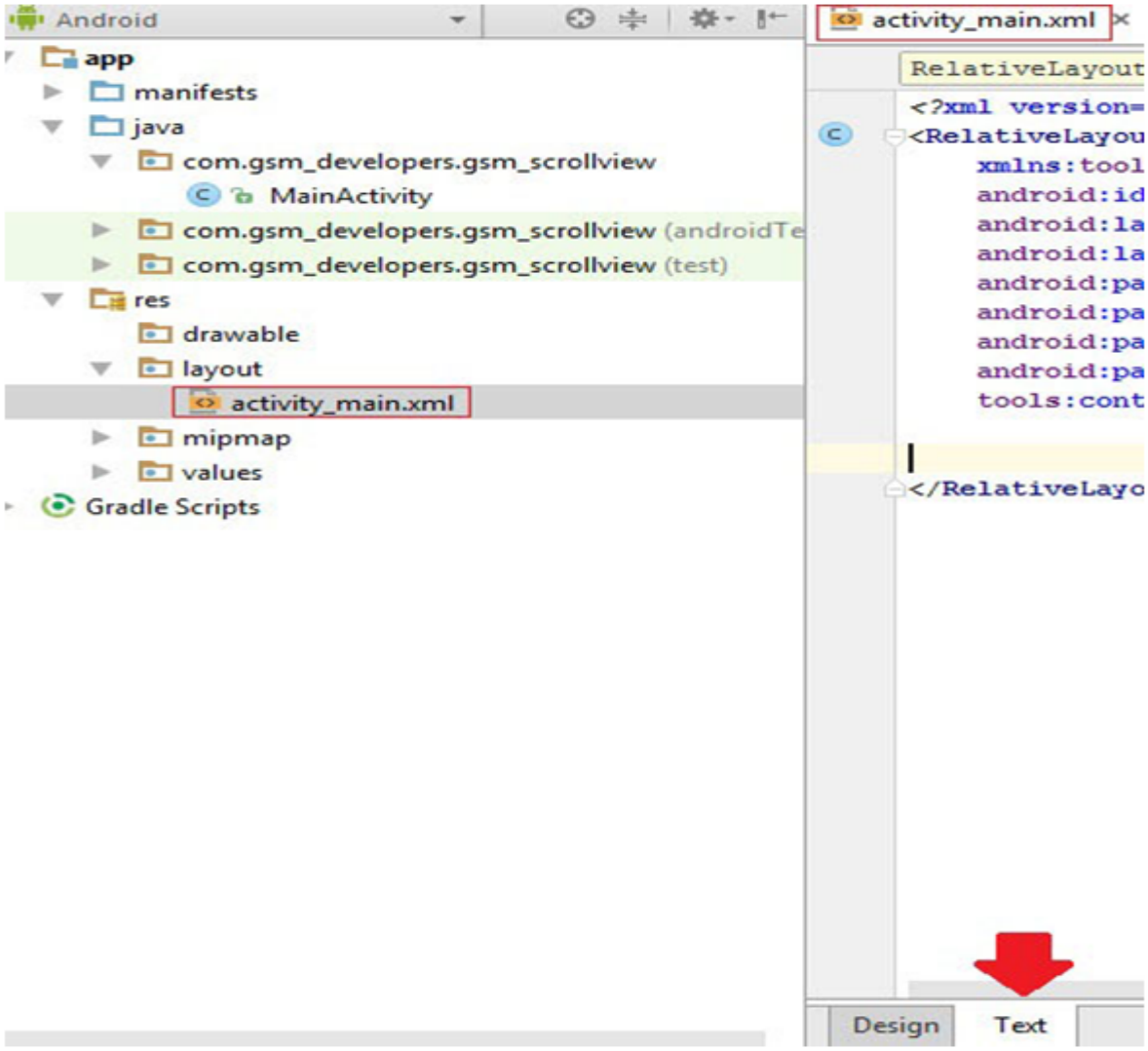
خروجی زیر را مشاهده خواهید کرد.

- اسکرول ویو (Scroll View) در اندروید

اسکرول ویو زمانی استفاده می شود که اطلاعات زیادی در صفحه موجود باشد پس نمی‌توان همه اطلاعات را در یک صفحه مشاهده کرد. باید بتوان با بالا پایین کردن صفحه تمام اطلاعات را بتوان مشاهده نمود. ما دو نوع اسکرول ویو عمودی و افقی در اندروید داریم که به دلیل کاربرد بیشتر نوع عمودی به آن می‌پردازیم.

یک پروژه با نام Gsm_ScrollView ایجاد می کنیم.

ایجاد یک اسکرول ویو در اندروید خیلی آسان است. در ابتدا به قسمت لایه اپلیکیشن یا همان Activity_main.xml در قسمت Res می رویم. سپس به قسمت Text لایه می رویم.



سپس یک تگ Scroll View باز می کنیم و عرض و ارتفاع آن را "fill_parent" می گذاریم.

مقدار fill_parent باعث می شود که عرض و ارتفاع لایه توسط اسکرول ویو پر شود.

کد:

```
<ScrollView
android:layout_width="fill_parent"
android:layout_height="fill_parent"></ScrollView>
```

حال، یک لایه جدید بر روی اسکرول ویو اضافه می کنیم تا عناصر بر روی لایه جدید اضافه بشوند. برای انجام این کار، پس از نوشتن تگ ScrollView ، تگ LinearLayout را اضافه می کنیم. بدین صورت یک لایه جدید اضافه می شود. (دقت کنید که حتما بین دو تگ شروع و پایان ScrollView باید تگ LinearLayout اضافه شود). عرض و ارتفاع این لایه را برابر با مقدار fill_parent قرار می دهیم. این کار باعث می شود که لایه جدید با اسکرول ویو که نوشتیم، در پر کردن صفحه لایه هماهنگ شود.

```
<ScrollView
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical">
   </LinearLayout>
</ScrollView>
```

اگر دقت کنید متوجه می شوید که خاصیت جدیدی در کد بالا اضافه شده است. این خاصیت باید حتما به لایه جدید اضافه شود تا لایه ما در حالت عموی قرار بگیرد. این خاصیت android:orientation=”vertical“ است. اگر این خاصیت وجود نداشته باشد عناصر اسکرول ویو به هم می ریزد.

حال می توانید هر ابزار و خاصیتی را به لایه جدید اضافه نمایید.

کد ScrollView ما برای لایه اپلیکیشن در زیر آمده است.

ما در کد خودمان برای تست و آزمایش و سادگی نشان دادن به شما ۱۲ تکست ویو ایجاد کرده ایم که ارتفاع هر کدام ۸۰ dp باشد تا بتوانید صفحه را به راحتی بالا و پایین ببرید.

برای این که دید بهتری نسبت به آن چه گفته شد، داشته باشید، تمامی کد اصلی برای قسمت لایه اپلیکیشن ما در زیر آورده شده است:

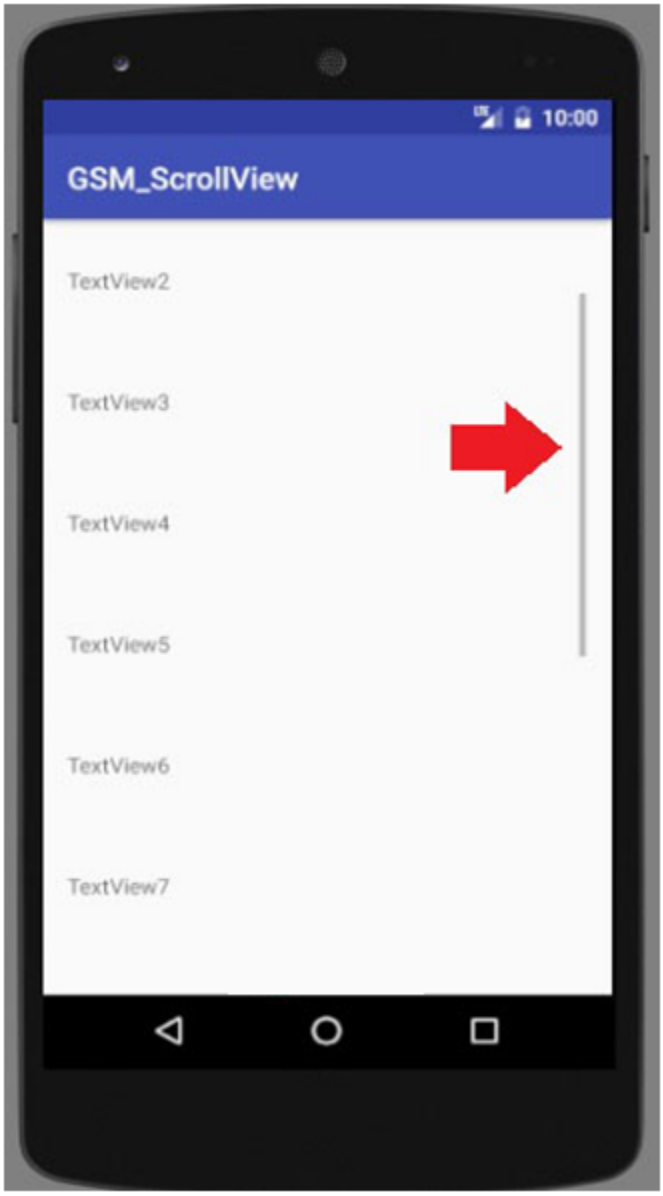
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
```

