سلام اندروید

آموزش شروع برنامه‌نویسی اندروید

(اولین خروجی ساده‌ای که گرفته می‌شود را در برنامه‌نویسی هلو ورد یا سلام دنیا میگند)

Creating better world, Starting with a hello world

Omid karbalaei

هدف اصلی این کتاب کوچیک ریختن ترس خواننده هاش از برنامه نویسی هست که خیلی ها فکر میکنند علم پیچیده ای هست، درصورتی که ما معتقدیم با یک آموزش خوب میشود به راحتی اون رو یاد گرفت.

ما در این کتاب کوچیک گرفتن اولین خروجی در اندروید استودیو با زبان جاوا ، فلاتر ، ری اکت نیتیو و بیسیک فور اندروید یاد میدهیم

در پایان این کتاب به شما یک آموزش ویدیویی باکیفیت و رایگان هدیه داده می‌شود که مفاهیم کمی پیشرفته‌تر از مباحث این کتاب را با آموزشی قابل‌فهم و راحت به شما یاد خواهد داد.

توصیه می‌کنم با پشتکار و جدیت تمام مطالب کتاب را مطالعه کنید و مطمئن باشید در ویدیو جبرانی که آخر کتاب به شما رایگان هدیه داده می‌شود، مفاهیم بیشتر را به‌راحتی یاد می‌گیرید.

سلام من امید کربلایی‌ام. بسیاری افراد من را با شهرت برنامه‌نویس تک‌انگشتی می‌شناسند. به نظرم بزرگ‌ترین مانع در برابر یادگیری برنامه‌نویسی این است که برخی فکر می‌کنند برنامه‌نویسی سخت و یادگیری آن زمان‌بر است، به دانش پیچیده ریاضی نیاز دارد یا اینکه یک لپ‌تاپ مدل بالا می‌خواهد.

من قصد دارم در این کتاب ک به شما کمک کنم که اولین برنامه اندروید خود را بسازید تا این تابوها در ذهنتان بشکند و به دنیا سلام کنید!

در شبی از سال‌های دبیرستان لیست کارهای مورد علاقه‌ام را در گوشی نوشتم. متأسفانه به‌دلیل معلولیت قادر به انجام هیچ کدام از آن‌ها نبودم!

کاری که بیشترین تناسب را با توانایی‌های جسمی من داشت، برنامه‌نویسی بود. متأسفانه وقتی تصمیم به شروع آن گرفتم، در تست اولیه و پیش‌نیازهای آن هم رد شدم. زیرا برای برنامه‌نویسی نه تنها باید بتوانیم همه دکمه‌های کیبورد را بزنیم؛ بلکه بیشتر اتفاقات مهم با شورت‌کات‌ها رقم می‌خورند. یعنی باید چند کلید را با هم فشار داد که عمل مورد نظر انجام بشود؛ اما به دلیل اینکه فقط یک انگشتم کار می‌کرد، در انجام آن ناتوان بودم و یک مدت ناامید شدم. این کار بین اتفاق‌های جذاب زیادی رخ داد که شرح آن در این کتاب کوچک نمی‌گنجد. تا اینکه نرم‌افزارهایی را در اندروید دیدم که کمک می‌کرد با گوشی، لپ‌تاپ را کنترل کنم. از آن روز بود که استارت تسخیر دنیا را با یک انگشت زدم. بنابراین از سال ۹۰ یک راه پیدا کردم که با گوشی به لپ‌تاپ وصل شوم و آن را کنترل کنم. با یک انگشت یادگیری برنامه‌نویسی را شروع کردم. بعد از چند برنامه موفق با دانلودهای بالا مدیریت گروه‌های کوچک برنامه‌نویسی را بر عهده گرفتم و در سال ۹۶ امید کربلایی یک کارآفرین شد. در آن دوره ۲۵ نیرو به شکل مستقیم و بیش از ۲۰۰ فریلنسر در کنار خودم داشتم. زمان شروع کار در جنوب شهر یعنی امیریه بودم و در زمان نوشتن این کتاب در زعفرانیه زندگی می‌کنم. شاید زندگی شما هم با این کتاب مثل من متحول بشود.

اگر به داستان شغلی من علاقه‌مند شده‌اید و می‌خواهید بیشتر در این زمینه بدانید، می‌توانید من را در شبکه‌های اجتماعی با آیدی omidkarbalaei01 دنبال کنید یا به سایت شخصی من omidmhks.com سر بزنید.

اکنون که احتمالا با شنیدن داستان من انگیزه گرفته‌اید، بیایید برای ساخت دنیای بهتر که با سلام کردن به آن شروع می‌شود، قدم برداریم.

گرفتن اولین خروجی با جاوا

همان‌طور که قبلا گفتم، به طور کلی برای شروع برنامه‌نویسی اندروید نیاز به نصب موارد زیر دارید:

1. نصب JDK (Java Development Kit)

2. نصب اندروید استودیو (Android Studio)

3. نوشتن اولین برنامه (Hello, World!)

1. نصب JDK

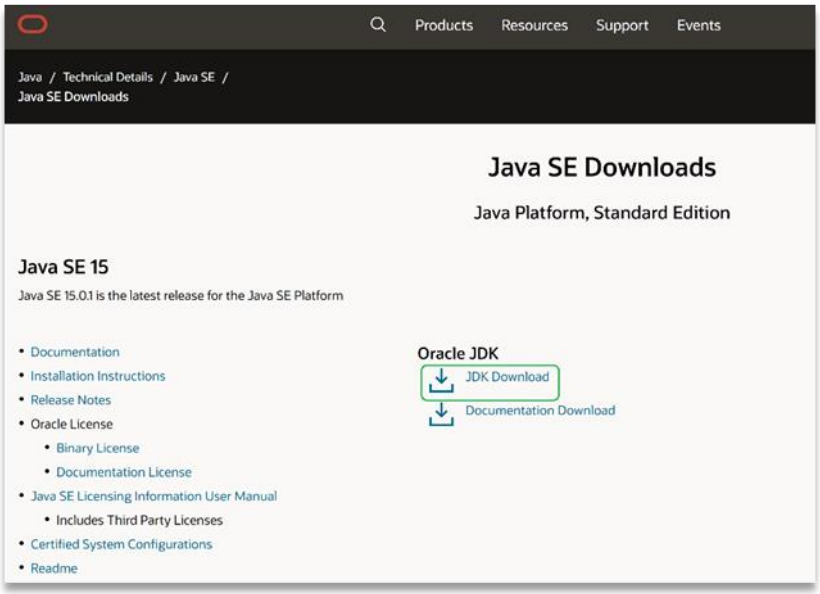
جاوا یک زبان برنامه‌نویسی شی‌گرا است که برنامه‌نویسان با استفاده از آن نرم‌افزارهای مختلفی را می‌نویسند و منتشر می‌کنند. برنامه‌نویسان برای توسعه نرم‌افزار با استفاده از زبان جاوا به نصب بسته JDK روی سیستم عامل خود نیاز دارند. آخرین نسخه JDK برای ویندوز و سایر سیستم عامل‌ها، ویرایش شماره 15 است. در ادامه نحوه نصب JDK روی سیستم عامل ویندوز 10 به تفصیل بیان شده است.

برای نصب JDK نسخه 15 باید مراحل زیر به ترتیب انجام شوند:

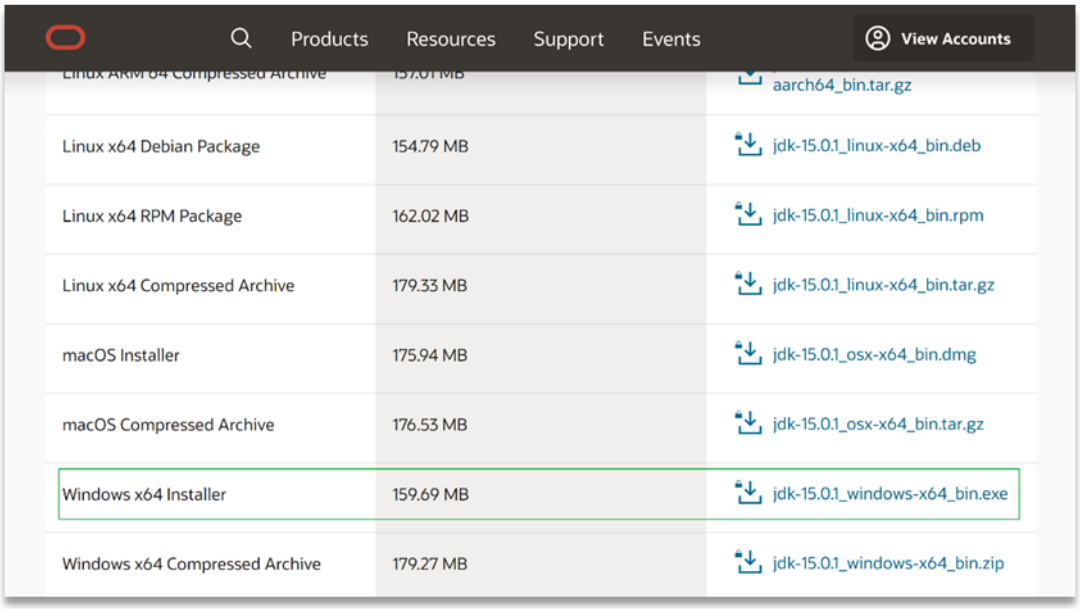
1- شروع نصب JDK 15

1-1- برای نصب این بسته باید صفحه دانلود بسته JDK از وب‌سایت اوراکل به نشانی www.oracle.com/java/technologies/javase-downloads.html را در مرورگر خود باز کنیم و سپس در صفحه باز شده روی گزینه JDK Download کلیک کنیم.

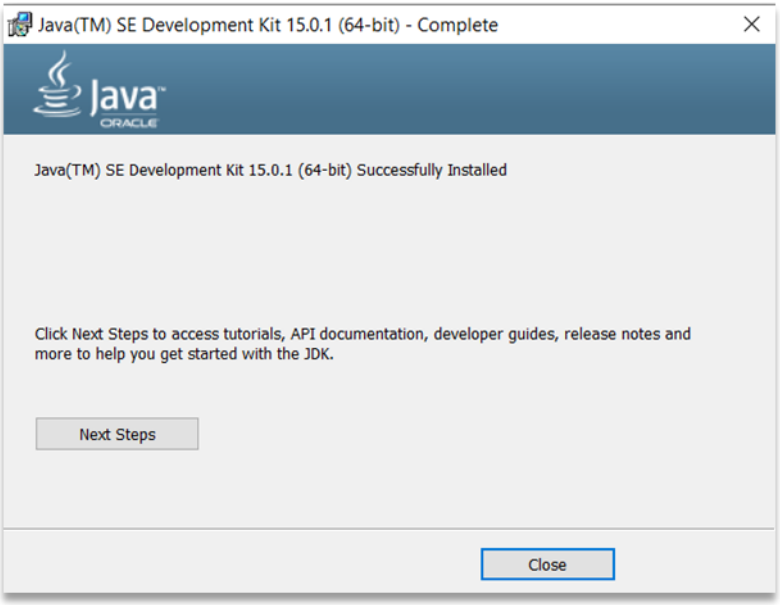
1-2- در این صفحه لینک‌های دانلود نسخه JDK 15 برای سیستم‌عامل‌های مختلف وجود دارد. به دلیل کاربرد بیشتر سیستم‌عامل ویندوز، آموزش این کتاب نیز اساس این سیستم‌عامل است. بر لینک متناظر با گزینه Windows x64 Installer کلیک می‌کنیم و برای اتمام دانلود فایل نصب JDK منتظر می‌مانیم.



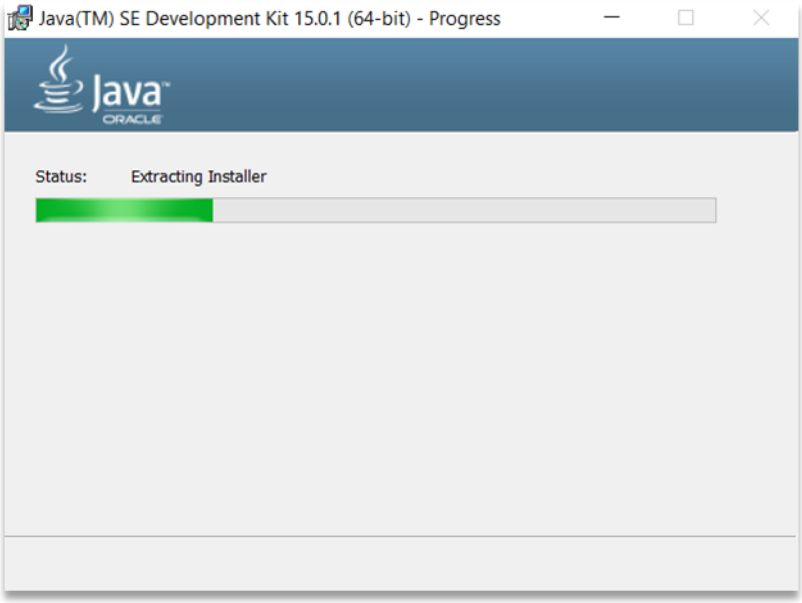
1-3- پس از اتمام دانلود بسته نصب، آن را باز کرده و با انتخاب Run as administrator، فایل JDK را اجرا می‌کنیم. پس از باز شدن بسته، روی گزینه Next کلیک می‌کنیم.



1-4- در صفحه بعدی نیز روی دکمه Next کلیک می‌کنیم.

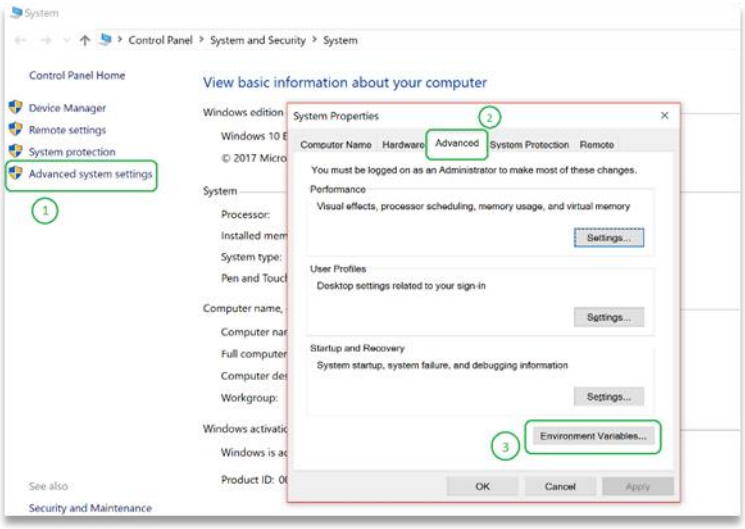


1-5- در شکل زیر نیز مرحله بعدی نصب نشان داده شده است که با کلیک روی دکمه Next به مرحله بعدی می‌رویم.



1-6- در این مرحله با کلیک روی دکمه Close، مراحل نصب بسته را به پایان می‌رسانیم.

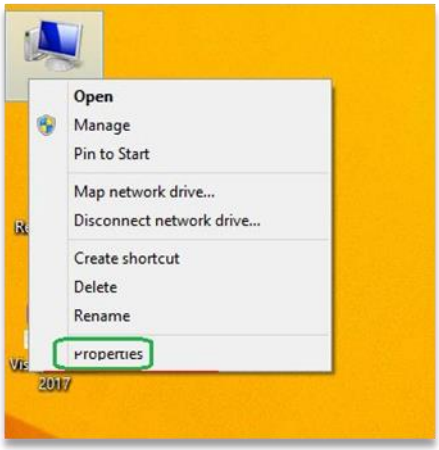
2- تنظیمات شناسایی جاوا



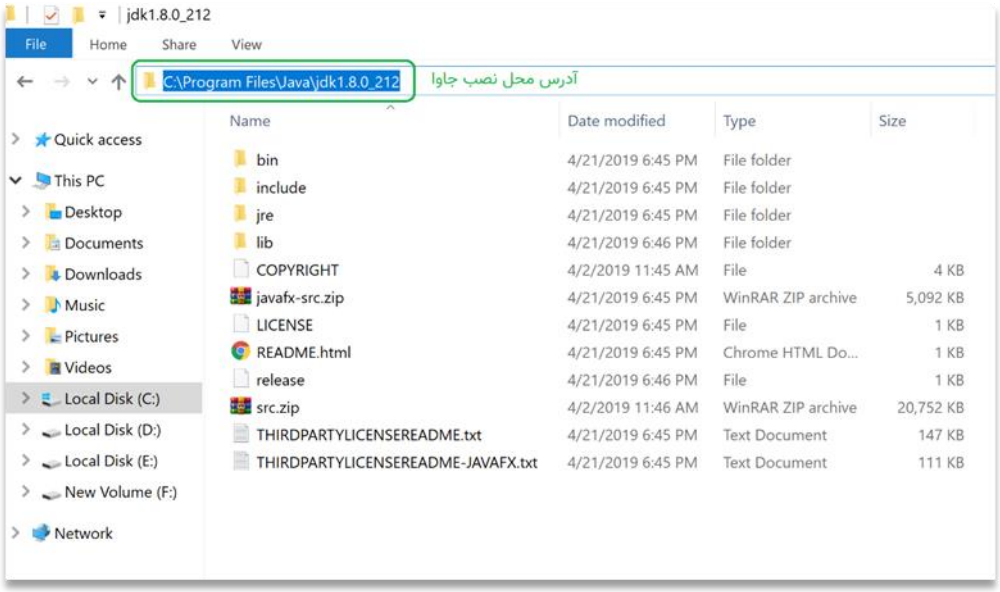
تا این مرحله بسته JDK به‌طور کامل نصب شد. قبل از نصب اندروید استودیو باید مسیر نصب جاوا را به سیستم‌عامل معرفی کنیم. در غیر این صورت اندروید استودیو، جاوا را نمی‌شناسد و نمی‌تواند مسیر نصب آن را پیدا کند. در ادامه مراحل شناسایی جاوا را برای سیستم‌عامل بررسی می‌کنیم:

2-1- در ابتدای کار روی This PC راست کلیک و گزینه Properties را انتخاب می‌کنیم.