```
    tools:context="com.gsm_developers.gsm_scrollview.MainActivity">

<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
        <TextView
            android:text="TextView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView6"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView8"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView10"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView11"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
        <TextView
            android:text="TextView12"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:height="80dp"/>
    </LinearLayout>
</ScrollView>
</RelativeLayout>
```

برنامه را در شبیه ساز، اجرا می کنیم.

خروجی ما به شکل زیر است.



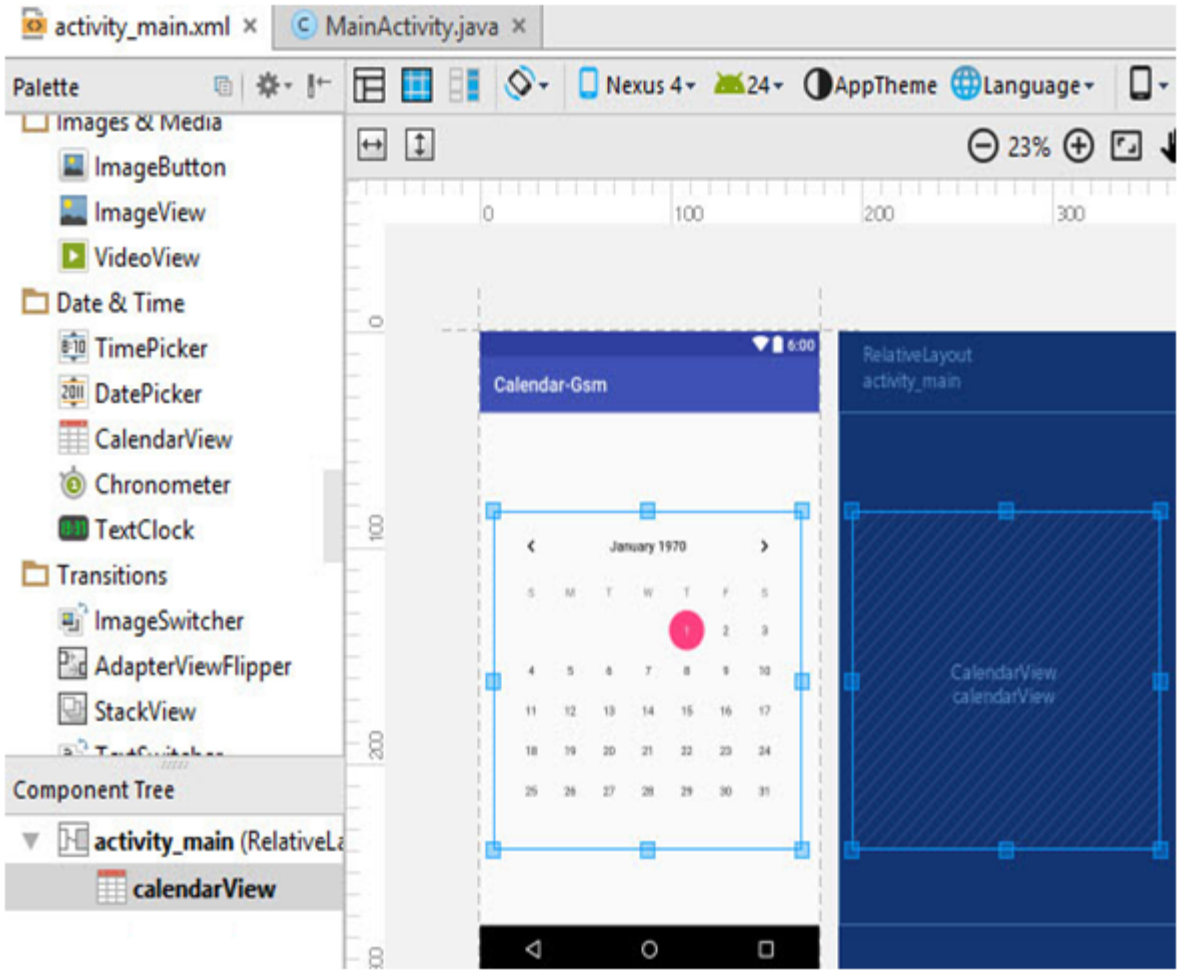
فصل پنجم : تقویم (CalendarView) در اندروید

برای استفاده از تقویم در این آموزش، تنها لازم است که تقویم را بر روی لایه ایجاد و سپس به قسمت جاوای برنامه معرفی کنیم.

مثالی که زده می شود، اینست که رویدادی به نام onDateChangeListener را به CalendarView تعریف می کنیم. وقتی بر روی تاریخ مشخصی کلیک شود، این رویداد کاری را برای ما انجام می دهد. (نشان دادن روز و ماه و سال)

یک پروژه جدید به نام Calendar-Gsm را ایجاد می کنیم.

حال به سراغ activity_main.xml می رویم و از قسمتPalette ها در پوشهDate & Time ، یک CalendarView را به لایه اپلیکیشن برنامه می کشیم.




```
<CalendarView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/calendarView"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="58dp" />
```

به سراغ MainActivity.java رفته و مراحل زیر را انجام می دهیم:

CalendarView را به کد جاوای برنامه، معرفی و نام گذاری می کنیم.

به CalendarView یک رویداد onDateChangeListener اضافه می کنیم.

در رویداد onDateChangeListener ، یک تابع با نام onSelectedDayChange ایجاد می‌کنیم تا برای ما سال و ماه و روز با استفاده از Toast به نمایش در بیاورد.

تمامی این مراحل را در چرخه (تابعonCreate) انجام می دهیم.

مرحله اول:

معرفی و نام گذاری CalendarView درMainActivity.java

```
CalendarView Taghvim = (CalendarView) findViewById(R.id.calendarView);
```

مرحله دوم:

ایجاد رویداد setOnDateChangeListener

```
Taghvim.setOnDateChangeListener(new OnDateChangeListener() {
```

مرحله سوم:

معرفی تابع onSelectedDayChange به برنامه برای نشان دادن سال و ماه و روز

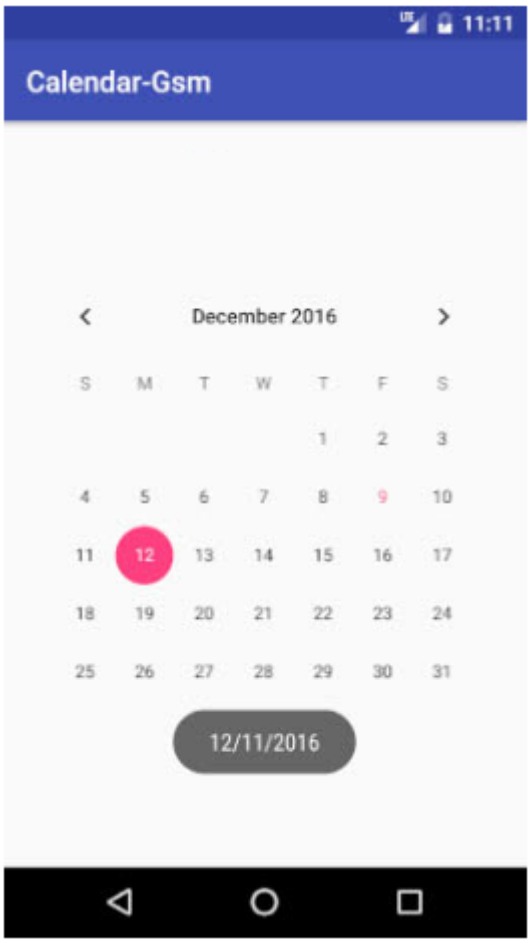
```
Toast.makeText(getApplicationContext(),dayOfMonth + "/" + month + "/" +
year, Toast.LENGTH_LONG).show();
```

کد کامل MainActivity

```
package com.gsm_developers.calendar_gsm;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.CalendarView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

      CalendarView Taghvim = (CalendarView) findViewById(R.id.calendarView);

      Taghvim.setOnDateChangeListener(new
CalendarView.OnDateChangeListener() {

         @Override
```

```
        public void onSelectedDayChange(CalendarView view, int year, int
month, int dayOfMonth) {

            Toast.makeText(getApplicationContext(),dayOfMonth + "/" + month +
"/" + year, Toast.LENGTH_LONG).show();
        }
    });
  }
}
```

وقتی برنامه را اجرا می کنیم، خروجی برنامه به شکل زیر است. با زدن هر تاریخی، سال و ماه و روز به کاربر نمایش داده می شود.



فصل ششم: پخش صوت در اپلیکیشن اندروید

یک پروژه جدید به نام Sound-Gsm ایجاد کردم.

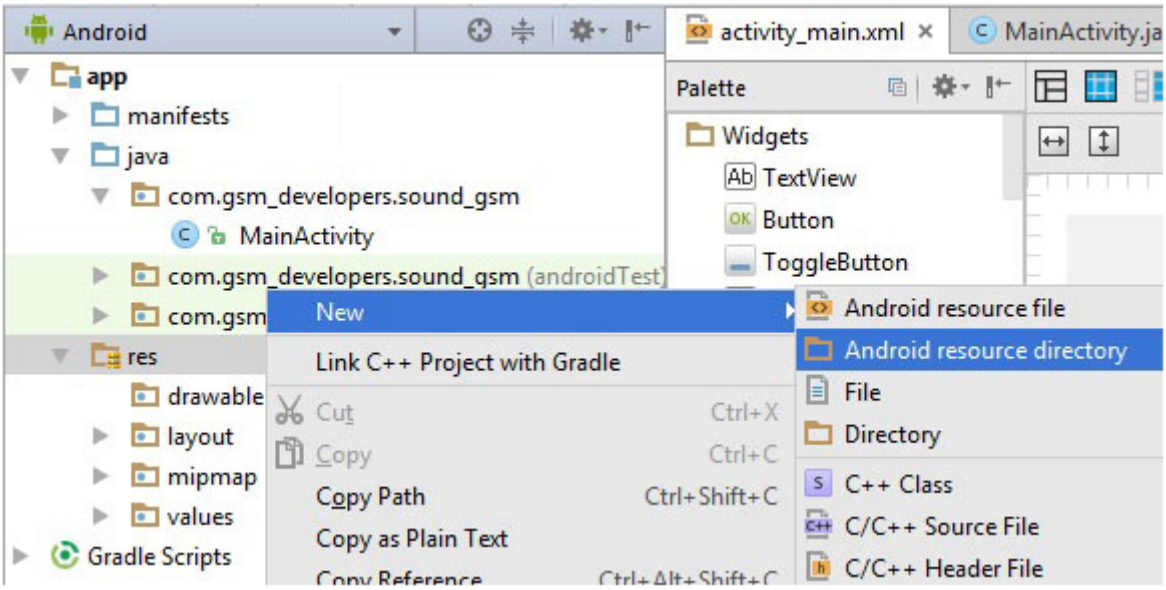
حال به قسمت لایه برنامه رفته و از Palette ، یک ویجت از نوع TextView را به درون لایه می کشم.

اسم این تکست ویو (TextView) را "Pakhshe Music" می گذاریم.

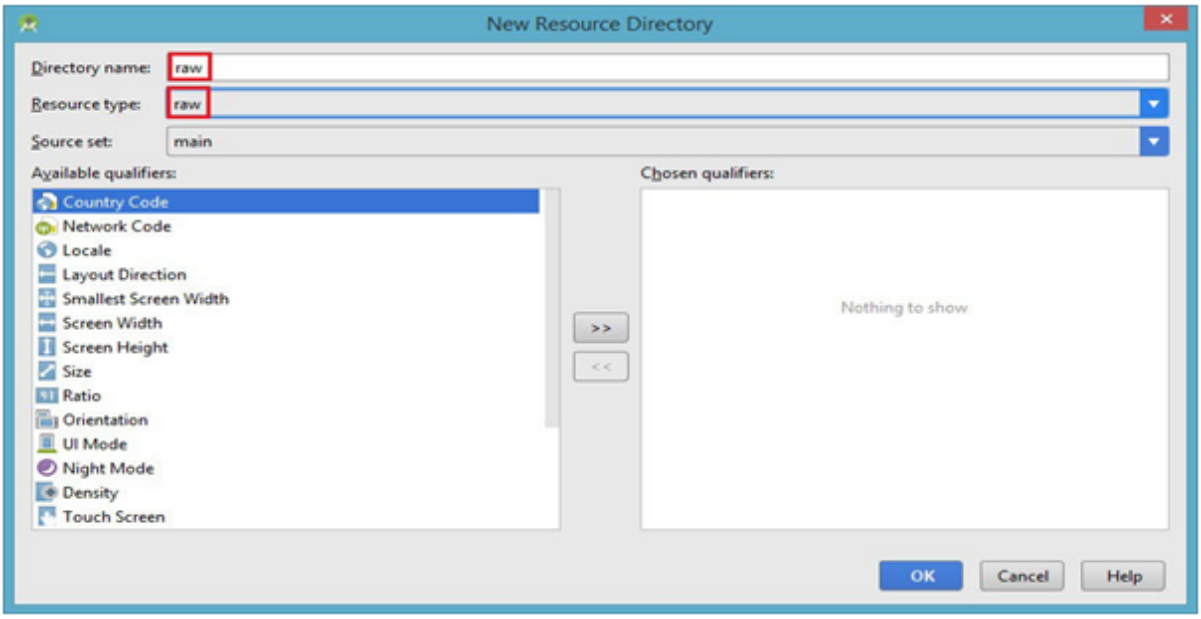
توجه داشته باشید که در برنامه ما، موزیک به صورت خودکار در پس زمینه اپلیکیشنی که نوشته ایم پخش می شود و ما در این بخش از آموزش، دخالتی برای پخش موسیقی نمی کنیم.

قبلا گفته شد که پوشه Res پوشه منابع ما است و عکس ها و فایل های ما برای استفاده در برنامه در این پوشه قرار می گیرد. پس باید یک پوشه جدید به نام Raw در پوشه Res ایجاد کنیم.

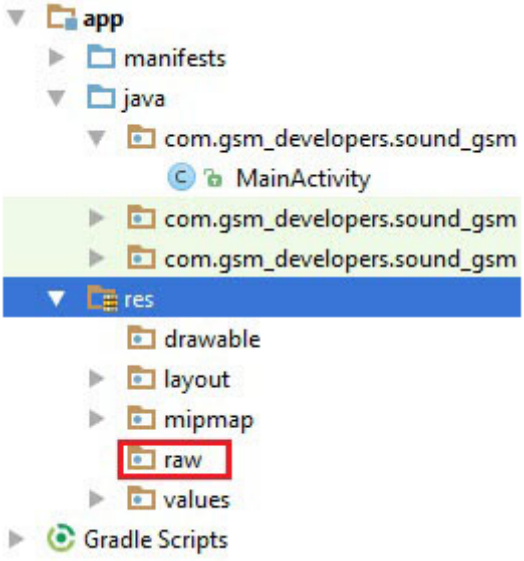
برای این کار طبق عکس، بر روی پوشه Res کلیک راست کرده و سپس از قسمت New ، بر روی Android Resource Directory کلیک می کنیم.



با باز شدن پنجره ی جدید طبق عکس، Raw را انتخاب کرده و سپس بر روی Ok کلیک می کنیم.



می بینید که پوشه raw در دایرکتوری Res ایجاد شده است.



حال فایل موسیقی خود را به این پوشه اضافه می کنیم.

من فایل موسیقی خودم را در پوشه raw و از طریق explorer خود ویندوز انجام دادم. پوشه raw را می توانید در این آدرس پیدا کنید:

AndroidStudioProjects\Sound-Gsm\app\src\main\res\raw

اسم آهنگ من air.mp3 است. توجه کنید که نام فایل و نام آن در کدنویسی اندروید را با حروف کوچک بنویسید. چون در برنامه نویسی اندروید فایل ها در منابع نباید با حروف بزرگ باشند.

به قسمت کدنویسی جاوا یعنی MainActivity.java رفته و کد زیر را در آن می نویسیم.

```
MediaPlayer mPlayer = MediaPlayer.create(MainActivity.this, R.raw.Air);
mPlayer.start();
```

برای نوشتن کد MediaPlayer کلاس زیر را به پروژه خود اضافه نمایید. و یا بر روی MediaPlayer زده و Alt + Enter را بزنید سپس Import Class را بزنید.