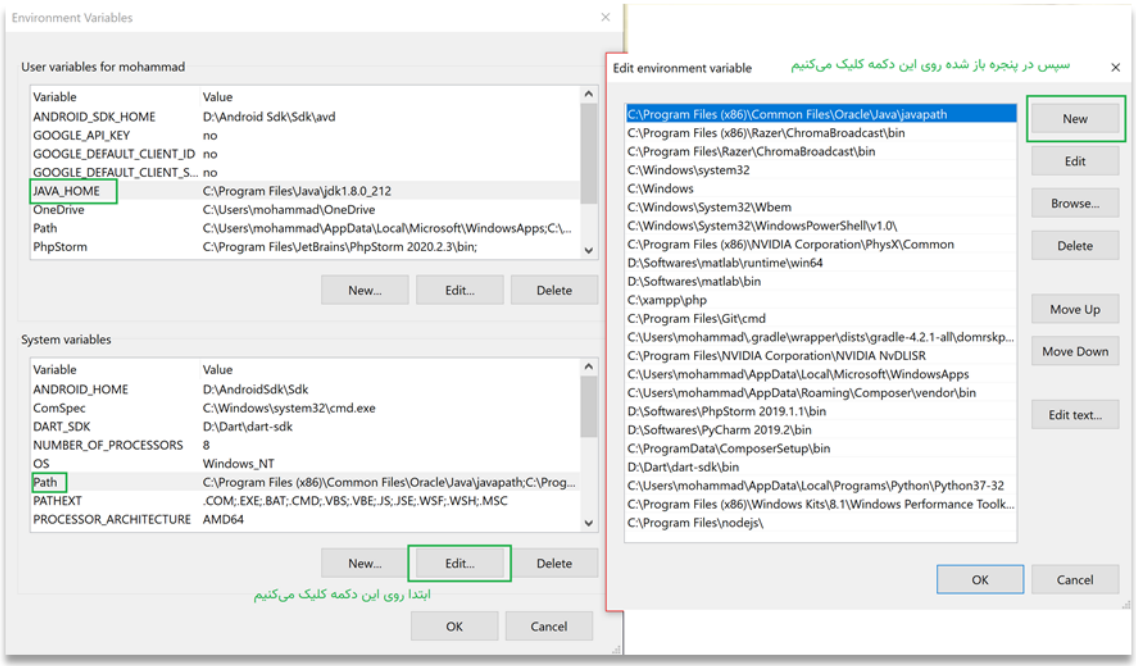
2-2- در پنجره باز شده در سمت چپ روی گزینه Advanced system settings کلیک کرده و در مرحله بعد Advanced و در نهایت گزینه Environment variables را انتخاب می‌کنیم.

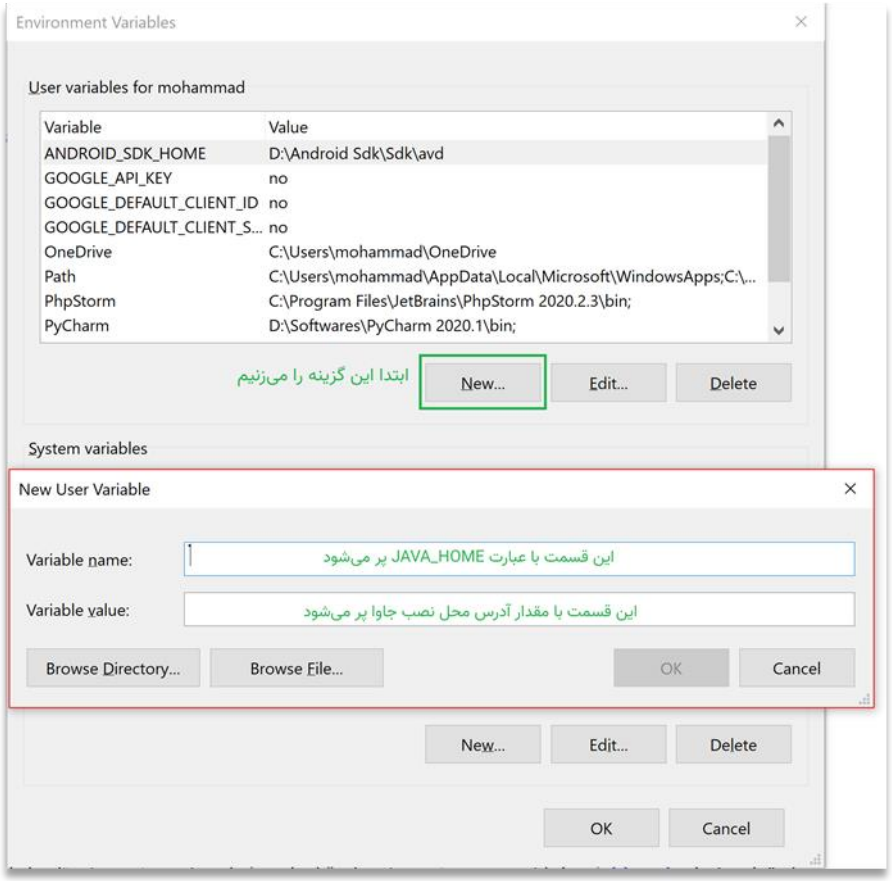


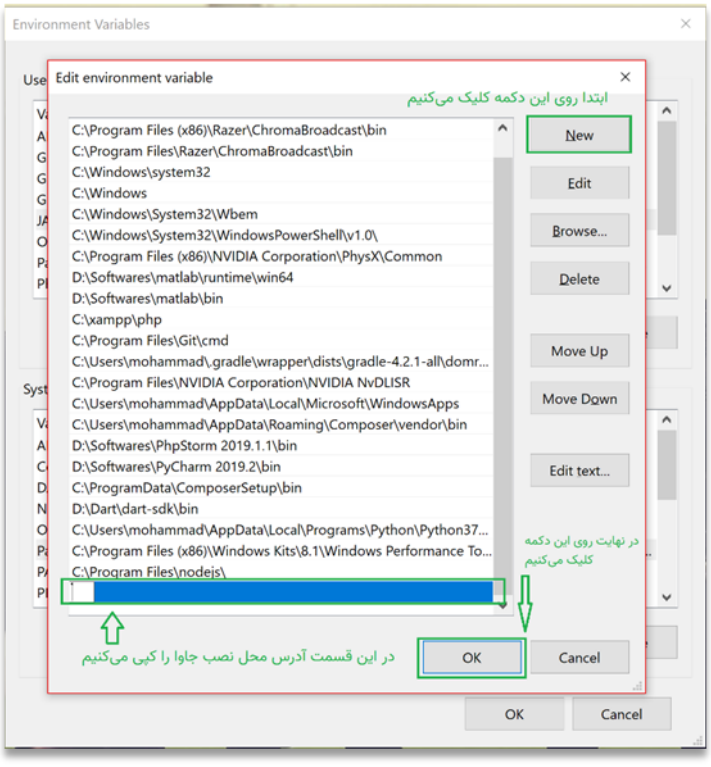
2-3- در این مرحله از پنجره باز شده روی گزینه New کلیک می‌کنیم. با کلیک روی این گزینه، پنجره جدیدی باز می‌شود که دو قسمت دارد. قسمت اول که Variable Name نام دارد را باید با عبارت "JAVA_HOME" نامگذاری کنیم. قسمت دوم که Variable Value نام دارد، باید با مقدار آدرس محل نصب جاوا تکمیل شود. بهتر است که آدرس محل نصب جاوا را به‌جای تایپ کردن، کپی کنید. روال این مرحله در عکس‌های زیر آورده شده است.



2-4- بعد از وارد کردن آدرس محل نصب جاوا و عبارت JAVA_HOME در قسمت‌های مذکور و زدن دکمه OK به پنجره قبلی بازگردانده می‌شویم. حال در این مرحله، درحالی که متغیر JAVA_HOME در حالت انتخاب قرار گرفته، از کادر پایین گزینه Path را انتخاب و روی دکمه Edit کلیک می‌کنیم. سپس در پنجره باز شده روی دکمه New کلیک می‌کنیم که متغیر جدید برای نوشتن آماده شود.







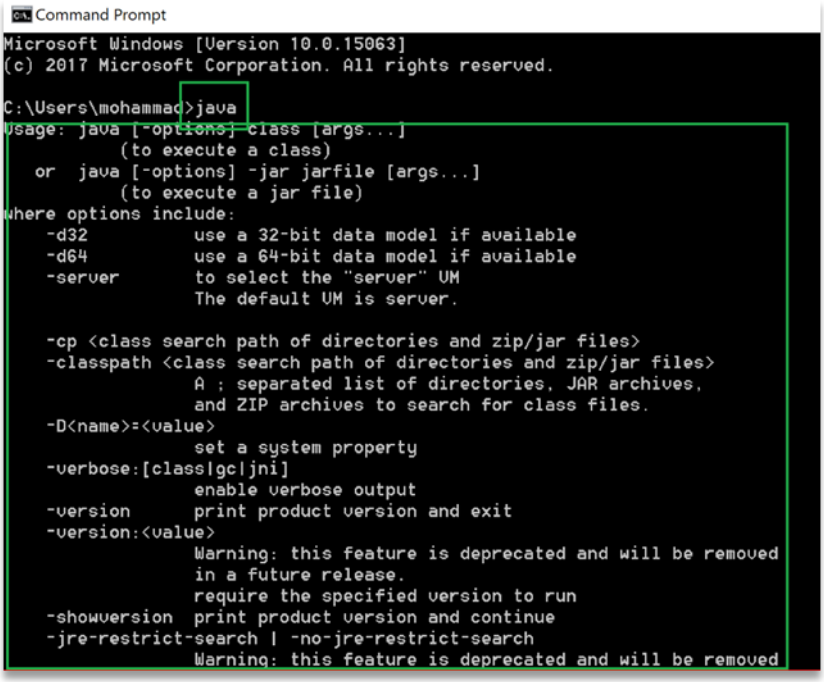
5

2-5- در نهایت بعد از کلیک روی دکمه New متغیر جدید باز می‌شود که مقدار آن را با آدرس محل نصب جاوا تکمیل می‌کنیم و دکمه Ok را می‌زنیم. سپس همه پنجره‌ها را با فشردن دکمه کلیک می‌بندیم و از این قسمت خارج می‌شویم.

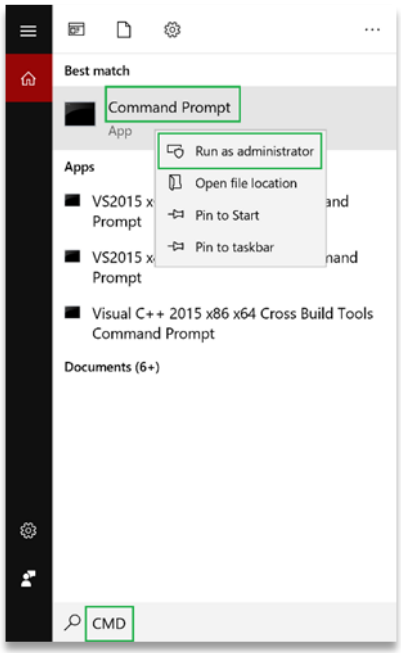
3- تست JDK

تا بدین مرحله کار نصب جاوا به پایان رسیده و برای اینکه بفهمیم جاوا به درستی نصب شده است یا خیر، باید مراحل زیر را دنبال کنیم.

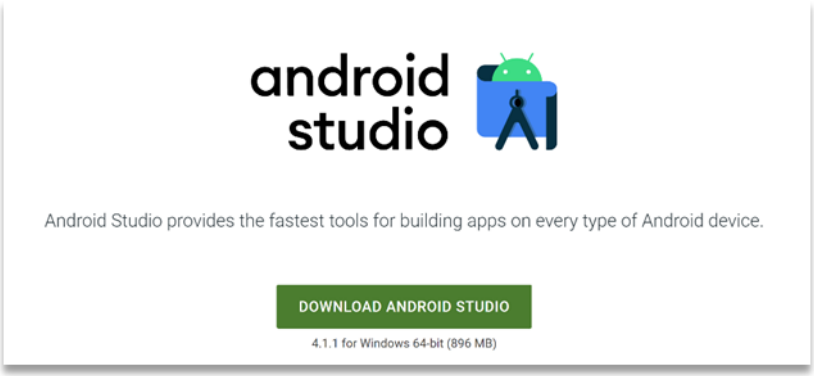
3-1- در قسمت جستجوی ویندوز کلمه CMD را تایپ و Command Prompt را انتخاب کنید. حتما روی آن راست کلیک کرده و گزینه Run as administraitor را انتخاب کنید که مشکلات احراز هویت برای شما به وجود نیاید.



3-2- حال کلمه جاوا را تایپ کنید و Enter را فشار دهید. اگر عبارت مخصوص به جاوا مانند شکل زیر در CMD ظاهر شد، یعنی مراحل نصب جاوا به درستی انجام شده است.



2. نصب اندروید استودیو (Android Studio)

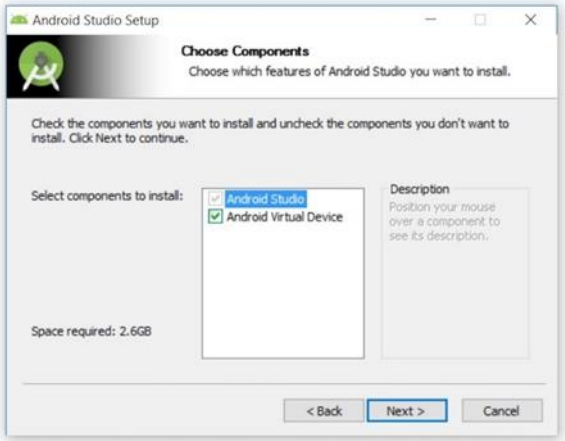


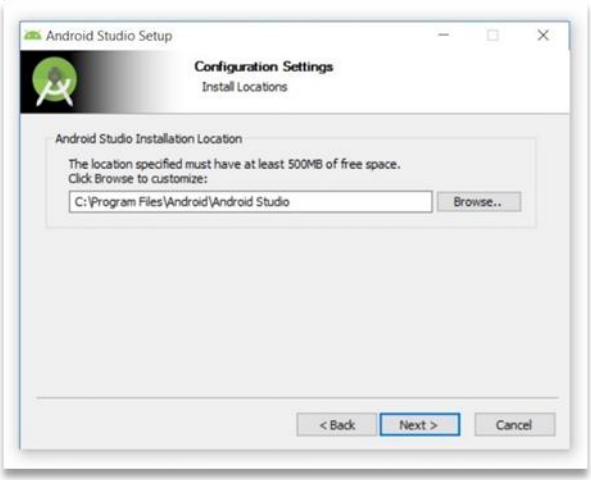
اندروید استودیو، محیط توسعه یکپارچه یا IDE برای توسعه نرم‌افزارهای اندرویدی است که به‌طور رسمی توسط گوگل معرفی شد. این IDE محیط ویرایشی بی‌نظیر و ابزاری برای توسعه نرم‌افزارهای اندرویدی است. برای شروع برنامه‌نویسی اندروید به نصب و اجرای این محیط نیاز دارید. مراحل زیر، گام‌های دانلود و نصب اندروید استودیو را نشان می‌دهد.