```
import android.media.MediaPlayer;
```

اگر برنامه را اجرا نمایید موسیقی با اجرای برنامه پخش می شود. ولی اگر برنامه را ببندید موسیقی همچنان در حال پخش خواهد بود. راه حل چیست؟

این مورد در حوزه چرخه حیات اکتیویتی ها تعریف می شود. در اینجا چون ما کد پخش موسیقی را در اکتیویتی onCreate نوشته ایم پس موسیقی هیچ گاه تا Stop شدن برنامه قطع نمی شود.

برای حل این مشکل در اکتیویتی onPause و یا onStop باید موسیقی را قطع کنیم تا وقتی از برنامه خارج شدیم موسیقی قطع شود.

کد زیر را به برنامه اضافه می کنیم تا موسیقی ما به حالت Pause برود.

همچنین با نوشتن mPlayer.Stop نیز می توان موسیقی را Stop نمود نه Pause!

```
protected void onPause() {
    super.onPause();
```

```
    mPlayer.pause();
  }
```

از آنجایی که متغیر mPlayer به صورت سراسری یا Global معرفی نشده است در تابع onPause شناخته نشده است. برای همین باید به صورت سراسری معرفی شود که با نوشتن آن خارج از تابع onCreate این اتفاق می افتد. کد زیر کد پایانی برنامه ما است.

کد کامل برنامه:

```
import android.media.MediaPlayer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
//تعریف متغیر به صورت سراسری
  public  MediaPlayer mPlayer;
//اکتیویتی onCreate
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
//تعریف متغیر برای موسیقی ما که این نام اضافه کرده بودیم
     mPlayer = MediaPlayer.create(MainActivity.this, R.raw.air);
//استارت موسیقی
     mPlayer.start();

  }
//اکتیویتی onPause
  protected void onPause() {
    super.onPause();
//پایز کردن موسیقی
    mPlayer.pause();
  }

}
```

می توانید با اضافه کردن اکتیویتی های onResume و یا onStop و ... با پخش موسیقی در این برنامه بازی کنید. همچنین می توانید کاری کنید که با کلیک بر روی دکمه، موسیقی پخش و یا قطع شود.

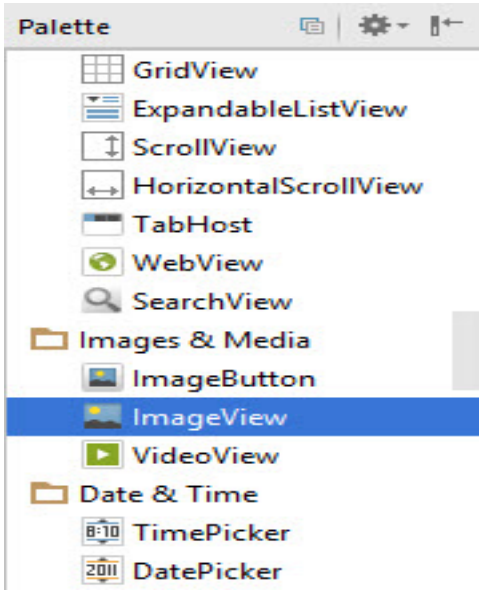
اگر برنامه را اجرا کنید می بینید که با آغاز شدن برنامه، موسیقی شروع به پخش کرده و سپس با خارج شدن از برنامه، موسیقی قطع می شود.

فصل هفتم: کار با دوربین در اندروید

گاهی پیش می آید که در برنامه اندرویدی نیاز به استفاده از دوربین داشته باشیم. اگر تا به حال از برنامه هایی مثل اینستاگرام و برنامه های ویرایش تصاویر استفاده کرده باشید، متوجه شدید که گزینه ای برای استفاده از دوربین دستگاه پلتفرمی شما وجود دارد.

یک پروژه جدید به نام Camera-Gsm ایجاد می کنیم. از قسمت Palette ها یک دکمه Button بر روی لایه می کشیم. و اسمش را Camera می گذاریم.

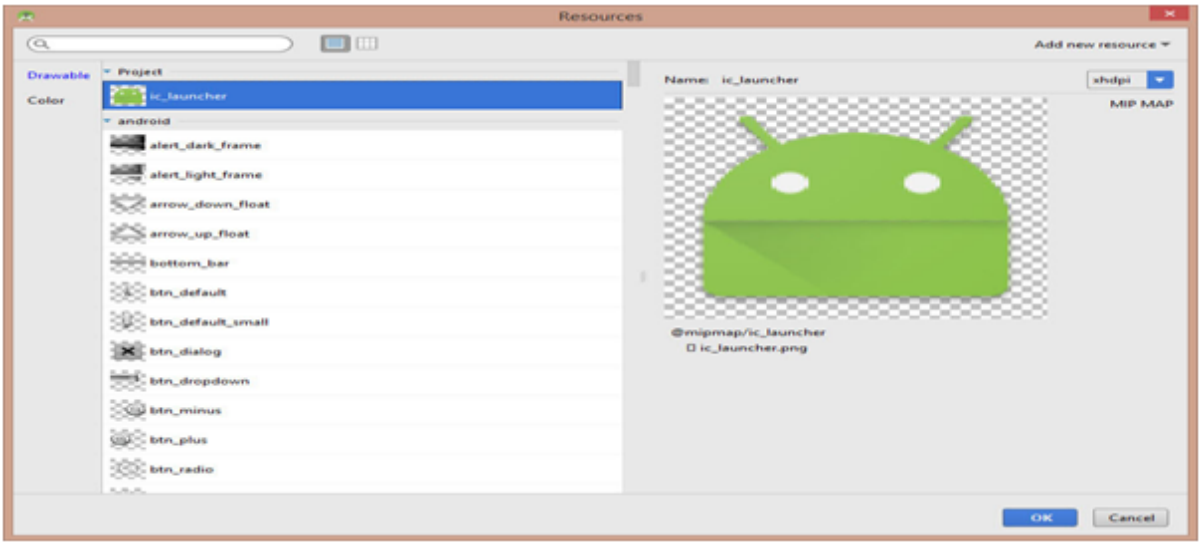
ابزار جدیدی که در این قسمت با آن کار می کنیم، ImageView می باشد. از قسمت Images & Media در جعبه ابزار، یک ImageView را به لایه اپلیکیشن مورد نظر می کشیم.



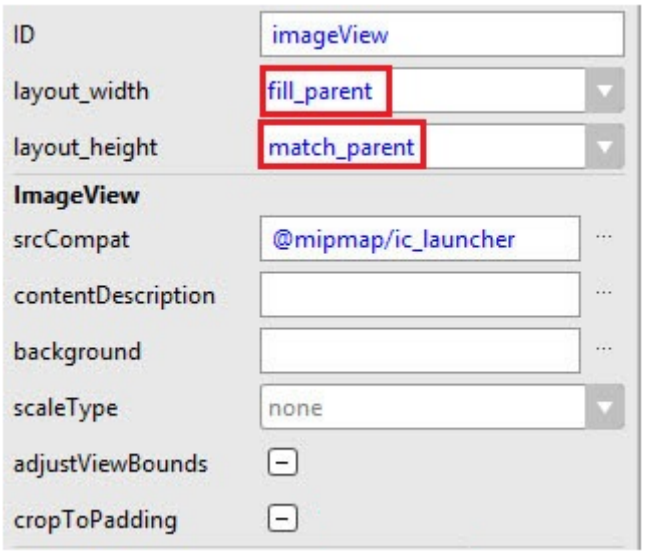
میخواهیم تصویری که دوربین می گیرد در این ImageView نمایش داده شود.

زمانی که ImageView را به سمت لایه اپلیکیشن خود، می کشیم، یک پنجره به شکل زیر باز می شود. این پنجره در واقع آیکونی را به صورت پیش فرض بر روی ImageView ایجاد می کند.

از آنجایی که این موضوع آموزشی است، ما در اینجا ic_Launcher را انتخاب می کنیم تا این شکلک اندروید، بر روی ImageView ایجاد شود که بتوانیم محل آن را تشخیص دهیم.



وقتی ImageView را به لایه اپلیکیشن می کشیم و پس از Ok کردن پنجره بالا، بر روی ImageView می زنیم. سمت راست پنجره Android Studio و از قسمت Properties دو مقدار layout_width و layout_height را که برای تعیین پهنا و ارتفاع ImageView ما می باشد، به شکل زیر تغییر می دهم. مقدار layout_width را fill_parent می گذارم تا عرض لایه اپلیکیشن را به صورت کامل پر کند. سپس مقدار layout_height را match_parent میگذارم تا ارتفاع ImageView به صورت اتوماتیک پر شود.



کد Button و ImageView:

```
<Button
    android:text="Camera"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="14dp"
    android:id="@+id/button"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />

<ImageView
    android:layout_height="match_parent"
    app:srcCompat="@mipmap/ic_launcher"
    android:id="@+id/imageView"
    android:layout_width="fill_parent"
    android:layout_above="@+id/button"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="54dp" />
```

کار ما با طراحی لایه اپلیکیشن تمام است. به سراغ فایل MainActivity.java می رویم تا کدهای جاوای برنامه را درآن می نویسیم.

یک مقدار را به صورت Global یعنی قبل از تابع ()onCreate تعریف میکنم تا هر زمانی که اکتیویتی ما بالا آمد، برنامه به صورت خودکار بالا بیاید.