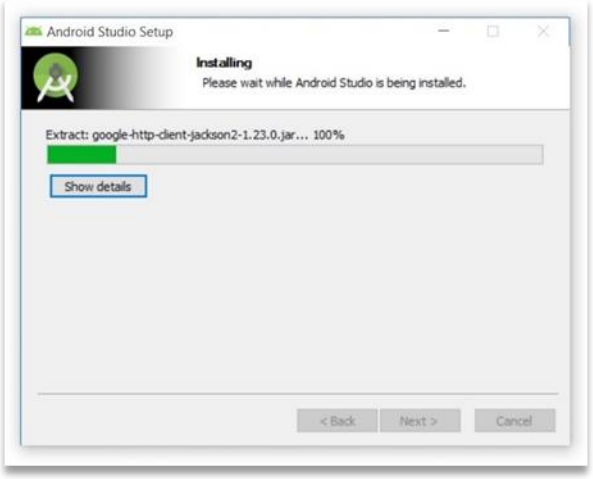
توجه: به دلیل تحریم کشورمون توسط شرکت گوگل، حتما آدرس IP سیستم خودتان را به کشوری غیر از ایران تغییر دهید که مراحل نصب و دانلود به خوبی انجام پذیرد. در غیر این صورت نرم‌افزار دانلود نمی‌شوید و دچار مشکل خواهید شد!

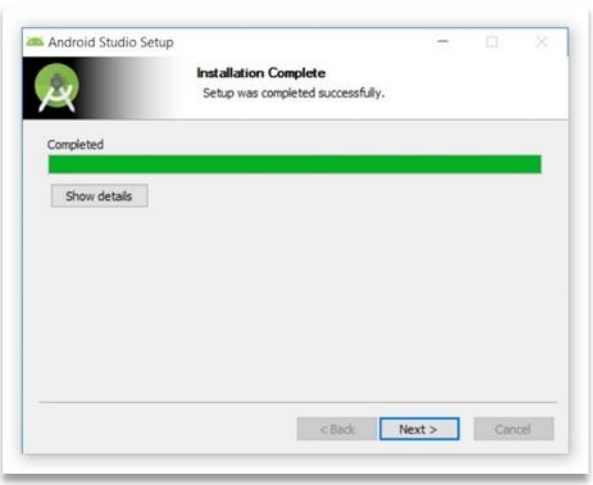
2-1- در اولین گام باید اندروید استودیو را دانلود کنیم. این نرم‌افزار به‌صورت رایگان عرضه شده و از آدرس www.developer.android.com/studio/index.html قابل‌دسترس است. گوگل این نرم‌افزار را برای تمامی سکوها اعم از ویندوز، لینوکس، مک وغیره پیاده‌سازی کرده و در اختیار کاربران قرار داده است. شکل زیر صفحه‌ای که لینک دانلود نرم‌افزار را دارد، نشان می‌دهد. با کلیک روی دکمه نشان داده شده در شکل، نرم‌افزار را دانلود می‌کنیم.

2-2 بعد از اتمام دانلود و باز کردن فایل اجرایی، در همه مراحل روی کلید Next کلیک و صبر می‌کنیم که نصب نرم‌افزار به پایان برسد.

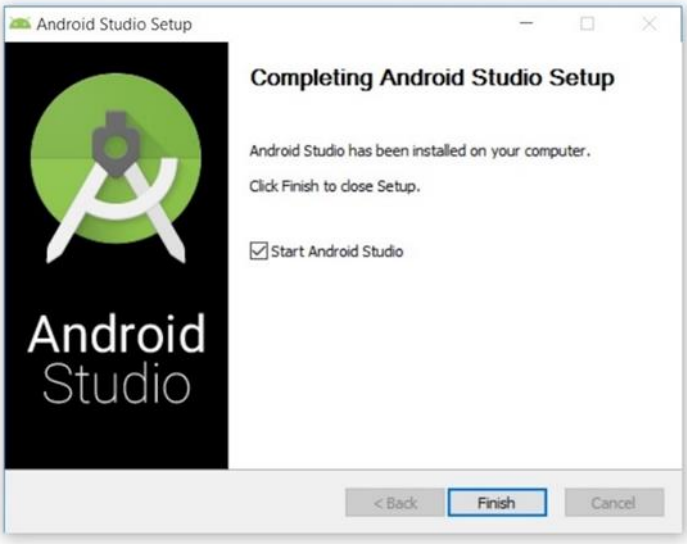




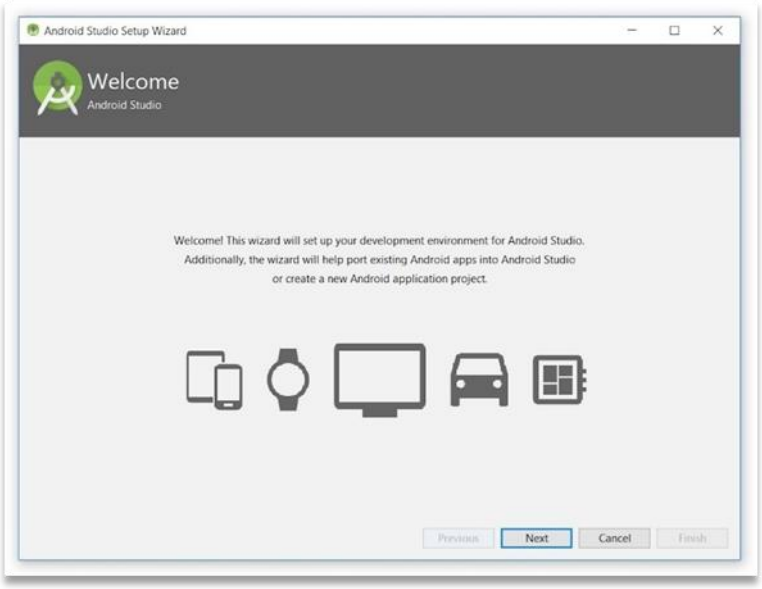




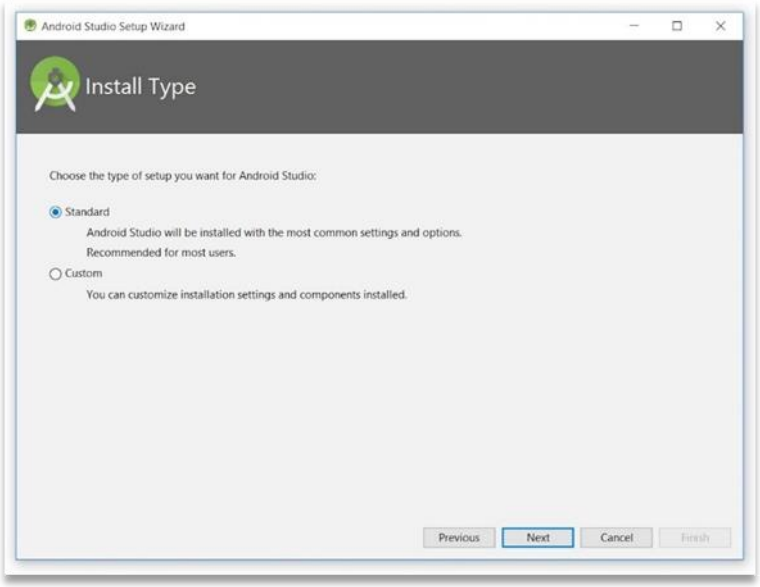
2-3- در نهایت روی دکمه Finish کلیک می‌کنیم و مراحل نصب به پایان می‌رسد.



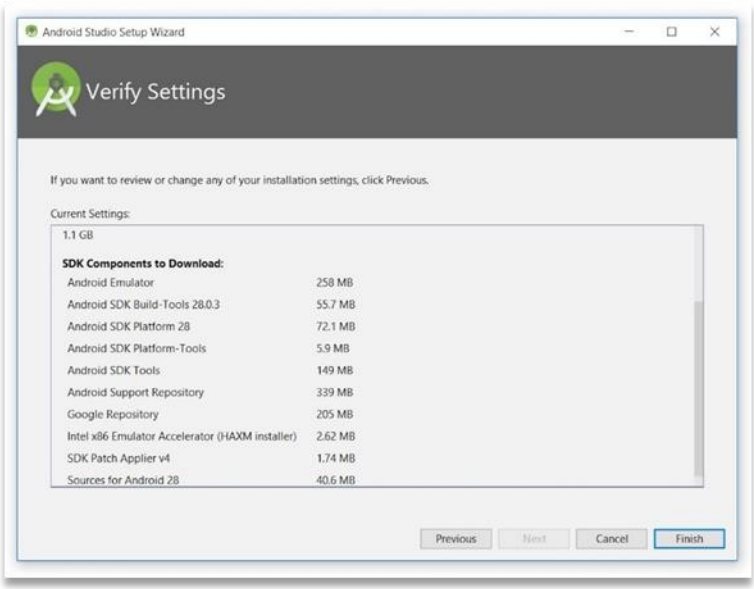
2-4- پس از نصب نوبت به اجرای اندروید استودیو می‌رسد. موقعی که نرم‌افزار را اجرا می‌کنیم، صفحه خوش‌آمدگویی برای ما ظاهر می‌شود. با کلیک روی دکمه Next به مرحله بعد وارد می‌شویم.

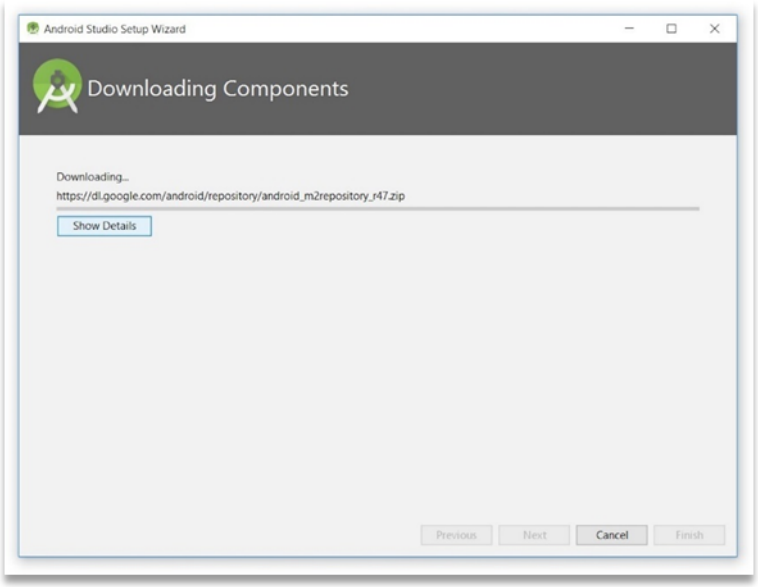


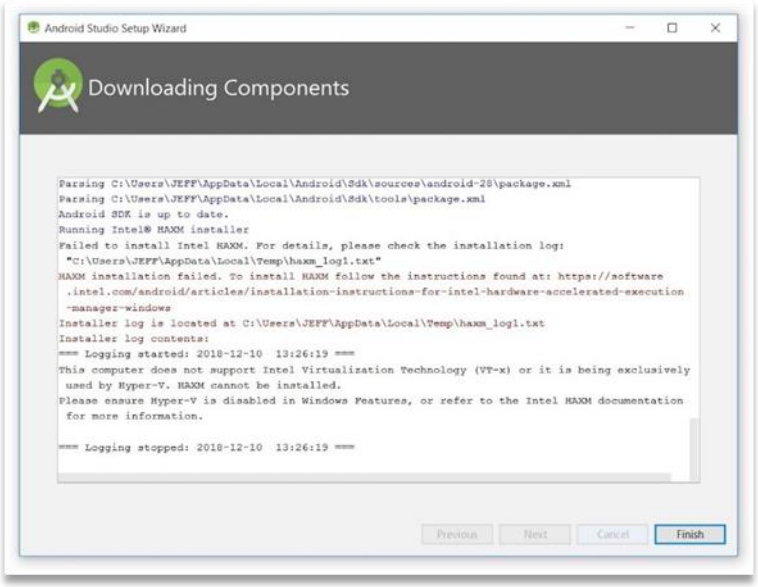
2-5- در پنجره بعدی، نرم‌افزار نوع نصب را از ما می‌پرسد که باید یکی از دو نوع استاندارد (Standard) و شخصی (Custom) را انتخاب کنیم. نوع استاندارد همه تنظیمات پیش‌فرض را دارد؛ اما در نوع شخصی می‌توان تنظیمات را تغییر داد. پیشنهاد می‌شود نوع نصب استاندارد را انتخاب کنید. روی دکمه Next کلیک می‌کنیم.



2-6- در این مرحله پنجره‌ای باز می‌شود که شامل اطلاعاتی برای دانلود Android SDK (software development kit) است. Android SDK شامل مجموعه‌ای از ابزار توسعه برای پیاده‌سازی نرم‌افزارهای اندرویدی است. این مجموعه کتابخانه‌ها، نمونه‌ها و آموزش‌هایی را شامل می‌شود که توسط گوگل ارائه شده است. برای نوشتن و پیاده‌سازی نرم‌افزار در بستر اندروید باید این مجموعه را دانلود کنیم. گوگل کار را برای ما بسیار راحت کرده به‌طوری‌که با کلیک روی دکمه Finish همه فایل‌های مورد نیاز از این مجموعه دانلود می‌شوند و در جای مناسب قرار می‌گیرند.







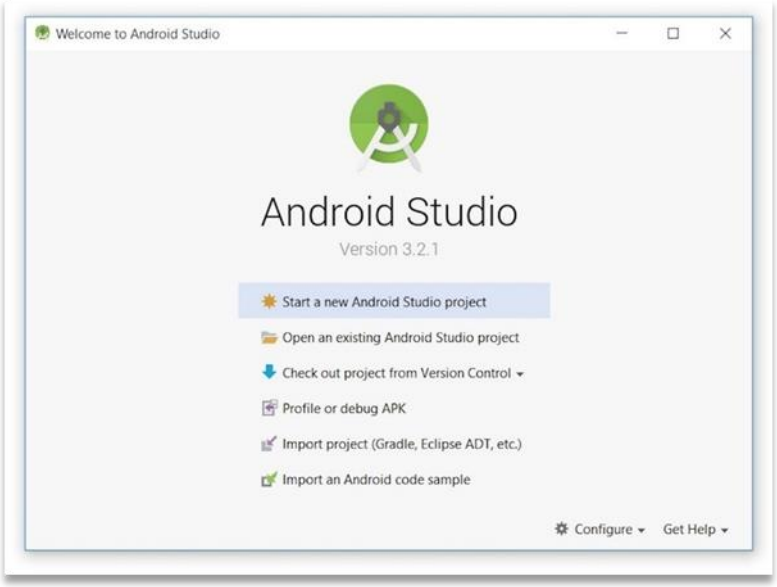
همان‌طور که در شکل بالا مشخص است، صبر می‌کنیم که مراحل دانلود و نصب SDK به پایان برسد. بعد از پایان کار، دکمه Finish را می‌زنیم و این مرحله نیز به پایان می‌رسد.

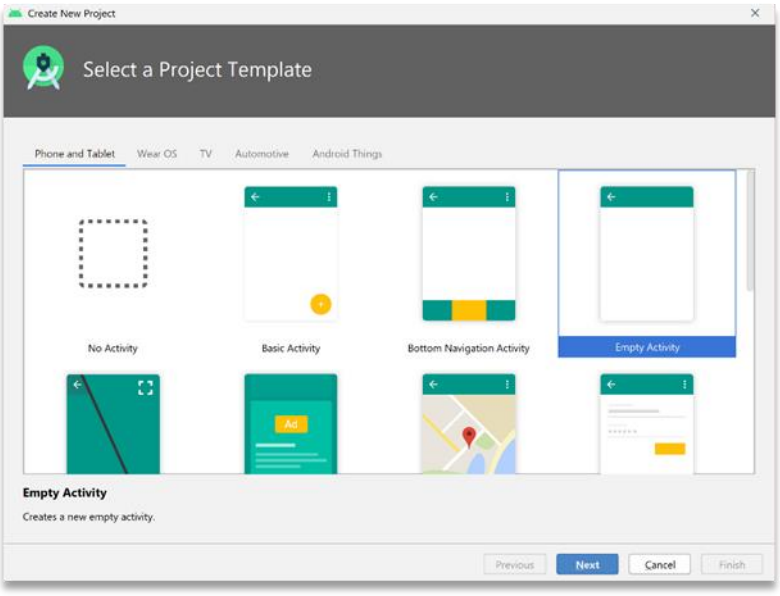
3. نوشتن اولین برنامه (Hello, World!)

حال نوبت به آخرین مرحله ساخت یک نرم‌افزار اندرویدی می‌رسد. ساده‌ترین نرم‌افزاری که می‌توان نوشت نرم‌افزاری است که پیامی با عنوان Hello, World! روی صفحه گوشی دارد. بدین منظور مراحل زیر را دنبال می‌کنیم.

1-3 بعد از گذر از مرحله قبل، پنجره‌ای جدید باز شده که شامل اطلاعاتی راجع به پروژه‌ها است. از آنجا که می‌خواهیم اولین پروژه خود را بنویسیم، روی گزینه اول کلیک می‌کنیم.

2-3 بعد از این مرحله، پنجره جدیدی باز می‌شود که اطلاعاتی درباره الگوی پروژه به ما می‌دهد. در این قسمت می‌توان الگوی نرم‌افزار را انتخاب کرد. از آنجایی که نرم‌افزار موردنظر ما یک نرم‌افزار بسیار ساده است، باید الگوی Empty Activiy را انتخاب کنیم.

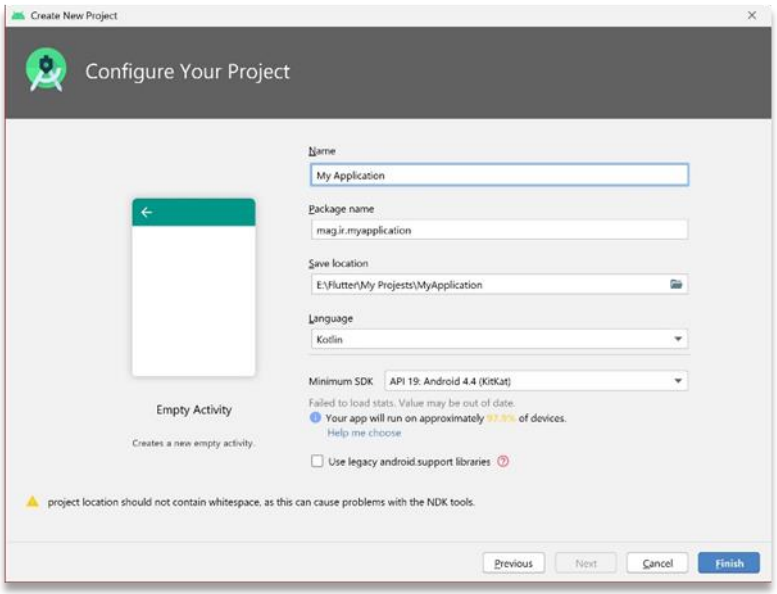




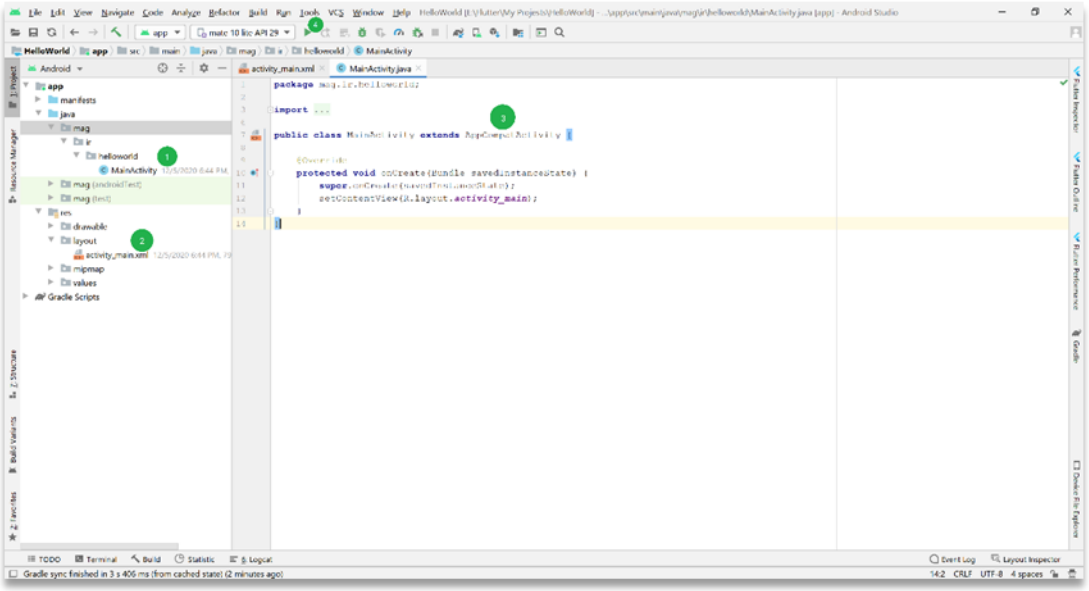
3-3 بعد از این مرحله، پنجره جدیدی باز شده که شامل اطلاعاتی راجع به پروژه است. اولین قسمت Application Name نام دارد که می‌توان نام پروژه را در این قسمت نوشت. نام پروژه هر مقدار دلخواهی می‌تواند باشد. قسمت بعدی Package Name نام دارد. در این قسمت نام بسته نرم‌افزار را می‌نویسیم. بسته نرم‌افزار برای هر پروژه‌ای مقداری منحصربه‌فرد دارد. قسمت بعدی محل ذخیره پروژه را نشان می‌دهد. در قسمت Language می‌توان زبان برنامه‌نویسی موردنظر خود را انتخاب کنیم. می‌توان بین دو زبان جاوا و کاتلین یک مورد را انتخاب کرد که ما جاوا را انتخاب می‌کنیم. در قسمت minimum SDK می‌توان پایین‌ترین SDK را برای پیاده‌سازی پروژه انتخاب کرد. هر چه این مقدار پایین‌تر باشد، تعداد گوشی بیشتری نرم‌افزار ما را پشتیبانی خواهد کرد.

توجه: به دلیل از رده خارج شدن اندرویدهای قدیمی، برای پیشگیری از بروز مشکلات، توصیه می‌شود مقدار این قسمت را روی API 19 قرار دهید.

با زدن روی دکمه Finish وارد مرحله بعد می‌شویم.



4-3 در این قسمت محتوای کلی پروژه در اندروید استودیو نشان داده می‌شود. در عکس زیر قسمت‌های مختلف نرم‌افزار شماره‌گذاری شده است که هر یک در ادامه توضیح داده می‌شوند.



بخش 1. در این قسمت همه فعالیت‌ها یا Activity ها قرار دارند. فعالیت بنیادی‌ترین مؤلفه در برنامه‌نویسی اندروید است. هر فعالیت یک چرخه حیات دارد که دارای آغاز و پایان منحصر به خود است.

بخش 2. در این قسمت مقادیر لایه‌ها یا Layout ها قرار دارند. هر لایه با پسوند XML ذخیره شده است که نقش رابط کاربری را در برنامه‌نویسی اندروید ایفا می‌کند.

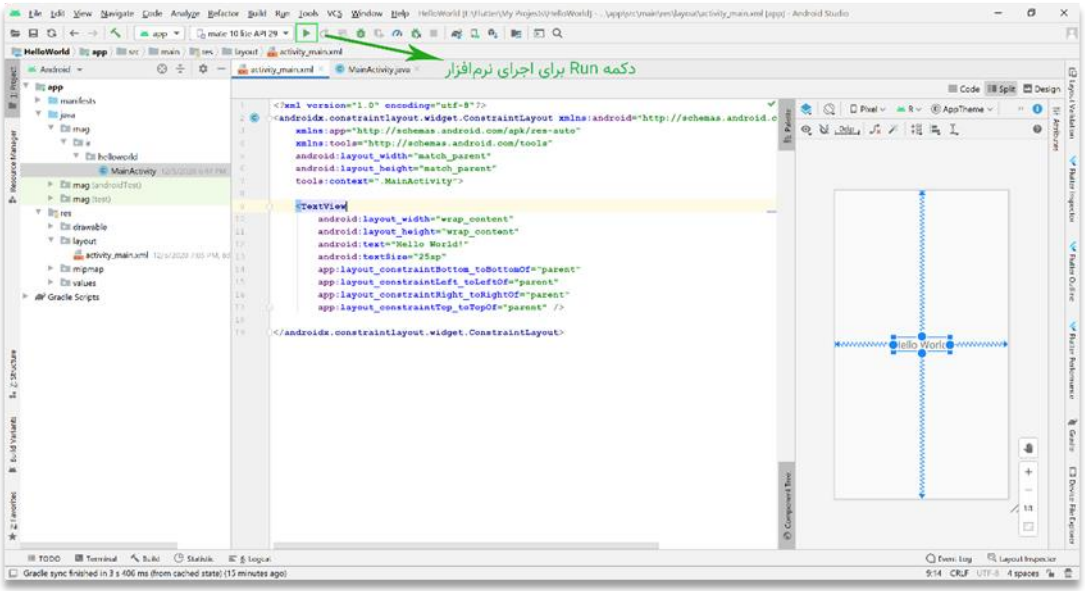
بخش 3. اصلی‌ترین فعالیت در اندروید، MainActivity است؛ یعنی هنگامی که نرم‌افزار روی گوشی اجرا می‌شود، این قسمت فراخوانی شده و کدهای آن اجرا می‌شود. این فعالیت در این پروژه به زبان جاوا نوشته شده است. در واقع هر فعالیتی یک کلاس است که از کلاس معروف AppCompatActivity به ارث برده شده است. در هر فعالیت روشی تحت عنوان OnCreate وجود دارد که به محض بالا آمدن نرم‌افزار اجرا می‌شود. همچنین در خط بعدی لایه مربوط به این فعالیت که activity_main نام دارد و حاوی اطلاعاتی پیرامون رابط کاربری صفحه اصلی یا فعالیت اصلی است را دربر می‌گیرد.

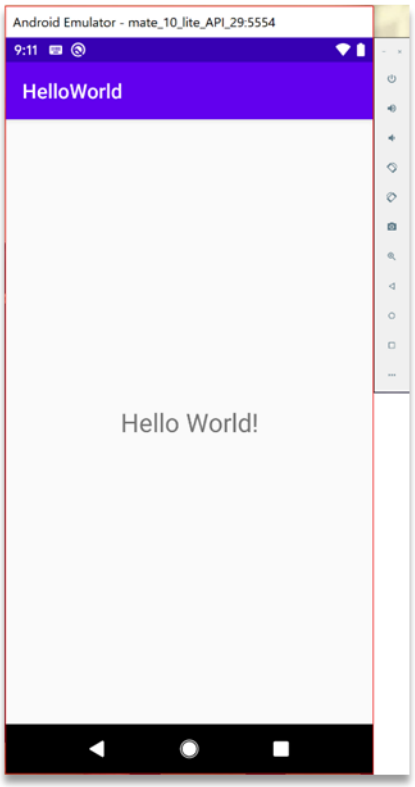
بخش 4. با زدن بر این دکمه، نرم‌افزار روی گوشی یا شبیه‌ساز گوشی اجرا می‌شود.

5-3 در این قسمت لایه مربوط به فعالیت اصلی که شامل اطلاعاتی مربوط به رابط کاربری است، نشان داده می‌شود که به زبان XML نوشته می‌شود.

این لایه شامل یک ConstraintLayout است که وظیفه لایه‌بندی صفحه گوشی را به عهده دارد و یکی از چندین کلاس لایه‌بندی به شمار می‌آید. همچنین یک TextView که متن را در برنامه‌نویسی اندروید نشان می‌دهد، است. همان‌طور که مشخص است متن Hello World در لایه نوشته شده است. دیگر مشخصات ذکر شده درون تگ‌ها هم به‌راحتی قابل‌تشخیص است.

6-3- با زدن روی دکمه اجرا یا Run برنامه روی شبیه‌ساز یا گوشی متصل به سیستم بالا می‌آید و Hello World به ما نشان داده می‌شود.





گرفتن اولین خروجی با فلاتر

1- نصب *Flutter* در ویندوز

وب‌سایت *flutter* سندهای آموزشی برای انواع سیستم‌عامل‌ها را آماده کرده است. ما در این آموزش فرض می‌کنیم تمامی روال نصب و ... روی سیستم‌عامل ویندوز می‌باشد.

1-1- نیازهای سیستم

برای نصب و اجرای *Flutter*، محیط توسعه شما باید این حداقل شرایط را داشته باشد:

- سیستم‌عامل: *Windows 7 SP1* یا بالاتر (64 بیتی)، مبتنی بر *x86-64*
- فضای حافظه: 1.32 گیگابایت (فضای دیسک برای *IDE/Tools* را شامل نمی شود).
- ابزار (*Tools*): *Flutter* به در دسترس بودن این ابزارها در محیط شما بستگی دارد.
  - *Windows PowerShell 5.0* یا جدیدتر (این از قبل به همراه ویندوز 10 نصب شده است).
  - *Git* برای ویندوز *2.x*، با گزینه *Use Git from the Windows Command Prompt*.

اگر *Git* برای ویندوز از قبل نصب شده باشد، مطمئن شوید که می توانید دستورات *git* را از خط فرمان یا *PowerShell* اجرا کنید.

1-2- دانلود *Flutter SDK*

1- بسته نرم‌افزاری زیر را بارگیری کنید تا آخرین نسخه پایدار *Flutter SDK* را دریافت کنید:

```
https://storage.googleapis.com/flutter_infra/releases/stable/windows/flutter_windows_1.22.6-stable.zip
```

برای سایر کانال‌های انتشار و ساخته‌های قدیمی‌تر، به صفحه نسخه‌های *SDK* رجوع کنید.

2- فایل *zip* را استخراج کرده و پوشه *Flutter* موجود را در محل نصب مورد نظر *Flutter SDK* قرار دهید (به عنوان مثال *C:\src\flutter*).

هشدار: Flutter را در دایرکتوری مانند C:\Program Files\ که به امتیازات بالاتر نیاز دارد نصب نکنید.

اگر نمی‌خواهید نسخه ثابت بسته نرم‌افزاری را نصب کنید، می توانید از مراحل 1 و 2 صرف نظر کنید، درعوض

```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

کد منبع را از *repo Flutter* در *GitHub* دریافت کنید و در صورت لزوم شاخه‌ها یا برچسب‌ها را تغییر دهید. مثلا:

اکنون آماده اجرای دستورات *Flutter* در کنسول *Flutter* هستید.

3-1- به‌روزرسانی مسیر

اگر می‌خواهید دستورات *Flutter* را در کنسول معمولی ویندوز اجرا کنید، این مراحل را برای اضافه کردن *Flutter* به متغیر محیط *PATH* (*PATH environment variable*) انجام دهید:

- از نوار جستجوی شروع، "*env*" را وارد کرده و ویرایش متغیرهای محیط (*Edit environment variables for your account*) را برای حساب خود انتخاب کنید.
- در بخش متغیرهای کاربر (*User variables*)، چک کنید که آیا ورودی‌ای به نام *Path* وجود دارد یا نه:
  - اگر ورودی وجود دارد، مسیر کامل *flutter\bin* را با استفاده از *;* به عنوان جداکننده از مقادیر موجود اضافه کنید.
  - اگر ورودی وجود ندارد، یک متغیر کاربر جدید با نام *Path* با مسیر کامل *flutter\bin* به عنوان مقدار ایجاد کنید.

برای اینکه این تغییرات اعمال شود باید پنجره‌های کنسول موجود را ببندید و دوباره باز کنید.

یادداشت: از انتشار نسخه 1.19.0 برنامه Flutter، Flutter SDK حاوی دستور dart در کنار دستور flutter است تا بتوانید برنامه های خط فرمان Dart را با سهولت بیشتری اجرا کنید. بارگیری Flutter SDK همچنین نسخه سازگار Dart را بارگیری می‌کند، اما اگر Dart SDK را به طور جداگانه بارگیری کرده‌اید، مطمئن شوید که نسخه Flutter دارت در اول مسیر یا path شما قرار دارد، زیرا ممکن است این دو نسخه با هم سازگار نباشند. دستور زیر (در macOS، linux و chrome OS) به شما می‌گوید که آیا دستورات flutter و dart از همان شاخه bin منشا می‌گیرند و بنابراین سازگار هستند. (بعضی از نسخه‌های ویندوز از دستور مشابهی پشتیبانی می‌کنند.)

```
$ which flutter dart
/path-to-flutter-sdk/bin/flutter
/usr/local/bin/dart
```

همانطور که در بالا نشان داده شده است، این دو دستور از یک دایرکتوری bin نیستند. مسیر خود را به روز کنید تا از دستورات path-to-flutter-sdk/bin/ قبل از دستورات از /usr/local/bin استفاده کنید (در این حالت). پس از به‌روزرسانی shell خود برای اعمال تغییر، با اجرای دستور which یا where دوباره باید نشان دهید که دستورات flutter و دارت اکنون از همان دایرکتوری آمده‌اند.