```
private static final int Cam_Req = 1;
```

سپس یک متغیر image از نوع ImageView را نیز به همین صورت یعنی گلوبال تعریف می کنم تا بتوانم در قسمت های مختلف برنامه از آن استفاده کنم.

```
ImageView image;
```

برای استفاده از ImageView باید کلاس زیر را به برنامه Import کنیم. (کدهای بالای قسمت جاوای برنامه) MainActivity.java یا اینکه می توانید با کلیدهای ترکیبی Alt + Enter بر روی کلمه ImageView زمانی که قرمز است کلاس را Import کنید.

```
import android.widget.ImageView;
```

در مرحله بعدی باید Button و ImageView را با استفاده از ID به برنامه منتقل کنم.

```
Button photoButton = (Button) findViewById(R.id.button);
image = (ImageView) findViewById(R.id.imageView);
```

سپس یک رویداد برای دکمه Button ایجاد می کنم تا هر وقت بر روی آن کلیک شد، اپلیکیشن به قسمت دوربین برود.

```
 photoButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
           Intent cameraIntent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
           startActivityForResult(cameraIntent, Cam_Req);

        }
    });
```

در کد بالا در داخل تابع ()onCreate ، با استفاده ازIntent ، اطلاعاتی را از برنامه می گیرد و به MediaStore می فرستد. سپس با استفاده از دستور StartActivityForResult ، اینتنت را به دوربین ارجاع می دهیم. این دستور باعث فعال شدن دوربین می شود.

پس تا به اینجا کار با دوربین تقریبا تمام شده است و دوربین آماده تصویر برداری است. ولی هنوز کدی برای ذخیره تصویر بر روی ImageView ننوشته ایم.

حال خارج از تابع ()onCreate کد زیر را می نویسم.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == Cam_Req) {
       Bitmap photo = (Bitmap) data.getExtras().get("data");
```

```
        image.setImageBitmap(photo);

    }
  }
```

این کد باعث می شود که اطلاعاتی که از دوربین گرفته شده است، با استفاده از کد Bitmap به تصویر تبدیل شود و با استفاده از کد زیر به ImageView فرستاده شود.

```
image.setImageBitmap(photo);
```

کد کامل MainActivity.java:

```
package com.gsm_developers.camera_gsm;

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

public class MainActivity extends AppCompatActivity {

   private static final int Cam_Req = 1;

   ImageView image;

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

      Button photoButton = (Button) findViewById(R.id.button);
      image = (ImageView) findViewById(R.id.imageView);

      photoButton.setOnClickListener(new View.OnClickListener() {

         @Override
         public void onClick(View v) {
            Intent cameraIntent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, Cam_Req);
```

```
        }
    });

}

protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == Cam_Req) {
        Bitmap photo = (Bitmap) data.getExtras().get("data");
        image.setImageBitmap(photo);

    }
}
}
```

اپلیکیشن را اجرا می کنیم. مشاهده می کنید که در مرحله اول یک دکمه به اسم Camera و یک ImageView با همان شکلک اندرویدی که انتخاب کرده بودیم، در اپلیکیشن ما وجود دارد.

فصل هشتم: ذخیره اطلاعات با SharedPreferences در اندروید

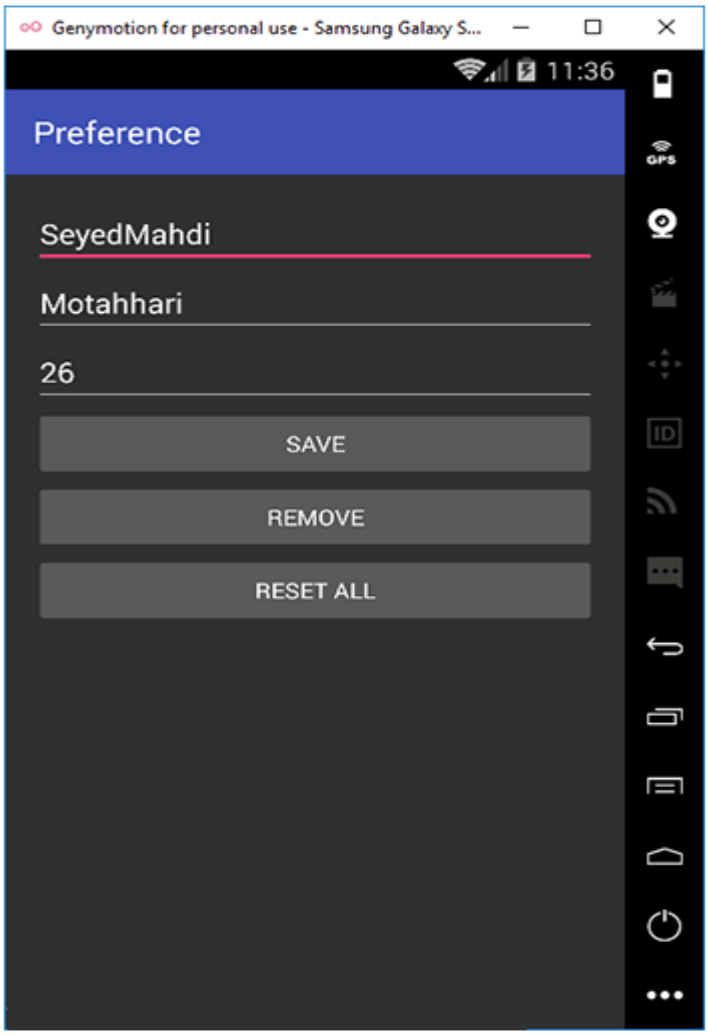
برای ذخیره تنظیمات اپلیکیشن (فونت و سایز برنامه، حالت شب/روز، رنگ پس زمینه و ...)، ثبت اطلاعات کاربر، نحوه نمایش و چیدمان لیست ها، فعال یا غیرفعال بودن دریافت نوتیفیکشن ها و به طور کلی داده‌های با حجم کم، معمولا از SharedPreferences استفاده می شود. SharedPreferences به ما اجازه می دهد تا اطلاعات را با فرمت (Key/Value کلید/مقدار) ذخیره و نگهداری کنیم. این اطلاعات با فرمت xml ذخیره شده و تا زمانی که اپلیکیشن از روی سیستم عامل حذف نشده و یا کاربر به صورت دستی در قسمت تنظیمات برنامه ها، Data اپلیکیشن را حذف نکند، باقی می ماند.