```
$ which flutter dart
/path-to-flutter-sdk/bin/flutter
/path-to-flutter-sdk/bin/dart
```

برای کسب اطلاعات بیشتر در مورد دستور dart، از خط فرمان dart -h را اجرا کنید.

1-4- اجرای *flutter doctor*

از پنجره کنسولی که فهرست *Flutter* در آن قرار دارد (به بالا مراجعه کنید)، دستور زیر را اجرا کنید تا ببینید آیا

```
C:\src\flutter>flutter doctor
```

برای تکمیل تنظیمات به سیستم عامل وابستگی دارید یا خیر:

این دستور محیط شما را بررسی می‌کند و گزارشی از وضعیت نصب *Flutter* را نمایش می‌دهد.

1-5- راه‌اندازی *Android*

1-5-1- نصب *Android Studio*

هشدار: ابزار flutter از Google Analytics برای گزارش ناشناس آمار استفاده از ویژگی‌ها و گزارش های خرابی اساسی استفاده می‌کند. این داده ها برای کمک به بهبود ابزارهای Flutter در طول زمان استفاده می‌شوند.

تجزیه و تحلیل ابزار Flutter در اولین بار ارسال نمی‌شود. برای غیرفعال کردن گزارش، flutter config --no-analytics را تایپ کنید. برای نمایش تنظیمات فعلی، flutter config را تایپ کنید. در صورت انصراف از تجزیه و تحلیل، یک رویداد انصراف ارسال می‌شود و سپس هیچ اطلاعات دیگری توسط ابزار Flutter ارسال نمی‌شود. با بارگیری Flutter SDK، شما با شرایط خدمات Google موافقت می‌کنید.

توجه: خط مشی رازداری Google نحوه کار با داده در این سرویس را توصیف می‌کند. علاوه بر این، Flutter شامل Dart SDK است که ممکن است معیارهای استفاده و گزارش خرابی را به Google ارسال کند.

یادداشت: Flutter برای تأمین وابستگی‌های سیستم‌عامل Android خود به نصب کامل Android Studio متکی است. با این حال، می‌توانید برنامه‌های Flutter خود را در بسیاری از ویراستارهای دیگر بنویسید.

1- *Android Studio* را بارگیری و نصب کنید.

2- *Android Studio* را شروع کرده و از «*Android Studio Setup Wizard*» استفاده کنید. با این کار جدیدترین *Android SDK*، *Android SDK Command-line Tools* و *Android SDK Build-Tools* نصب می شود، که هنگام توسعه برای *Android* مورد نیاز *Flutter* هستند.

1-5-2- دستگاه *Android* خود را تنظیم کنید

برای آماده‌سازی اجرا و آزمایش برنامه *Flutter* خود در دستگاه *Android*، به دستگاه *Android* مجهز به *Android* نسخه 4.1 (سطح *API 16*) یا بالاتر نیاز دارید.

1- گزینه‌های برنامه‌نویس (*developer options*) و اشکال زدایی *USB* (*USB debugging*) را در دستگاه خود فعال کنید. دستورالعمل‌های دقیق در اسناد *Android* موجود است.

2- فقط در *Windows*: درایور *USB Google* را نصب کنید.

3- با استفاده از کابل *USB*، تلفن خود را به رایانه وصل کنید. اگر دستگاه شما اجازه دسترسی خواست، به رایانه خود اجازه دسترسی به دستگاه خود را بدهید.

4- در ترمینال، دستور *flutter devices* را اجرا کنید تا بررسی کنید *Flutter* دستگاه *Android* متصل شما را تشخیص می‌دهد. به طور پیش‌فرض، *Flutter* از نسخه *Android SDK* که در آن ابزار *adb* شما مستقر است استفاده

می‌کند. اگر می‌خواهید *Flutter* از نصب دیگری از *Android SDK* استفاده کند، باید متغیر محیط *ANDROID_SDK_ROOT* را در آن فهرست نصب تنظیم کنید.

1-5-3- شبیه‌ساز *Android* را تنظیم کنید

برای آماده‌سازی اجرا و آزمایش برنامه *Flutter* خود بر روی شبیه‌ساز *Android*، این مراحل را دنبال کنید:

1- شتاب *VM* (*VM acceleration*) را روی دستگاه خود فعال کنید.

2- *Android Studio* را راه‌اندازی کنید، روی نماد *AVD Manager* کلیک کنید و ایجاد دستگاه مجازی (*Create Virtual Device*) را انتخاب کنید.

- در نسخه های قدیمی *Android Studio*، باید در عوض *Android Studio>Tools>Android>AVD Manager* را انتخاب کرده و ایجاد دستگاه مجازی را انتخاب کنید. (زیر منوی *Android* فقط وقتی در داخل یک پروژه *Android* هستیم، وجود دارد).
- اگر پروژه‌ای باز ندارید، می‌توانید *Configure>AVD Manager* را انتخاب کرده و *Create Virtual Device* را انتخاب کنید.

3- تعریف دستگاه را انتخاب کرده و گزینه *Next* را انتخاب کنید.

4- برای نسخه‌های *Android* که می‌خواهید شبیه‌سازی کنید، یک یا چند تصویر (*image*) سیستم را انتخاب کنید و گزینه *Next* را انتخاب کنید. تصویر *x86* یا *x86_64* توصیه می شود.

5- در بخش *Emulated Performance*، گزینه *Hardware - GLES 2.0* را برای فعال کردن شتاب سخت‌افزاری انتخاب کنید.

6- تأیید کنید که پیکربندی *AVD* صحیح است و *Finish* را انتخاب کنید.

7- در *Android Virtual Device Manager*، روی *run* در نوار ابزار کلیک کنید. شبیه‌ساز راه‌اندازی شده و *canvas* پیش‌فرض نسخه و سیستم عامل انتخابی شما را نمایش می‌دهد.

2- تنظیم ویرایشگر

با استفاده از هر ویرایش‌گر متنی که با ابزارهای خط فرمان ما ترکیب شده است می‌توانید با *Flutter* برنامه بنویسید. با این حال، توصیه می‌کنیم برای تجربه‌ای بهتر از یکی از افزونه‌های ویرایش‌گر ما استفاده کنید. این افزونه‌ها تکمیل کد، برجسته‌سازی نحو (*syntax highlighting*)، کمک‌های ویرایش ویجت، پشتیبانی از اجرا و رفع اشکال و موارد دیگر را برای شما فراهم می‌کنند. مراحل زیر را برای افزودن یک افزونه ویرایش‌گر برای *Android Studio* و *IntelliJ* دنبال کنید.

2-1- نصب *Android*

Android Studio یک تجربه کامل و یکپارچه *IDE* را برای *Flutter* ارائه می‌دهد.

- *Android Studio*، نسخه 3.0 یا بالاتر

همچنین می توانید از *IntelliJ* استفاده کنید:

- *IntelliJ IDEA Community*، نسخه 2017.1 یا بالاتر
- *IntelliJ IDEA Ultimate*، نسخه 2017.1 یا بالاتر

2-2- نصب *flutter* و افزونه *Dart*

برای نصب این موارد:

- *Android Studio* را شروع کنید.
- *plugin preferences* را باز کنید (*Configure>Plugins* از *v3.6.3.0* یا بالاتر).
- *Flutter plugin* را انتخاب کرده و روی *Install* کلیک کنید.
- در صورت درخواست نصب *Dart plugin*، بله را کلیک کنید.
- وقتی از شما خواسته شد، روی *Restart* کلیک کنید.

یادداشت: قبل از نسخه v3.6.3.0، دسترسی به تنظیمات plugin به صورت زیر بود:

1- plugin preferences را باز کنید (در MacOS به بخش Preferences>Plugins بروید؛ در Windows & Linux به File>Settings>Plugins بروید).

2- Marketplace را انتخاب کنید، Flutter plugin را انتخاب کنید و روی Install کلیک کنید.

3- آزمون و خطا

در این صفحه نحوه ایجاد یک برنامه *Flutter* جدید از الگوها، اجرای آن و تجربه "بارگیری مجدد داغ" (*hot reload*) پس از ایجاد تغییر در برنامه، توضیح داده شده است.

3-1- ساخت *application*

1- *IDE* را باز کرده و یک پروژه جدید *Flutter* بسازید.

2- *Flutter application* را به عنوان نوع پروژه انتخاب کنید. سپس روی *Next* کلیک کنید.

3- تأیید کنید که مسیر *Flutter SDK* موقعیت *SDK* را مشخص می‌کند (اگر قسمت متن خالی است، *Install SDK* را انتخاب کنید).

4- نام پروژه را وارد کنید (به عنوان مثال، *myapp*). سپس روی *Next* کلیک کنید.

5- روی *finish* کلیک کنید.

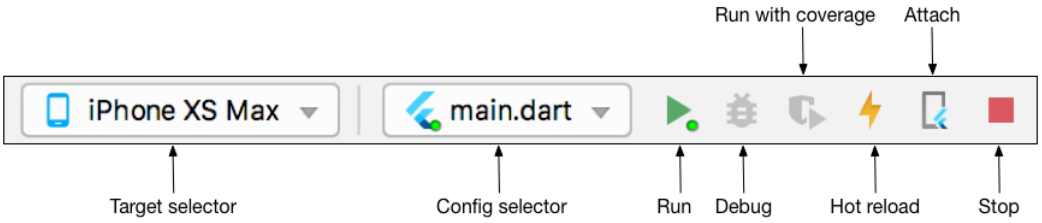
6- منتظر بمانید تا *Android Studio*، *SDK* را نصب کرده و پروژه را ایجاد کند.

یادداشت: هنگام ایجاد یک برنامه Flutter جدید، برخی از افزونه‌های Flutter IDE یک نام دامنه شرکت را به ترتیب معکوس درخواست می‌کنند، چیزی مانند com.example. هنگام انتشار برنامه، نام دامنه شرکت و نام پروژه به عنوان نام بسته (package) برای Android (Bundle ID برای iOS) استفاده می‌شود. اگر فکر می‌کنید ممکن است برنامه منتشر شود، بهتر است اکنون نام بسته را مشخص کنید. پس از انتشار برنامه نمی‌توان نام بسته را تغییر داد، بنابراین نام را منحصر به فرد انتخاب کنید.

دستورات بالا یک دایرکتوری پروژه *Flutter* به نام *myapp* ایجاد می‌کنند که شامل یک برنامه آزمایشی ساده است که از *Material components* استفاده می‌کند.

3-2- اجرای *application*

1- نوار ابزار اصلی (*main toolbar*) *Android Studio* را بیابید:



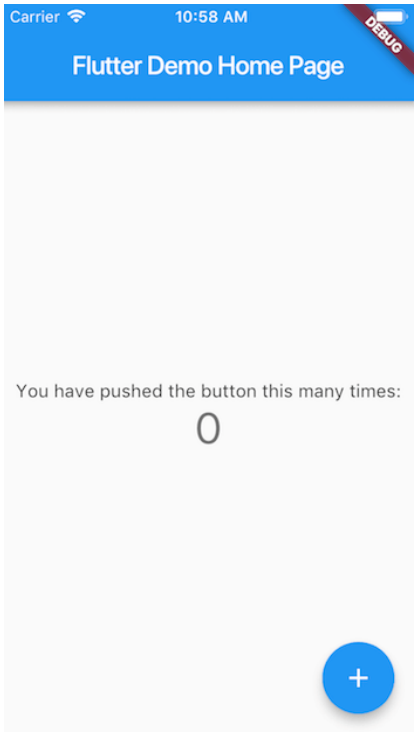
نکته: کد برنامه شما در lib/main.dart قرار دارد. برای توصیف سطح بالا (high-level) درباره آنچه هر بلوک کد انجام می‌دهد، به نظرات (comments) بالای آن فایل مراجعه کنید.

2- در انتخاب‌گر هدف (*target selector*)، دستگاه *Android* را برای اجرای برنامه انتخاب کنید. اگر هیچ دستگاهی موجود نباشد، *Tools>Android>AVD Manager* را انتخاب کرده و یکی را در آن‌جا ایجاد کنید. برای جزئیات بیشتر، به مدیریت *AVD*ها مراجعه کنید.

3- روی نماد اجرا (*Run icon*) در نوار ابزار کلیک کنید، یا مورد منو *Run>Run* را انتخاب کنید.

هشدار: هنگام راه‌اندازی برنامه خود در سیستم‌عامل Mac، اگر این خطا را مشاهده کردید: نمی توانید به Lockdownd متصل شوید ، کد خطا -17. مطمئن شوید که رایانه خود را به فهرست دستگاه‌های مورد اعتماد اضافه کرده‌اید.

پس از اتمام ساخت برنامه، برنامه شروع‌کننده (*starter*) را در دستگاه خود مشاهده خواهید کرد.



3-3- *hot reload* را امتحان کنید

Flutter یک چرخه توسعه سریع (*fast development cycle*) با *Stateful Hot Reload* را ارائه می‌دهد که توانایی بارگیری مجدد کد یک برنامه زنده در حال اجرا بدون راه‌اندازی مجدد یا از دست دادن حالت برنامه می‌باشد. تغییر در منبع برنامه، به *IDE* یا ابزار خط فرمان خود بگویید که می‌خواهید *hot reload* انجام شود و تغییر در شبیه‌ساز یا دستگاه خود را مشاهده کنید.

1- *lib/main.dart* را باز کنید.

2- این رشته را تغییر دهید.

```
'You have pushed the button this many times'
```

```
'You have clicked the button this many times'
```

3- تغییرات خود را ذخیره کنید: *Save All* را فراخوانی کنید یا بر روی *Hot Reload* کلیک کنید.
رشته به روز شده را بلافاصله در برنامه در حال اجرا مشاهده خواهید کرد.

هشدار: برنامه خود را متوقف نکنید. اجازه دهید برنامه شما اجرا شود.

4- ساخت اولین برنامه با *Flutter*

این قسمت راهنمای ایجاد اولین برنامه *Flutter* شما است. اگر با کدنویسی شی‌گرا و مفاهیم اساسی برنامه‌نویسی مانند متغیرها، حلقه‌ها و شرط‌ها آشنا هستید، می‌توانید این آموزش را کامل متوجه شوید. شما به تجربه قبلی پیرامون برنامه‌نویسی دارت، موبایل یا وب نیاز ندارید.

4-1- مثال اول: ساخت برنامه *Hello World!*

با استفاده از دستورالعمل‌های شروع به کار با اولین برنامه *Flutter*، یک برنامه *Flutter* ساده و الگوی ایجاد شده ایجاد کنید. نام پروژه را *startup_namer* بگذارید (به جای *flutter_app*).

نکته: اگر "New Flutter Project" را به عنوان گزینه‌ای در IDE خود نمی‌بینید، مطمئن شوید که افزونه‌های Flutter و Dart را نصب کرده‌اید.

شما باید ابتدا فایل *lib/main.dart* را ویرایش کنید، جایی که کد دارت در آن موجود است.

1- محتوای فایل *lib/main.dart* را با کد زیر جایگزین کنید. همه کدها موجود در *lib/main.dart* را حذف کنید و با کد زیر جایگزین کنید. این کد عبارت "*Hello World*" را در مرکز صفحه نمایش می‌دهد.

```
// Copyright 2018 The Flutter team. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Welcome to Flutter'),
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

2- برنامه را به روشی که *IDE* توصیف می‌کند، اجرا کنید. بسته به دستگاه خود، باید *Android*، *iOS* یا خروجی وب را ببینید.

خروجی اندروید

خروجی IOS

نکته: اولین باری که برنامه را روی دستگاه فیزیکی اجرا می‌کنید، بارگیری آن مدتی طول می‌کشد. پس از آن، می‌توانید برای به‌روزرسانی سریع از hot reload استفاده کنید. اگر برنامه در حال اجرا باشد، ذخیره‌سازی نیز عمل hot reload را انجام می‌دهد. هنگام اجرای برنامه به صورت مستقیم از کنسول با استفاده از flutter run، **r** را وارد کنید تا بارگیری مجدد انجام شود.

1-1-4- مشاهدات

- این مثال یک برنامه *Material* ایجاد می‌کند. *Material* یک زبان طراحی بصری است که به صورت استاندارد در موبایل و وب وجود دارد. *Flutter* مجموعه‌ای غنی از ویجت‌های *Material* را ارائه می‌دهد. ایده خوبی است که از عبارت *"Material-Material-Design: true"* در قسمت *flutter* فایل *pubspec.yaml* استفاده کنید. این مورد به شما این امکان را می‌دهد که از ویژگی‌های *Material* بیشتر استفاده کنید، مانند مجموعه آیکن‌های از پیش تعریف شده آن.
- متد *main()* از نماد پیکان (*=>*) استفاده می‌کند. برای توابع یا متدهای یک خطی (*one-line*) از این نماد استفاده کنید.
- این برنامه کلاس *StatelessWidget* را گسترش (*extend*) می‌دهد، که باعث می‌شود برنامه به صورت ویجت ساخته شود. در *Flutter*، تقریباً همه چیز یک ویجت است، از جمله ترازبندی (*alignment*)، لایه‌گذاری (*padding*) و طرح‌بندی (*layout*).
- ویجت *Scaffold*، از کتابخانه *Material*، یک نوار برنامه (*app bar*) پیش‌فرض و یک ویژگی بدنه (*property body*) را ارائه می‌دهد که درخت ویجت (*widget tree*) را برای صفحه اصلی نگه می‌دارد. زیرشاخه ویجت می تواند کاملاً پیچیده باشد.

- وظیفه اصلی ویجت ارائه متد *build()* است که نحوه نمایش ویجت را از نظر سایر ویجت‌های سطح پایین‌تر توصیف می‌کند.
- بدنه برای این مثال از یک ویجت *Center* شامل یک ویجت فرزند (*child*) *Text* تشکیل شده است. ویجت *Center* زیردرخت کوچک ویجت خود را در مرکز صفحه هماهنگ می‌کند.

5- مثال دوم: استفاده از یک بسته خارجی (*external package*)

در این مرحله، شما شروع به استفاده از یک بسته منبع باز (*open source*) به نام *english_words* می‌کنید که شامل چند هزار کلمه انگلیسی پرکاربرد به علاوه برخی از توابع سودمند است. شما می‌توانید بسته *english_words* و همچنین بسیاری از بسته‌های منبع باز دیگر را در وبسایت *pub.dev* پیدا کنید.