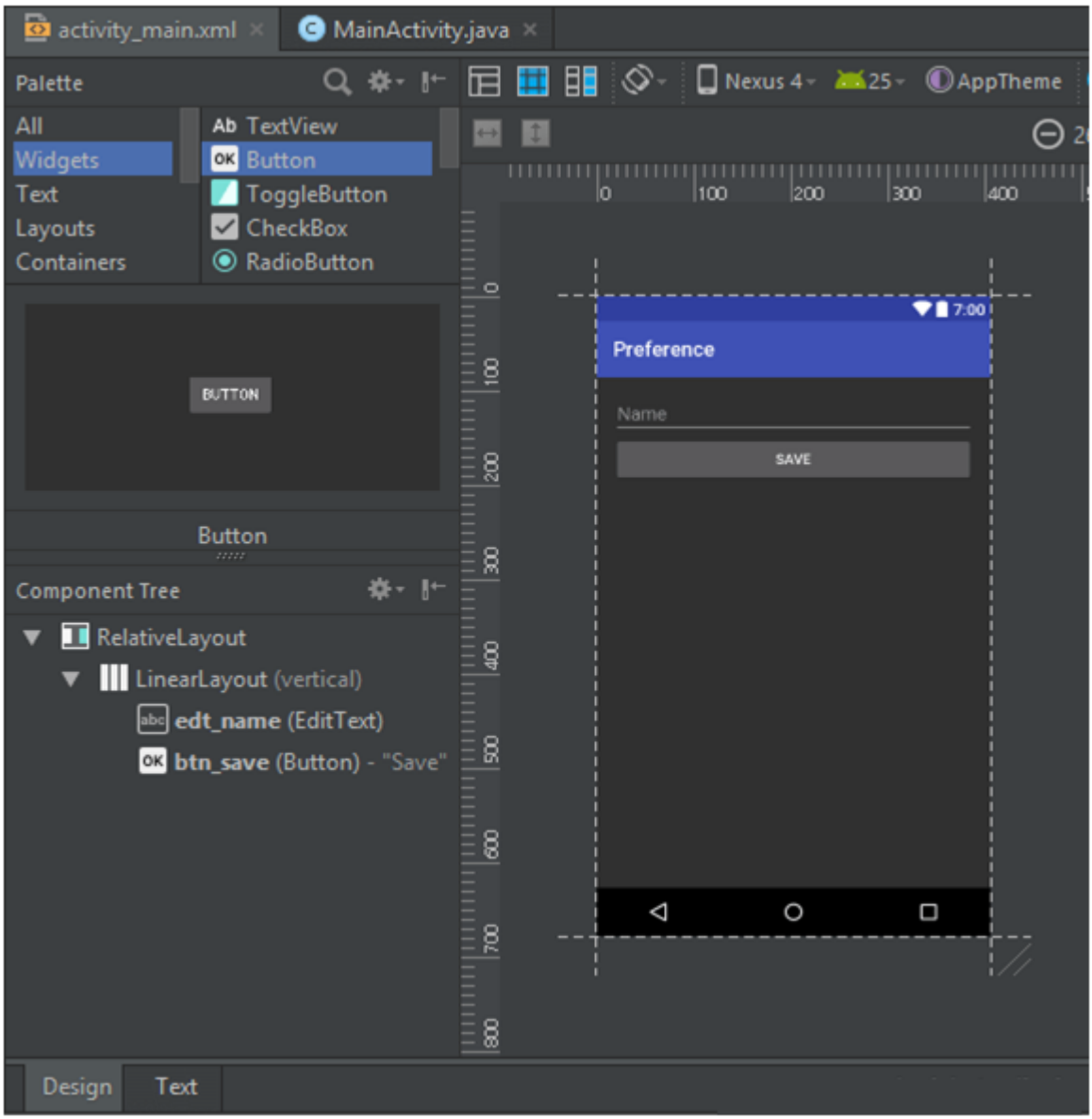
متدهای SharedPreferences

برای دستیابی به Preference ها به سه متد دسترسی داریم:

– getPreferences()
– getSharedPreferences()
– getDefaultSharedPreferences()

در این مبحث از متد getSharedPreferences() استفاده می کنیم. قبل از ادامه توضیحات، یک پروژه جدید می سازم که نام آن را Preference گذاشته ام. در ابتدا اشیاء را طبق تصویر زیر به اکتیویتی activity_main.xml اضافه می کنیم:





در قدم بعد، این دو Widget را در MainActivity.java تعریف می کنیم:

- activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<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"
   android:paddingLeft="16dp"
   android:paddingRight="16dp"
   android:paddingTop="16dp"
   android:paddingBottom="16dp"
   tools:context="ir.android_studio.preference.MainActivity">
```

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">

   <EditText
      android:id="@+id/edt_name"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:hint="Name"
      android:inputType="textPersonName" />

   <EditText
      android:id="@+id/edt_family"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:hint="Family"
      android:inputType="textPersonName" />

   <EditText
      android:id="@+id/edt_age"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:hint="Age"
      android:inputType="number" />

   <Button
      android:id="@+id/btn_save"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Save" />

   <Button
      android:id="@+id/btn_remove"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Remove" />
```

```
    <Button
      android:id="@+id/btn_clear"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Reset All" />
  </LinearLayout>
</RelativeLayout>
```

- MainActivity.java

```
package ir.android_studio.preference;

import android.content.Context;
import android.content.SharedPreferences;
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.Toast;

public class MainActivity extends AppCompatActivity {

  EditText edtName;
  EditText edtFamily;
  EditText edtAge;
  Button btnSave;
  Button btnRemove;
  Button btnClear;
  SharedPreferences shPref;
  public static final String MyPref = "MyPrefers";
  public static final String Name = "nameKey";
  public static final String Family = "familyKey";
  public static final String Age = "ageKey";

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

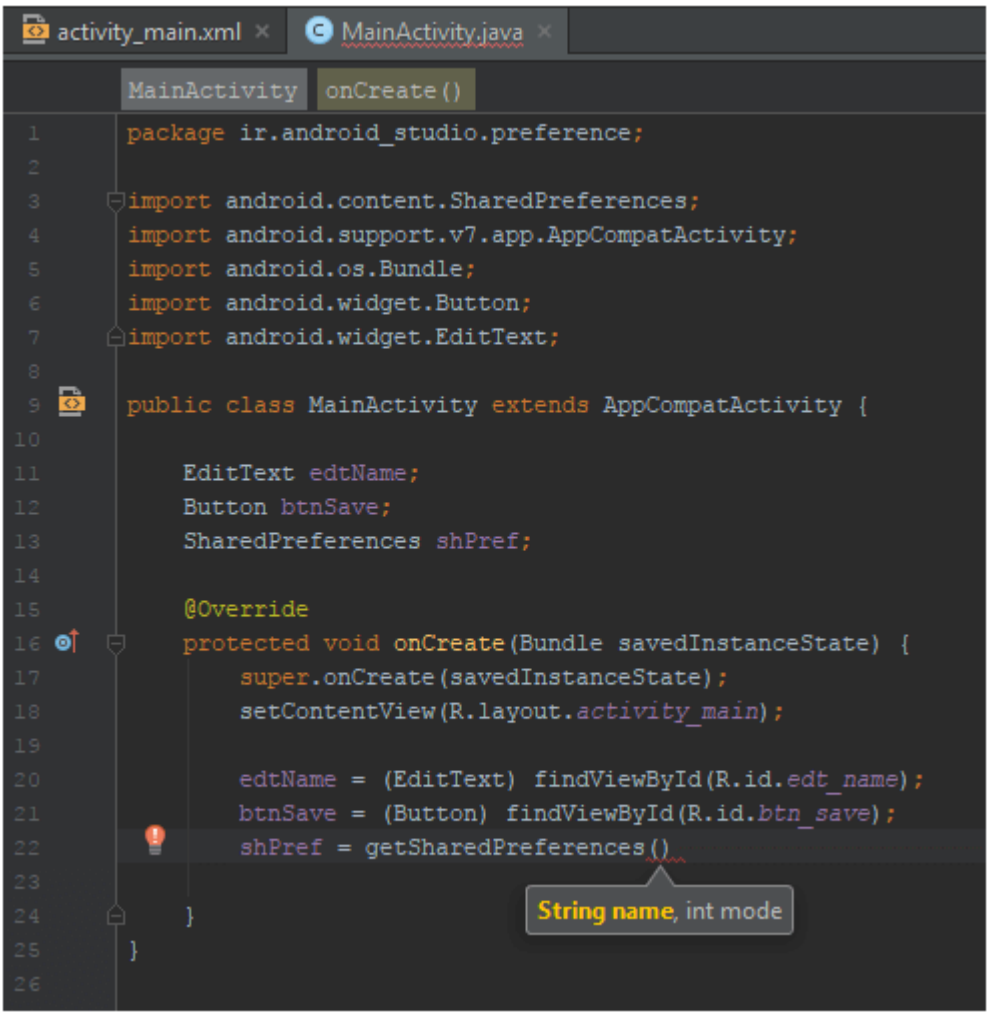
```
        edtName = (EditText) findViewById(R.id.edt_name);
        edtFamily = (EditText) findViewById(R.id.edt_family);
        edtAge = (EditText) findViewById(R.id.edt_age);
        btnSave = (Button) findViewById(R.id.btn_save);
        btnRemove = (Button) findViewById(R.id.btn_remove);
        btnClear = (Button) findViewById(R.id.btn_clear);
        shPref = getSharedPreferences(MyPref, Context.MODE_PRIVATE);

        btnSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String n = edtName.getText().toString();
                String f = edtFamily.getText().toString();
                int a = Integer.parseInt(edtAge.getText().toString());

                SharedPreferences.Editor sEdit = shPref.edit();
                sEdit.putString(Name, n);
                sEdit.putString(Family, f);
                sEdit.putInt(Age, a);
                sEdit.apply();

                Toast.makeText(MainActivity.this, "Saved",
Toast.LENGTH_LONG).show();

            }
        });

        btnRemove.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                SharedPreferences.Editor rEdit = shPref.edit();
                rEdit.remove(Name);
                rEdit.remove(Family);
                rEdit.apply();

                Toast.makeText(MainActivity.this, "Removed!",
Toast.LENGTH_LONG).show();
```

```
        }
    });

    btnClear.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            SharedPreferences.Editor cEdit = shPref.edit();
            cEdit.clear();
            cEdit.apply();

            Toast.makeText(MainActivity.this, "All data cleared!",
Toast.LENGTH_LONG).show();

        }
    });

    if (shPref.contains(Name)) {
        edtName.setText(shPref.getString(Name, null));
    }

    if (shPref.contains(Family)) {
        edtFamily.setText(shPref.getString(Family, null));
    }

    if (shPref.contains(Age)) {
        edtAge.setText(String.valueOf(shPref.getInt(Age, 0)));
    }

  }
}
```

حالا از اینترفیس SharedPreferences یک نمونه می سازم. من نام shPref را برای این نمونه در نظر گرفتم:

```
SharedPreferences shPref;
```

در ادامه shPref را مساوی متد getSharedPreferences() قرار می دهم:



ملاحظه می کنید این متد دو پارامتر ورودی می گیرد. اولی name و دومی mode. پارامتر نخست مربوط به نام فایلی است که برای ذخیره سازی داده ها استفاده می شود و اختیاری است. من MyPref وارد می‌کنم (یک متغیر از نوع String با نام MyPref و مقدار MyPrefers ساخته ام. برای موارد بعدی نیز متغیر تعریف می کنم تا از وارد کردن دستی نامها جلوگیری شود که در نهایت منجر به افزایش سرعت و کاهش خطا می شود). پارامتر دوم مربوط به mode روش عملیاتی است.

از MODE_PRIVATE استفاده می کنیم:

```
shPref = getSharedPreferences("MyPref", Context.MODE_PRIVATE)
```

خواسته ما از برنامه این است که پس از وارد کردن نام و لمس کلید Save توسط کاربر، نام وارده ذخیره شده و با بستن و اجرای مجدد اپلیکیشن، اطلاعات وارد شده نمایش داده شود. سپس متد مربوط به دکمه btnSave را کامل کرده و سایر دکمه ها را نیز به همین روش ایجاد می کنیم.

در خط اول متغیری از جنس String با نام n تعریف کردم که با دستور

```
edtName.getText().toString()
```

Text ورودی کاربر را از edtName گرفته و در خود ذخیره می کند (توسط getText رشته ورودی دریافت و به وسیله toString به رشته از جنس String تبدیل می شود).

حالا به یک ادیتور نیاز داریم تا به واسطه آن اطلاعات را ذخیره و یا ویرایش کنیم. یک نمونه از SharedPreferences.Editor با نام sEdit ساختم که با shPref.edit() آماده ذخیره و یا ویرایش اطلاعات است.

در مرحله بعد باید محتویات درون n را به Name منتقل کنیم. این کار توسط متد putString انجام می‌شود:

```
sEdit.putString(Name, n);
```

همانطور که از نام متد پیداست، برای ذخیره و یا ویرایش داده های از نوع String بکار می رود. به همین ترتیب برای مقادیر صحیح عددی از متد putInt ، مقادیر boolean متدputBoolean ، مقادیر شناور متد putFloat و مقادیر long متد putLong استفاده می شود.

در نهایت توسط متد commit() تغییرات اعمال شده ذخیره می شود:

```
sEdit.commit();
```

اندروید استودیو به من توصیه می کند به جای commit از متد apply استفاده کنم، بنابراین sEdit.apply() را جایگزین کردم (تفاوتهایی بین این دو وجود دارد که در صورت تمایل با جستجوی عبارت "commit() vs apply()" به جوابهای خوبی خواهید رسید.) برای زیبایی کار یک پیغام از نوع Toast به دکمه اضافه کردم تا پس از لمس دکمه عبارت Saved به کاربر نمایش داده شود. تا اینجای کار اگر کد ما مشکلی نداشته باشد اطلاعات وارد شده باید به درستی ذخیره شود. اما هنوز نیاز به یک دستور شرطی داریم تا بعد از خروج کاربر از برنامه و ورود مجدد به آن، چک کند اگر مقادیری در Preference ذخیره شده در محل موردنظر نمایش داده شود. به کد زیر دقت کنید:

```
if (shPref.contains(Name)) {
   edtName.setText(shPref.getString(Name, null));
}
```

این کد می گوید اگر shPref حاوی Name است، توسط متد getString مقدار آنرا گرفته و توسط متد setText روی ویجت edtName نمایش بده.

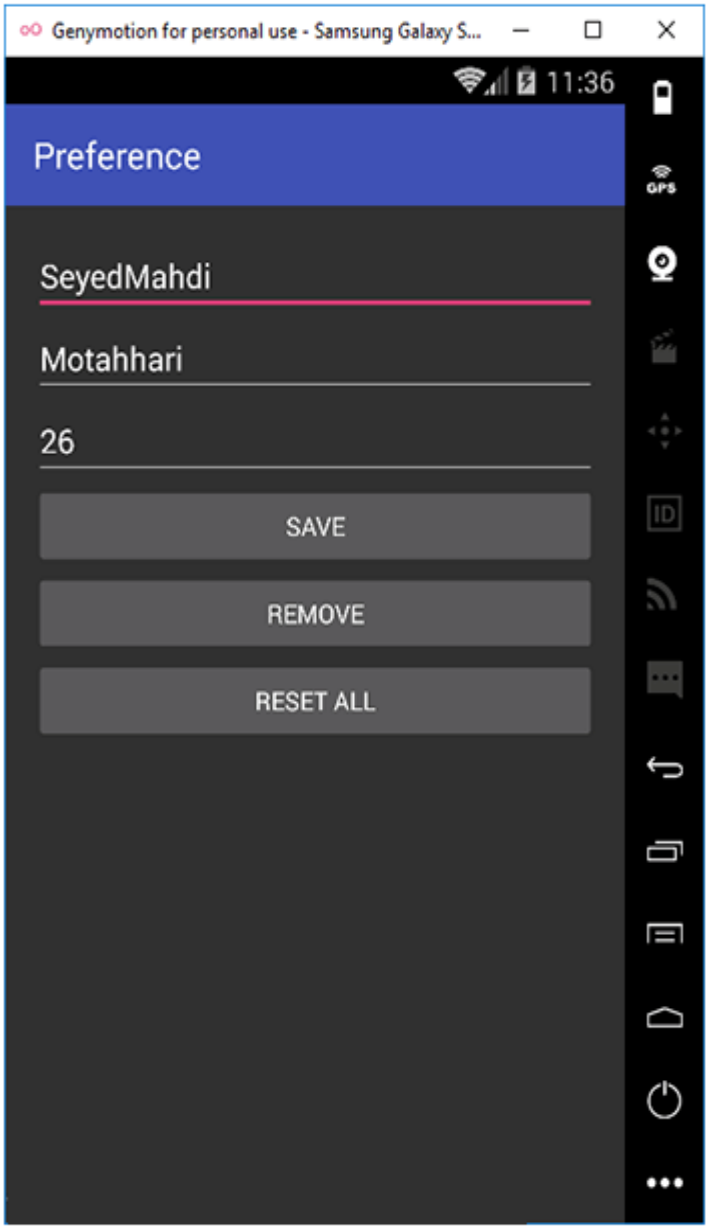
مشابه متدهای set ، متدهای get شامل getString ، getInt، getBoolean، getFloat و getLong برای بازیابی داده های ذخیره شده در اختیار ما قرار گرفته است. این متدها نیز دو پارامتر ورودی دارند که اولی کلید مربوط به داده ذخیره شده و پارامتر دوم مقدار پیش فرض است. پیش فرض به این معنی که اگر در کلید مربوطه به عنوان مثال Name داده ای ذخیره نشده بود، مقداری که اینجا تعیین کرده ایم به کاربر نمایش داده شود. یک مثال میزنم:

```
edtName.setText(shPref.getString(Name, "Name not saved!"));
```

در حالت بالا اگر قبلا در Name چیزی ذخیره نشده باشد، پس از اجرای اپلیکیشن در ویجت edtName عبارت Name not saved! قرار می گیرد. اما چون من قبلا در لایه رابط کاربری مقدار Name را به عنوان hint برای edtName درنظر گرفته ام، از این پارامتر صرف نظر کرده و مقدار null قرار داده ام. البته به جای null رشته خالی یعنی "" هم می توان استفاده کرد:

```
edtName.setText(shPref.getString(Name, ""));
```

حال پروژه را اجرا می کنیم:



نام، نام خانوادگی و سن را وارد کرده و با زدن دکمه Save آن را ذخیره می کنیم. مقدار int به تنهایی نمی‌تواند توسط setText به ویجت ارسال شود. راه حل ساده این است که یک رشته خالی به آن اضافه کنیم:

```
edtAge.setText(shPref.getInt(Age, 0) + "");
```

اما من روشی را انتخاب می کنم که اصولی باشد. به اینصورت که توسط String.valueOf() عدد صحیح به String تبدیل شده و سپس به edtAge فرستاده می شود. عدد صفر هم به عنوان مقدار پیش فرض تعیین شده است.

برای سادگی و مختصر شدن کد بهتر بود سن را هم از نوع String تعریف کرده و فقط در رابط کاربری، فیلد مورد نظر محدود به کاراکترهای عددی می شد. با این حال بهانه ای شد تا از متد putInt و موارد مربوط به تبدیل String به Integer نیز استفاده شود.

اما در Shared Preferences دو متد دیگر نیز در اختیار داریم:

- remove(“KeyName”): توسط این متد می توان دیتا یا دیتاهای مدنظر را حذف کرد.
- clear():با اجرا شدن این متد کلیه اطلاعات ذخیره شده حذف می گردد.

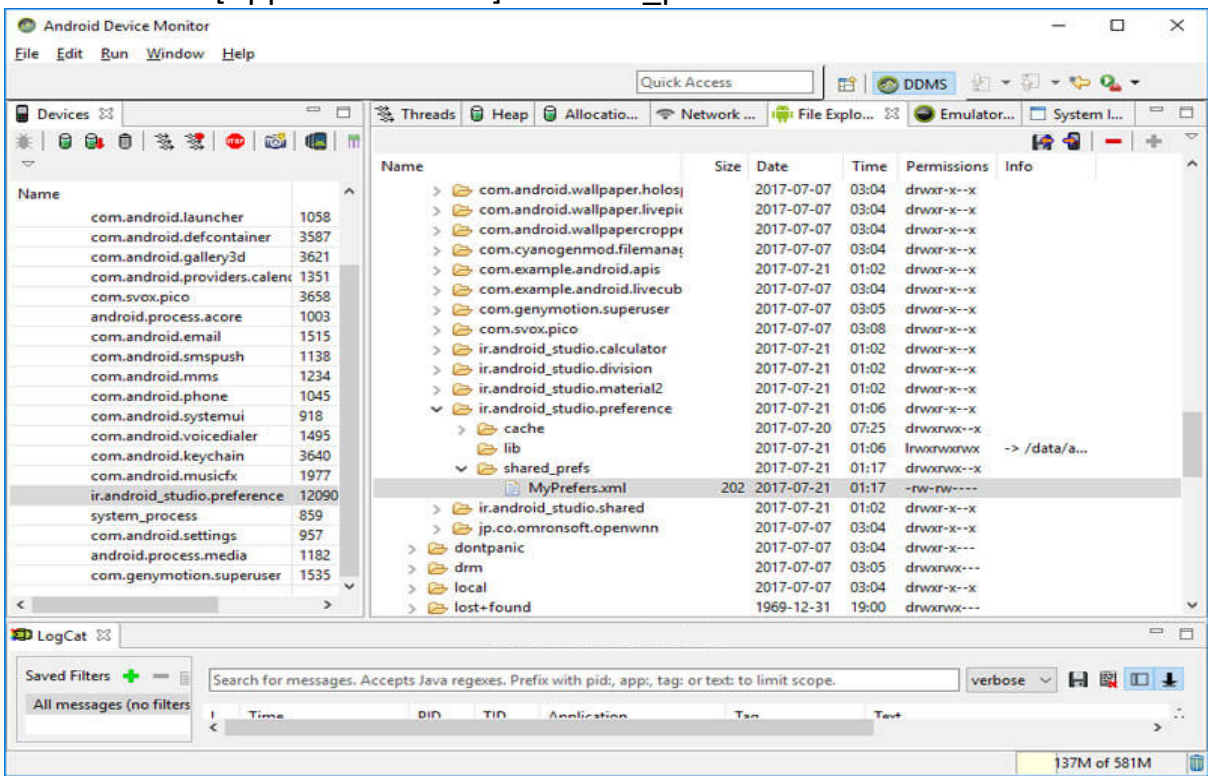
با کلیک دکمه Remove ، نام و نام خانوادگی و با کلیک Reset تمامی اطلاعات حذف می شود. متد مربوط به دکمه ها را مانند دکمه save ابتدا نمونه راذساخته و سپس متدهای مورد نیاز را فراخوانی می‌کنیم. توجه داشته باشید در صورتی که متد apply() یا commit() اضافه نشود، تغییرات اعمال نخواهد شد.

از اپلیکیشن خارج شده و مجددا آن را اجرا می کنیم. برنامه به درستی عمل کرده و موارد ذخیره شده در EditText ها نمایش داده می شود. اگر نام فعلی را تغییر داده و مجدد ذخیره کنیم، جایگزین نام قبلی خواهد شد. انتظار داریم با زدن گزینه Remove و سپس بستن و اجرای مجدد برنامه، نام و نام خانوادگی حذف شده باشد. اطلاعات فیلدهای نام و نام خانوادگی با موفقیت حذف شد. در مرحله بعد برای اطمینان از عملکرد صحیح برنامه، ابتدا نام و نام خانوادگی را مجدد ذخیره کرده و در نهایت روی Reset all کلیک می‌کنیم. متد clear() نیز به درستی عمل کرده و پس از خروج از برنامه و اجرای مجدد، اطلاعاتی که وارد کرده بودم نمایش داده نمی شود.

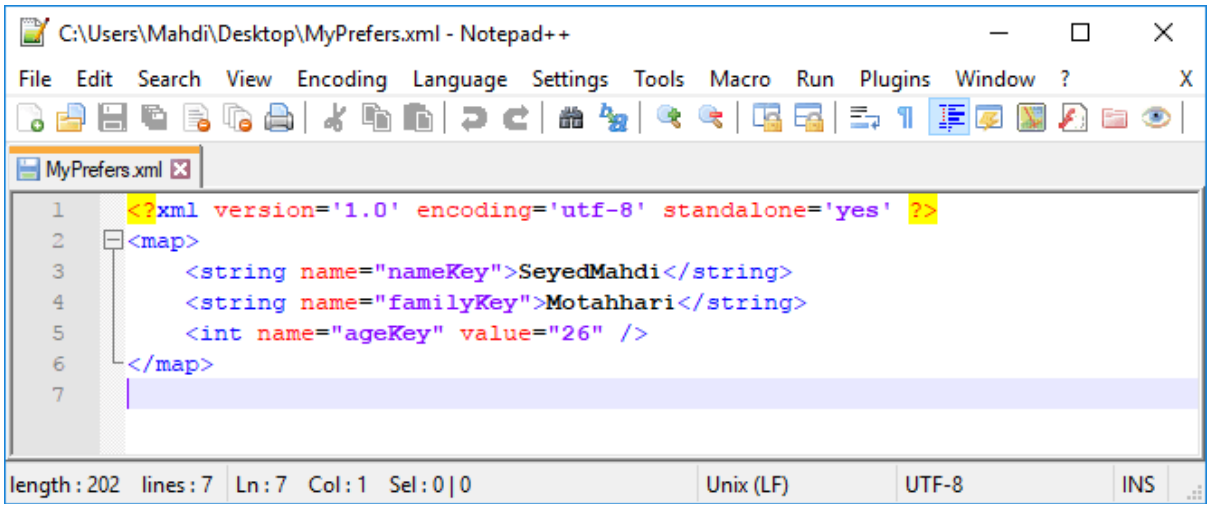
دسترسی به فایل Preference

امکان دسترسی به فایل Preference نیز وجود دارد. از منوی Tools > Android پنجره Android Device Monitor را باز می کنم. در سمت چپ (Device) نام اپلیکیشن را انتخاب کرده سپس در سمت راست در تب File Manager مسیر زیر را دنبال می کنم:

Data > data > [Application name] > shared_prefs



فایل با همان نامی که تعیین کرده بودم ایجاد شده است. توسط گزینه Pull a file from a device فایل MyPrefers.xml را روی رایانه ذخیره و باز می کنم:



مشاهده می کنید تمامی اطلاعات درون این فایل ذخیره شده است.