1- فایل *pubspec.yaml* دارایی‌ها (*assets*) و وابستگی‌های (*dependencies*) یک برنامه *Flutter* را مدیریت می‌کند. در *pubspec.yaml*، بسته *english_words* (3.1.5 یا بالاتر) را مانند کد زیر به لیست وابستگی‌ها اضافه کنید:

{step1_base → step2_use_package}/pubspec.yaml

```
@@ -8,4 +8,5 @@
 dependencies:
   flutter:
     sdk: flutter
   cupertino_icons: ^0.1.2
+  english_words: ^3.1.5
```

2- هنگام مشاهده فایل *pubspec.yaml* در نمای ویرایش‌گر *Android Studio*، روی *Pub get* کلیک کنید. این گزینه بسته را وارد (*pull*) پروژه شما می‌کند. موارد زیر را باید در کنسول مشاهده کنید:

3- در فایل *lib/main.dart*، بسته جدید را وارد کنید:

```
$ flutter pub get
Running "flutter pub get" in startup_namer...
Process finished with exit code 0
```

lib/main.dart

```
import 'package:flutter/material.dart';
import 'package:english_words/english_words.dart';
```

هنگام تایپ، *Android Studio* به شما پیشنهاد می‌دهد تا کتابخانه‌هایی را وارد پروژه کنید. سپس رشته واردات (*import*) را به رنگ خاکستری نشان می‌دهد تا به شما اطلاع دهد که کتابخانه وارداتی استفاده نشده است (تاکنون).

4- به جای استفاده از رشته "*Hello World*" از بسته کلمات انگلیسی برای تولید متن استفاده کنید:

{step1_base → step2_use_package}/lib/main.dart

```
@@ -9,6 +10,7 @@
 class MyApp extends StatelessWidget {
   @override
   Widget build(BuildContext context) {
+    final wordPair = WordPair.random();
     return MaterialApp(
       title: 'Welcome to Flutter',
       home: Scaffold(
@@ -16,7 +18,7 @@
           title: Text('Welcome to Flutter'),
         ),
         body: Center(
-          child: Text('Hello World'),
+          child: Text(wordPair.asPascalCase),
         ),
       ),
     );
```

یادداشت: "pascal case" (همچنین به عنوان "upper camel case" نیز شناخته می‌شود)، به این معنی است که هر کلمه در رشته، از جمله کلمه اول، با یک حرف بزرگ شروع می‌شود. بنابراین، "uppercamelcase" به "UpperCamelCase" تبدیل می‌شود.

5- اگر برنامه در حال اجرا است، برای به‌روزرسانی برنامه از *hot reload* استفاده کنید. هر بار که روی *hot reload* کلیک می‌کنید یا پروژه را ذخیره می‌کنید، باید یک جفت کلمه متفاوت را که به طور تصادفی انتخاب شده است، در برنامه در حال اجرا مشاهده کنید. این به این دلیل است که تولید جفت کلمات در داخل متد *build* ایجاد می‌شود، که هر بار که *MaterialApp* به رندر نیاز دارد، یا هنگام تغییر پلتفرم در *Flutter Inspector* اجرا می‌شود.

5- یک ویجت *Stateful* اضافه کنید

خروجی اندروید

خروجی IOS

ویجت‌های *stateless* غیرقابل تغییر (*immutable*) هستند، به این معنی که ویژگی‌های آن‌ها نمی‌توانند تغییر کنند و همه مقادیر نهایی (*final*) هستند. ویدجت‌های *statefull* حالتی را حفظ می‌کنند که ممکن است در طول عمر ویجت تغییر کند. پیاده‌سازی یک ویجت *stateful* حداقل به دو کلاس نیاز دارد: 1) کلاس *StatefulWidget* که نمونه‌ای از آن می‌سازیم. 2) کلاس *State*. کلاس *StatefulWidget* به خودی خود تغییرناپذیر است و می‌توان آن را دور ریخت و دوباره تولید کرد، اما کلاس *State* در طول عمر ویجت پابرجاست. در این مرحله، یک ویجت *statefull* را اضافه خواهیم کرد. *RandomWords* که کلاس *State* خود را ایجاد می کند تحت نام *_RandomWordsState*. سپس از *RandomWords* به عنوان یک *child* در داخل ویجت *MyApp stateless* موجود استفاده خواهید کرد.

1- کد *boilerplate* را برای یک ویجت مناسب ایجاد کنید. در فایل *lib/main.dart*، مکان نما را بعد از همه کدها قرار دهید، چند بار *enter* را وارد کنید تا از یک خط تازه شروع شود. در *IDE* خود، شروع به تایپ کلمه *stful* کنید. *editor* از شما می‌پرسد که آیا می‌خواهید یک ویجت *Stateful* ایجاد کنید. دکمه *enter* را فشار دهید تا قبول شود. کد *boilerplate* برای دو کلاس ظاهر می‌شود و مکان‌نما برای شما قرار می‌گیرد تا نام ویجت *stateful* خود را وارد کنید.

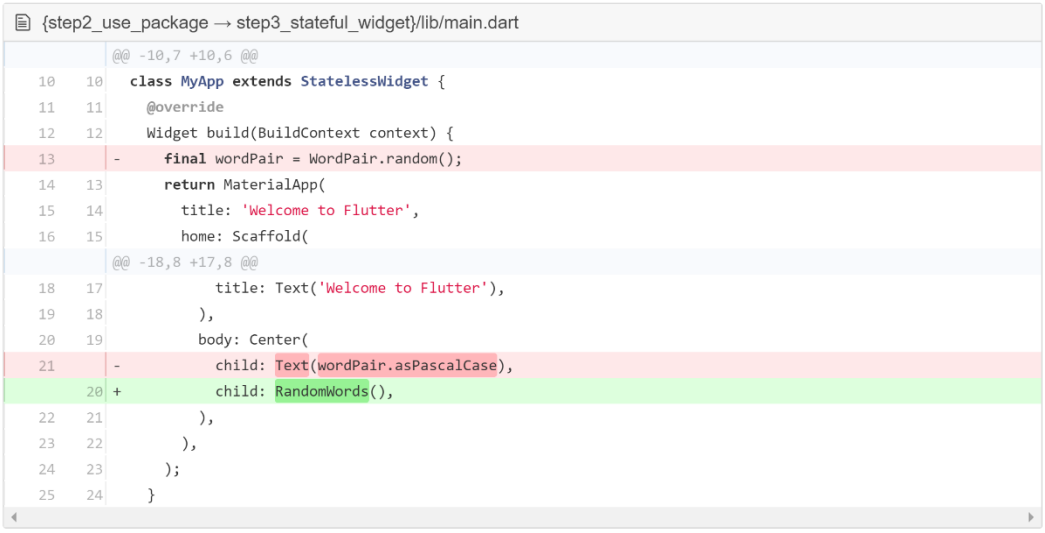
2- *RandomWords* را به عنوان نام ویجت خود وارد کنید. ویجت *RandomWords* در کنار ایجاد کلاس *State* خود کار کوچک دیگری انجام می‌دهد. هنگامی که *RandomWords* را به عنوان ویجت *stateful* وارد کردید، *IDE* به طور خودکار کلاس *State* همراهش را به‌روز می‌کند و نام آن را *_RandomWordsState* می‌گذارد. به طور پیش‌فرض، نام کلاس *State* با یک پیشوند "_" قرار می‌گیرد. *IDE* همچنین به طور خودکار کلاس *State* را برای گسترش (*extend*) از *State<RandomWords>* به روز می‌کند، که نشان می‌دهد که شما از یک کلاس *State* عمومی (*Generic*) استفاده می‌کنید که مخصوص استفاده با *RandomWords* است. بیشتر منطق برنامه در اینجا وجود دارد، این حالت (*state*) را برای ویجت *RandomWords* حفظ می‌کند. این کلاس لیستی از جفت کلمات ایجاد شده را ذخیره می‌کند که با پیمایش کاربر بی‌نهایت رشد می‌کند.

3- متد *build()* در *_RandomWordsState* را به روز کنید:

```
class _RandomWordsState extends State<RandomWords> {
    @override
    Widget build(BuildContext context) {
      final wordPair = WordPair.random();
      return Text(wordPair.asPascalCase);
    }
  }
```

```
class RandomWords extends StatefulWidget {
  @override
  _RandomWordsState createState() => _RandomWordsState();
}

class _RandomWordsState extends State<RandomWords> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

4- با ایجاد تغییراتی که در کد زیر نشان داده شده، کد تولید کلمه را از *MyApp* حذف کنید:



5- برنامه را دوباره راه‌اندازی کنید. برنامه باید مانند گذشته رفتار کند و هر بار که برنامه را بارگیری مجدد یا ذخیره می‌کنید، یک جفت کلمه را به کاربر نشان دهد.

نوشتن اولین برنامه با ری اکت نیتیو

1- مقدمه‌ای بر *React Native*

افراد مختلفی از *React Native* استفاده می‌کنند: از توسعه‌دهندگان پیشرفته *iOS* گرفته تا مبتدیان *React*، تا افرادی که برای اولین بار در حرفه خود برنامه‌نویسی را شروع می‌کنند. برای کار با *React Native*، باید درکی از اصول *JavaScript* داشته باشید. اگر در *JavaScript* تازه وارد هستید، می‌توانید به آدرس

https://developer.mozilla.org/en-US/docs/Web/JavaScript

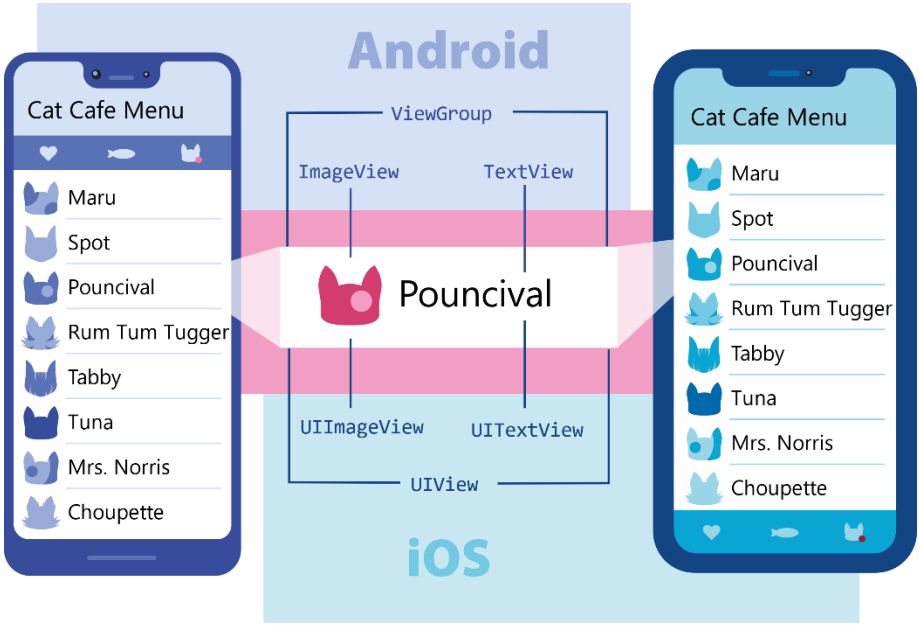
مراجعه نمایید.

1-1- مؤلفه‌های اصلی و بومی (*Core and Native Components*)

React Native یک چارچوب منبع باز برای ساخت برنامه‌های *Android* و *iOS* با استفاده از کتابخانه جاوااسکریپتی *React* و قابلیت‌های بومی پلتفرم برنامه است. با استفاده از *React Native*، از *JavaScript* برای دسترسی به *API*های پلتفرم خود و همچنین توصیف ظاهر و رفتار رابط کاربر (*UI*) خود با استفاده از مؤلفه‌های *React* استفاده می‌کنید: بسته‌های کد قابل استفاده مجدد و قابل نصب.

1-1-1- نماها و توسعه موبایل (*Views and mobile development*)

در توسعه اندروید و *iOS*، یک نمای (*view*) اصلی سازمان رابط کاربری است: یک عنصر مستطیل شکل کوچک روی صفحه که می‌تواند برای نمایش متن، تصاویر یا پاسخ به ورودی کاربر استفاده شود. حتی کوچک‌ترین عناصر بصری یک برنامه، مانند یک خط متن یا یک دکمه، از نوع نما هستند. برخی از انواع نماها می‌توانند شامل نماهای دیگری باشند.



1-1-2- مؤلفه‌های بومی

در توسعه اندروید، شما نماهایی را در *kotlin* یا جاوا می‌نویسید. در توسعه *iOS*، شما از *Swift* یا *Objective-C* استفاده می‌کنید. در *React Native* می‌توانید با استفاده از مولفه‌های *React* این دیدگاه‌ها را با *JavaScript* فراخوانی کنید. هنگام اجرا، *React Native* نماهای مربوط به *Android* و *iOS* را برای آن مؤلفه‌ها ایجاد می‌کند. از آنجا که اجزای *React Native* با همان نماهای *Android* و *iOS* پشتیبانی می‌شوند، برنامه‌های *React Native* مانند سایر برنامه‌ها ظاهر، احساس و عملکرد خوبی دارند. به این اجزای پشتیبانی شده از پلتفرم، اجزای بومی می‌گویند.

React Native به شما امکان می‌دهد که مؤلفه‌های بومی خود را برای *Android* و *iOS* متناسب با نیازهای منحصر به فرد برنامه خود بسازید. *React Native* همچنین شامل مجموعه‌ای از مؤلفه‌های بومی و آماده برای استفاده است که می‌توانید از امروز برای شروع ساخت برنامه خود استفاده کنید. این‌ها مؤلفه‌های اصلی *React Native* هستند.

1-1-3- مؤلفه‌های اصلی

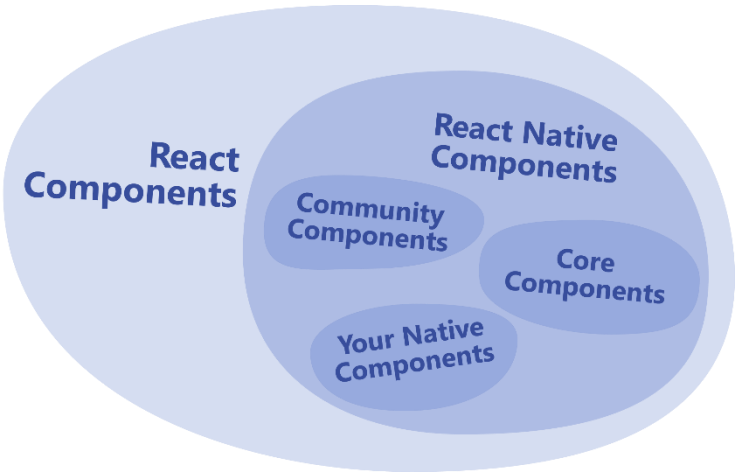
React Native دارای مؤلفه‌های اصلی بسیاری برای همه موارد است، از کنترل فرم (*form control*) گرفته تا شاخص‌های فعالیت (*activity indicators*). همه آن‌ها را می‌توانید از طریق

https://reactnative.dev/docs/components-and-apis

پیدا کنید. شما بیشتر با اجزای اصلی زیر کار خواهید کرد:

تشریح	در وب	نمای *IOS*	نمای اندروید	مؤلفه رابط کاربری در *React Native*
ظرفی (*container*) که از طرح‌بندی *flexBox*، سبک، برخی کنترل‌های لمسی و قابلیت دسترسی پشتیبانی می‌کند.	* * غیر قابل اسکرول	*<UIView>*	*<ViewGroup>*	*<View>*
برای نمایش، سبک‌دهی، رشته‌های متن و حتی کنترل لمس صفحه	*<p>*	*<UITextView>*	*<TextView>*	*<Text>*
انواع مختلف تصاویر را نمایش می‌دهد	*<img>*	*<UIImageView>*	*<ImageView>*	*<Image>*
یک ظرف پیمایش عمومی که می‌تواند شامل چندین مؤلفه و نما باشد	* *	*<UIScrollView>*	*<ScrollView>*	*<ScrollView>*
به کاربر اجازه می دهد متن را وارد کند	*<input type="text">*	*<UITextField>*	*<EditText>*	*<TextInput>*

از آنجا که *React Native* از همان ساختار *API* به عنوان مؤلفه‌های *React* استفاده می‌کند، برای شروع باید *API*های مؤلفه *React* را درک کنید.



2-1- اساس و پایه *React*

React Native روی *React*، یک کتابخانه منبع باز محبوب برای ساخت رابط کاربر با *JavaScript* اجرا می‌شود. برای استفاده بیشتر از *React Native*، خودش به درک *React* کمک می‌کند. ما قصد داریم مفاهیم اصلی را در پس‌زمینه *React* پوشش دهیم:

- مؤلفه‌ها
- *JSX*
- *props*
- *state*

1-2-1- اولین مؤلفه

در ادامه این آموزش *React* از گربه‌ها در مثال‌های خود استفاده می‌کند. موجوداتی دوست‌داشتنی و نزدیک که به نام و کافه احتیاج دارند. در اینجا اولین مؤلفه به نام *Cat* آورده شده است:

```
import React from 'react';
import { Text } from 'react-native';

const Cat = () => {
  return (
    <Text>Hello, I am your cat!</Text>
  );
}

export default Cat;
```

Hello, I am your cat!

نحوه انجام این کار به شرح زیر است: برای تعریف مؤلفه *Cat* خود، ابتدا از واژه کلیدی *import* از *JavaScript* برای *import* کردن *React* و مؤلفه اصلی *Text* استفاده می‌کنیم:

```
import React from 'react';
import { Text } from 'react-native';
```

مؤلفه شما به عنوان یک تابع شروع می‌شود:

```
const Cat = () => {};
```

شما می توانید مؤلفه‌ها را به عنوان طرح اصلی (*blueprint*) در نظر بگیرید. هر چند خروجی یک مؤلفه تابع یک عنصر *React* می‌باشد. عناصر *React* به شما امکان می‌دهند آنچه را که می‌خواهید روی صفحه مشاهده کنید توصیف کنید. در اینجا مؤلفه *Cat* یک عنصر *<Text>* را ارائه می‌دهد. شما می‌توانید مؤلفه عملکرد خود را با استفاده از دستور *export default* از *JavaScript* برای استفاده در سراسر برنامه خود صادر کنید. مانند زیر:

```
const Cat = () => {
  return <Text>Hello, I am your cat!</Text>;
};

export default Cat;
```

اکنون نگاهی دقیق به جمله *return* بیندازیم. عبارت *<Text>Hello, I am your cat!</Text>* از نوعی *Syntax* از *JavaScript* استفاده می‌کند که نوشتن عناصر را راحت می‌کند: *JSX*.

JSX -2-2-1

React و *React Native* از *JSX* استفاده می‌کنند، یک نحو که به شما امکان می‌دهد عناصری را در *JavaScript* بنویسید مانند: *<Text>Hello, I am your cat!</Text>*. اسناد *React* یک راهنمای جامع برای *JSX* دارند که می‌توانید برای کسب اطلاعات بیشتر به آن‌ها مراجعه کنید. از آنجا که *JSX* مبتنی بر جاوااسکریپت است، می‌توانید از متغیرهای داخل آن استفاده کنید. در اینجا شما یک نام برای گربه تحت *name*، اعلام می‌کنید و آن را داخل *{}* در درون *<Text>* جایگذاری می‌کنید.

هر عبارت *JavaScript* بین *{}* کار خواهد کرد، از جمله فراخوانی‌های تابع مانند *{getFullName("Rum", "Tum", "Tugger")}*

3-2-1- مؤلفه‌های شخصی

شما قبلاً با مؤلفه‌های اصلی *React Native* آشنا شده‌اید. *React* به شما امکان می‌دهد این مؤلفه‌ها را درون یکدیگر به صورت تودرتو بنویسید تا مؤلفه‌های جدید ایجاد کنید. این مؤلفه‌های قابل استفاده مجدد (*reusable*) و تودرتو در قلب الگوی *React* قرار دارند. به عنوان مثال، می‌توانید *Text* و *TextInput* را در *view* زیر به صورت تودرتو بنویسید و *React Native* آنها را با هم رندر می‌کند:

```
import React from 'react';
import { Text, TextInput, View } from 'react-native';

const Cat = () => {
  return (
    <View>
      <Text>Hello, I am...</Text>
      <TextInput
        style={{
          height: 40,
          borderColor: 'gray',
          borderWidth: 1
        }}
        defaultValue="Name me!"
      />
    </View>
  );
}

export default Cat;
```

Hello, I am...

Name me!

با استفاده از *<Cat>*

در Android، معمولاً نماهای خود را در داخل LinearLayout، FrameLayout، RelativeLayout و غیره قرار می‌دهید تا نحوه چیدمان فرزندان را روی صفحه تنظیم کنید. در React Native، View از Flexbox برای لایه‌بندی فرزندان خود استفاده می‌کند.

می‌توانید چندین بار و در چند مکان این مؤلفه را بدون تکرار کد خود رندر کنید:

```
import React from 'react';
import { Text, TextInput, View } from 'react-native';

const Cat = () => {
  return (
    <View>
      <Text>I am also a cat!</Text>
    </View>
  );
}

const Cafe = () => {
  return (
    <View>
      <Text>Welcome!</Text>
      <Cat />
      <Cat />
      <Cat />
    </View>
  );
}

export default Cafe;
```

Welcome!
I am also a cat!
I am also a cat!
I am also a cat!

هر مؤلفه‌ای که مؤلفه‌های دیگری را رندر کند، یک مؤلفه اصلی است. در اینجا، *cafe* مؤلفه *parent* یا والد است و هر گربه یک مؤلفه *child* یا فرزند است. می‌توانید به تعداد دلخواه گربه را در کافه خود قرار دهید. هر *<Cat>* یک عنصر منحصر به فرد را رندر می‌کند که می توانید آن را با *props* سفارشی کنید.

props **-4-2-1**

Props مخفف *"Properties"* است. *props* به شما امکان می‌دهد اجزای *React* را سفارشی کنید. به عنوان مثال، در اینجا شما به هر *<Cat>* نام متفاوتی برای رندر *Cat* می‌دهید:

```jsx
import React from 'react';
import { Text, View } from 'react-native';

const Cat = (props) => {
  return (
    <View>
      <Text>Hello, I am {props.name}!</Text>
    </View>
  );
}

const Cafe = () => {
  return (
    <View>
      <Cat name="Maru" />
      <Cat name="Jellylorum" />
      <Cat name="Spot" />
    </View>
  );
}

export default Cafe;
```

Hello, I am Maru!
Hello, I am Jellylorum!
Hello, I am Spot!

اکثر مؤلفه‌های اصلی *React Native* را می توان با *props* نیز سفارشی کرد. به عنوان مثال، هنگام استفاده از تصویر، شما به آن یک *prop* با نام *source* ارسال می‌کنید تا مشخص کند چه تصویری را نشان

```jsx
import React from 'react';
import { Text, View, Image } from 'react-native';

const CatApp = () => {
  return (
    <View>
      <Image
        source={{uri: "https://reactnative.dev/docs/assets/p_cat1.png"}}
        style={{width: 200, height: 200}}
      />
      <Text>Hello, I am your cat!</Text>
    </View>
  );
}

export default CatApp;
```

Hello, I am your cat!

دهد:

Image دارای بسیاری از *props*های مختلف است، از جمله سبک که یک شی *JS* را برای طراحی و طرح‌بندی مربوط به جفت خصیصه-ارزش می پذیرد. شما می‌توانید بسیاری از چیزها را با کمک *props* و مؤلفه‌های اصلی شامل *Image*، *Text* و *View* بسازید! اما برای ساختن چیزی تعاملی، به *state* احتیاج دارید.

State **-5-2-1**

در حالی که می‌توانید از *props*ها به عنوان آرگومان‌هایی استفاده کنید که برای پیکربندی نحوه نمایش اجزا استفاده می‌شود، *state* مانند ذخیره‌سازی اطلاعات شخصی یک مؤلفه است. *state* برای مدیریت داده‌هایی که با گذشت زمان تغییر می‌کنند یا ناشی از تعامل کاربر است مفید می‌باشد. *state* به مؤلفه‌ها حافظه می‌دهد!

به عنوان یک قاعده کلی، هنگام پیکربندی یک مؤلفه برای رندر از props استفاده کنید. برای پیگیری اطلاعات داده‌های مؤلفه‌ای که انتظار دارید با گذشت زمان تغییر کند، از state استفاده کنید.

مثال زیر در یک کافه

گربه اتفاق می‌افتد که در آن دو گربه گرسنه منتظر غذا هستند. گرسنگی آن‌ها، که انتظار داریم با گذشت زمان تغییر کند (برخلاف نام آنها)، به عنوان *state* ذخیره می‌شود. برای تغذیه گربه‌ها، دکمه‌های آن‌ها را باید فشار دهید (که وضعیت آنها را به روز می‌کند). با فراخوانی *React's useState Hook* می‌توانید *state* را به یک مؤلفه اضافه کنید. *hook* نوعی تابع است که به شما امکان می‌دهد ویژگی‌های *React* را به چنگ بیندازید. به عنوان مثال، *useState* یک *hook* است که به شما امکان می‌دهد *state* را به مؤلفه‌های کاربردی اضافه کنید.


```
import React, { useState } from 'react';
```

ابتدا می‌خواهید *useState* را از *import React* کنید:

سپس با فراخوانی *useState* در داخل تابع، *state* مؤلفه را اعلام می‌کنید. در این مثال، *useState* یک متغیر *state* به نام *isHungry* ایجاد می‌کند:

فراخوانی *useState* دو کار انجام می‌دهد:

- این یک "متغیر *state*" با مقدار اولیه ایجاد می‌کند - در این حالت متغیر *state*، *isHungry* نام دارد و

```
const Cat = (props) => {
  const [isHungry, setIsHungry] = useState(true);
  // ...
};
```

مقدار اولیه آن *true* است

- این یک تابع برای تنظیم مقدار متغیر *state* (*setIsHungry*) ایجاد می‌کند.

مهم نیست که از چه اسامی استفاده می‌کنید. اما فکر کردن در مورد الگویی به صورت *[<getter>, <setter>] = useState(<initialValue>)* می‌تواند مفید باشد. بعد مؤلفه اصلی *Button* را اضافه می‌کنید و به آن یک *prop* به نام *onPress* پاس می‌دهید:

```
<Button
  onPress={() => {
    setIsHungry(false);
  }}
  //..
/>
```

حالا، وقتی کسی دکمه را فشار می‌دهد، *onPress* با فراخوانی *setIsHungry(false)* اجرا می‌شود. این متغیر حالت *isHungry* را روی *false* تنظیم می‌کند. وقتی *isHungry* به صورت *false* است، ویژگی *disable* دکمه روی *true* تنظیم شده و عنوان آن نیز تغییر می‌کند:

در نهایت، گربه‌های خود را درون یک مؤلفه کافه قرار دهید:

```
<Button
  //..
  disabled={!isHungry}
  title={isHungry ? 'Pour me some milk, please!' : 'Thank you!'}
/>
```

```
const Cafe = () => {
  return (
    <>
      <Cat name="Munkustrap" />
      <Cat name="Spot" />
    </>
  );
};
```

2- تنظیم محیط توسعه

این قسمت به شما کمک می‌کند تا اولین برنامه *React Native* خود را نصب و بسازید. ساده‌ترین راه برای شروع، کار با *Expo CLI* است. *Expo* مجموعه‌ای از ابزارهای ساخته شده پیرامون *React Native* است و گرچه

```
npm install -g expo-cli
```

از ویژگی‌های بسیاری برخوردار است، اما مهم‌ترین ویژگی در حال حاضر برای ما این است که می‌توان با کمک آن در عرض چند دقیقه یک برنامه *React Native* را نوشت. شما فقط به آخرین نسخه *Node.js* و دستگاه موبایل یا شبیه‌ساز نیاز خواهید داشت. با فرض اینکه شما نسخه *Nodejs 12* یا بالاتر را نصب کرده‌اید، می‌توانید از *npm* برای نصب ابزار خط فرمان *Expo CLI* استفاده کنید:

سپس دستورات زیر را اجرا کنید تا یک پروژه جدید *React Native* به نام "*AwesomeProject*" ایجاد کنید:
با این کار یک سرور توسعه برای شما راه‌اندازی می‌شود.

2-1- اجرای برنامه *React Native* ساخته شده

```
expo init AwesomeProject

cd AwesomeProject
npm start # you can also use: expo start
```

برنامه *Expo client* را در تلفن *iOS* یا *Android* خود نصب کنید و به همان شبکه بی‌سیمی متصل شوید که رایانه‌تان به آن وصل است. در *Android*، از برنامه *Expo* برای اسکن کد *QR* از *terminal* خود برای باز کردن پروژه خود استفاده کنید. در *iOS*، از اسکنر کد *QR* داخلی برنامه دوربین استفاده کنید.

2-2- تغییر برنامه

اکنون که برنامه را با موفقیت اجرا کردید ،اجازه دهید آن را اصلاح کنیم. *App.js* را در ویرایشگر متن مورد نظر خود باز کرده و برخی خطوط را ویرایش کنید. هنگامی که تغییرات را ذخیره کردید، برنامه باید به طور خودکار بارگیری شود. تبریک! شما اولین برنامه *React Native* خود را با موفقیت اجرا و اصلاح کرده‌اید.

2-3- اجرای برنامه خود روی شبیه‌ساز یا دستگاه مجازی

Expo CLI به شما امکان می‌دهد بدون ایجاد محیط توسعه، برنامه *React Native* خود را روی دستگاه فیزیکی اجرا کنید. اگر می خواهید برنامه خود را روی شبیه‌ساز *iOS* یا دستگاه مجازی *Android* اجرا کنید، لطفاً به دستورالعمل‌های "*React Native CLI Quickstart*" مراجعه کنید تا نحوه نصب *Xcode* یا تنظیم محیط توسعه اندروید را بیاموزید. پس از تنظیم این موارد، می‌توانید با اجرای *npm run android* در دستگاه مجازی *Android* یا در *iOS Simulator* با اجرای *npm run ios* (فقط *macOS*) برنامه خود را راه‌اندازی کنید.

3- ساخت برنامه *Hello World!*

React Native مانند *React* است، اما از مؤلفه‌های بومی به جای مؤلفه‌های وب به عنوان بلوک‌های سازنده استفاده می‌کند. بنابراین برای درک ساختار اساسی یک برنامه *React Native*، باید برخی از مفاهیم اساسی *React* را مانند *JSX*، مؤلفه‌ها، *state* و *props* درک کنید. اگر با *React* از قبل آشنا هستید، هنوز باید بعضی از موارد خاص *React-Native* را یاد بگیرید، مانند مؤلفه‌های بومی. این آموزش برای همه مخاطبان است، چه شما تجربه

کدنویسی با *React* داشته باشید و چه نداشته باشید. مطابق با سنت‌های باستانی مردم ما، ابتدا باید برنامه‌ای بسازیم که به غیر از گفتن "*Hello World!*" هیچ کاری نمی‌کند.

1- اول از همه، ما باید *React* را *import* کنیم تا بتوانیم از *JSX* استفاده کنیم که در نهایت به مؤلفه‌های بومی هر سیستم‌عامل تبدیل می‌شود.

2- در خط 2، ما مؤلفه‌های *Text* و *View* را از *react native*، *import* می‌کنیم.

سپس به تابع *HelloWorldApp* برخورد می‌کنیم، که یک مؤلفه عملکردی است و رفتاری مشابه *React* برای وب دارد. این تابع یک مولفه *View* را با برخی از سبک‌ها و متن به عنوان فرزند خود برمی‌گرداند. مؤلفه *Text* به ما امکان رندر کردن متن را می‌دهد، در حالی که مؤلفه *View* یک ظرف (*container*) را ارائه می‌دهد. این ظرف شامل چندین سبک است، بیایید کاری را که هرکدام انجام می‌دهند تجزیه و تحلیل کنیم.

اولین سبکی که وجود دارد، *flex: 1* است. *flex prop* تعیین می‌کند که چگونه *item*ها در فضای در دسترس در امتداد محور اصلی پر می‌شوند. از آنجا که ما فقط یک ظرف داریم، تمامی فضای موجود مؤلفه والد را دربر می‌گیرد. در این حالت، تنها مؤلفه همین است، بنابراین فضای صفحه نمایش موجود را اشغال می‌کند.

سبک بعدی *justifyContent: "center"* است. این سبک فرزندان یک ظرف را در مرکز محور اصلی ظرف تراز می‌کند و در آخر ما *alignItems: "center"* را داریم که فرزندان یک ظرف را در مرکز محور عرضی ظرف تراز می‌کند.

اول از همه، *ES2015* (همچنین به عنوان *ES6* شناخته می‌شود) مجموعه‌ای از پیشرفت‌ها در *JavaScript* است که اکنون بخشی از استاندارد رسمی است، اما هنوز توسط همه مرورگرها پشتیبانی نمی‌شود، بنابراین اغلب هنوز در توسعه وب استفاده نمی‌شود.

مورد غیرمعمول دیگر در این مثال کد *<View><Text>Hello world!</Text></View>* است. این *JSX* است - یک نحو برای تعبیه *XML* در *JavaScript*. بسیاری از چارچوب‌ها از یک زبان الگو اختصاصی استفاده می‌کنند که به شما امکان می‌دهد کدی را در زبان علامت‌گذاری قرار دهید. در *React*، این حالت معکوس است. *JSX* به شما امکان می‌دهد زبان نشانه‌گذاری خود را در داخل کد بنویسید. به نظر می‌رسد *HTML* در وب است، مگر اینکه به جای موارد وب مانند * * یا **، شما از مؤلفه‌های *React* استفاده می‌کنید. در این حالت، *<Text>* یک مؤلفه اصلی است که مقداری متن را نمایش می‌دهد و *View* مانند * * یا ** است.

1-3- مؤلفه‌ها

این کد *HelloWorldApp* را به عنوان یک مؤلفه جدید تعریف می‌کند. هنگامی که در حال ساخت یک برنامه *React Native* هستید، مؤلفه‌های جدید زیادی تولید خواهید کرد. هر چیزی که روی صفحه مشاهده می‌کنید نوعی مؤلفه است.

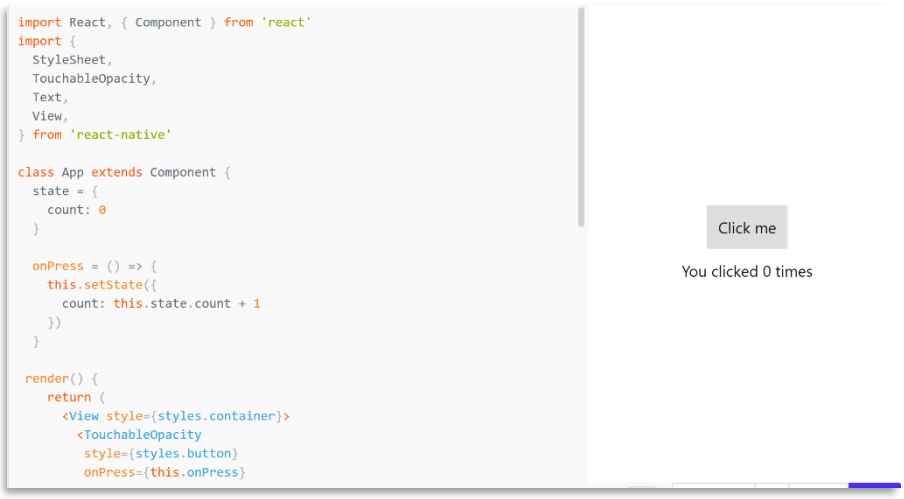
2-3- *Props*

بیشتر مؤلفه‌ها هنگام ایجاد، با پارامترهای مختلف می‌توانند سفارشی شوند. این پارامترهای ایجاد را *props* می‌نامند. مؤلفه‌های شخصی شما همچنین می‌توانند از *props* استفاده کنند. با این کار می‌توانید یک مؤلفه واحد درست کنید که در مکان‌های مختلف برنامه شما استفاده می‌شود.

استفاده از *name* به عنوان یک *props* به ما امکان می‌دهد تا مؤلفه *Greeting* را سفارشی کنیم، بنابراین می‌توانیم از آن مؤلفه برای هر یک از *Greeting*ها استفاده مجدد کنیم. در این مثال از مؤلفه *Greeting* در *JSX* نیز استفاده می‌شود. قدرت انجام این کار باعث جالب‌شدن *React* است. مورد جدید دیگر که در اینجا رخ می‌دهد ، مؤلفه *View* است. *View* به عنوان ظرفی برای اجزای دیگر مفید است تا به شما در کنترل سبک و طرح کمک کند. با استفاده از *props* و مؤلفه‌های اصلی *Text*، *Image* و *View*، می‌توانید طیف گسترده‌ای از صفحات ایستا را ایجاد کنید. برای یادگیری اینکه چگونه برنامه خود را با گذشت زمان تغییر دهید، باید در مورد *State* بیاموزید.

3-3- *State*

برخلاف *props* که فقط خواندنی هستند و نباید اصلاح شوند، *state* به مؤلفه‌های *React* اجازه می‌دهد تا با گذشت زمان در واکنش به اقدامات کاربر، پاسخ‌های شبکه و هر چیز دیگری خروجی خود را تغییر دهند. در یک مؤلفه *React*، *props* متغیرهایی هستند که ما از مؤلفه والدین به مؤلفه فرزند منتقل می‌کنیم. به طور مشابه، *state* نیز متغیرهایی هستند با این تفاوت که آن‌ها به عنوان پارامتر منتقل نمی‌شوند، بلکه مؤلفه آن‌ها را به صورت داخلی شروع و مدیریت می‌کند. در مثال زیر استفاده از کلاس‌ها را نشان می‌دهیم.



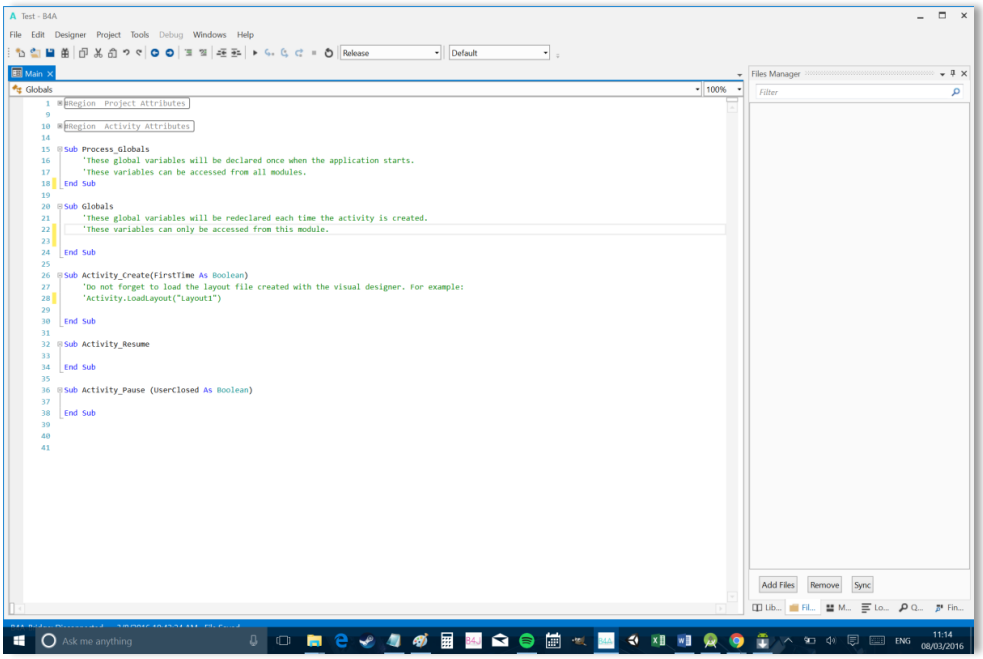
ساخت اولین برنامه با B4A

1- مقدمه‌ای بر *B4A*

کلید انجام کارهای زیاد این است که مطمئن شوید ابزار مناسبی برای کار دارید و وقتی صحبت از توسعه موبایل می‌شود، این به معنای انتخاب *IDE* مناسب برای برنامه خاصی است که می‌خواهید ایجاد کنید. اگر می‌خواهید چیزی بسازید که با زیبایی‌های اندروید سازگار باشد یا اگر می‌خواهید برای اطمینان از حداکثر پشتیبانی از رایج‌ترین مسیر استفاده کنید، *Android Studio* عالی است. *Unity* برای ساخت بازی‌های سه بعدی بدون نیاز به استخدام یک تیم کامل از توسعه‌دهندگان بسیار مناسب است. و همچنین *Basic4Android* یکی از بهترین گزینه‌ها برای مواردی است که می‌خواهید به سرعت یک برنامه اندروید بسازید. همچنین فکر می‌کنیم این یک ابزار نسبتاً خوب برای مبتدیان و ایجاد برنامه‌های بین پلتفرمی (*cross platform*) است.

قبل از ادامه کار، توجه داشته باشید که *Basic4Android* یک نرم افزار رایگان نیست. اگر به *IDE*های پرداخت‌کردنی علاقه ندارید، می‌توانید روش دیگری را امتحان کنید. همچنین یک دوره آزمایشی رایگان در وب‌سایت (*www.b4x.com*) وجود دارد و ثبت سفارش نیز باعث دسترسی شما به انجمن می‌شود (که با افراد بسیار مفیدی پر شده است). *Basic4Android* یک ابزار *IDE* (محیط توسعه یکپارچه) و "*RAD*" (توسعه سریع برنامه) است. همانطور که از این عنوان پیداست، هدف آن تسهیل ایجاد سریع و آسان برنامه‌ها است. در عین حال، سعی می‌کند این کار را بدون محدود کردن شما به هیچ وجه در حد توانایی برنامه‌های شما انجام دهد.

Basic4Android یک برنامه ساز (*app builder*) که کاربران بدون کدنویسی برنامه بسازند نیست. شما باید برای ساخت برنامه کدنویسی کنید اما برای مبتدیان دسترسی به آن بیشتر است و بسیاری از ویژگی‌های جالبی



برای ساده‌سازی روند وجود دارد و به هیچ وجه محدود به آنچه می‌توانید ایجاد کنید نیستید. سناریوهایی وجود دارد که *Basic4Android* بهترین انتخاب برای پروژه شما نخواهد بود. همانطور که در ابتدا گفتیم، انتخاب ابزار مناسب برای کار بخشی از این چالش است. اما به عنوان کسی که می‌تواند از *IDE*های متعددی استفاده کند، بازهم به لطف ساده و سرراست بودن این زبان، برای انجام چندین کار مختلف به استفاده از *B4A* روی می‌آوریم. در ادامه شرحی بر زبان *BASIC* که در *Basic4Android* مورد استفاده قرار می‌گیرد داریم.

اگر در برنامه‌نویسی تازه‌کار هستید، ممکن است دریابید که *BASIC* کمی راحت تر از جاوا است. نحو *BASIC* اغلب بیشتر شبیه به انگلیسی ساده و روان است و این یکی از ویژگی‌هایی است که باعث می‌شود کاربر پسند باشد. برای نشان دادن این مورد، این دو خط کد را که یک کار را انجام می‌دهند مقایسه کنید:

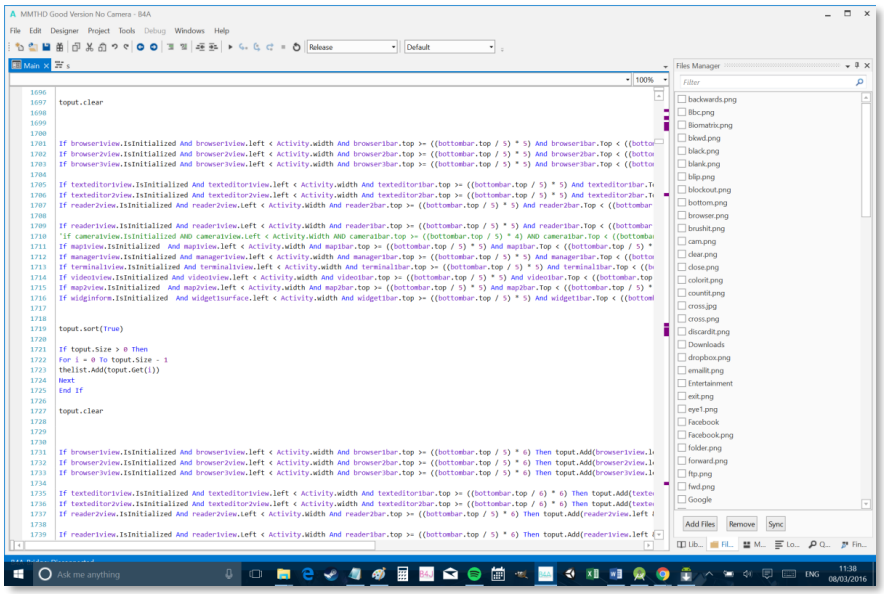
```
BASIC

IF level = 3 THEN titlebar.Text = "level 3"
```

در قبال:

مورد اول بسیار مشهودتر و ساده‌تر دومی است و اگر در برنامه‌نویسی تازه‌وارد باشید راحتتر آن را درک می‌کنید. (و هیچ ; وجود ندارد که آن را فراموش کنید!) جاوا همچنین شی‌گراتر از *Basic4Android* است. این مورد

```
JAVA
if (level == 3){
   titlebar.setText("level 3");
}
```

ممکن است توسط برخی به عنوان یک مزیت شناخته شود و برای پروژه‌های بزرگتر مفید باشد. هرچند برای یک مبتدی، کمی کار بیشتر لازم است تا با آن سر و کله بزند.



Basic4Android کارهای زیادی را در پشت صحنه برای شما انجام می‌دهد. به عنوان مثال، برای آنکه کد دوم در
Java/Android Studio کار کند، باید کلاس مربوطه را *import* کنید. به همین ترتیب، ایجاد متغیرهای سراسری بسیار آسان‌تر است، نوشتن رشته ها نیز آسان است، همچنین تنظیم تایمرها و ... . در هر دو مورد لازم است ویجتی را که به آن رجوع می‌کردید را مقداردهی اولیه کنید، اما *Basic4Android* انجام این کار را برای شما ساده تر کرده است. حتی راه‌اندازی *Basic4Android* مراحل بسیار کمتر و بسیار آسان‌تر از راه‌اندازی *Android Studio* دارد. در مورد اشکال‌زدایی و تست نیز همین مورد وجود دارد. می‌توانید از *B4A Bridge* (یک برنامه تلفن همراه رایگان) برای آزمایش برنامه‌های دستگاه خود از طریق بلوتوث استفاده کنید و سپس هنگام اجرا، اشکال‌زدایی آنها را انجام دهید.

1-1- آیا *Basic4Android* می تواند هر آنچه را که نیاز دارید انجام دهد؟

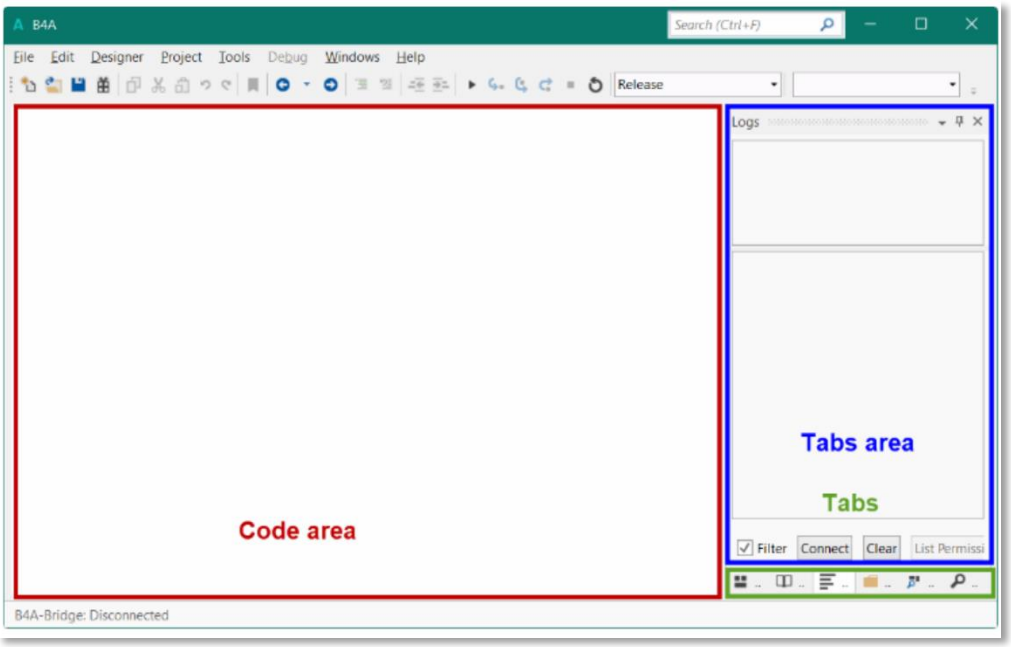
سوالی که احتمالاً بسیاری از افراد می‌پرسند این است: "آیا واقعاً می‌توانید در *Basic4Android* کاری انجام دهید که در *Android Studio* انجام می‌دهید؟". در کل پاسخ این سوال مثبت است. *Basic4Android* به هر کاربر اجازه می‌دهد تا کتابخانه‌های خود را که به زبان جاوا نوشته شده‌اند ایجاد و به اشتراک بگذارد. هر کدی که می‌توانید با *Android Studio* استفاده کنید، قابلیت قرارگیری در کتابخانه *B4A* را دارند. از آنجا که نحو برای *Basic4Android*، *Basic4Java* و *Basic4iOS* تا حد زیادی یکسان است، می‌توانید یک برنامه ساده در رایانه ایجاد کنید و سپس برخی از کتابخانه‌ها و ویژگی‌های خاص پلتفرم را برای انتقال آن به *Android* و *iPhone* تغییر دهید. همچنین می‌توانید از *B4J* (که رایگان است) برای ایجاد برنامه‌های کوچک دسکتاپ برای اهداف خود یا ایجاد برنامه‌هایی که نیاز به برقراری ارتباط با رایانه دارند (مانند کنترلرهای رسانه) استفاده کنید.

2- نصب *B4A* و *Android SDK*

B4A یک ابزار 100% توسعه آزاد برای برنامه‌های آندروید است و همچنین شامل تمام ویژگی‌های مورد نیاز برای توسعه سریع هر نوع برنامه آندروید است. از دیگر ویژگی‌های مهم *B4A* این است که دیگر درگیر کار با *xml* نخواهید بود. شما می‌توانید با استفاده از موارد زیر برنامه‌ها را اشکال‌زدایی و توسعه دهید:

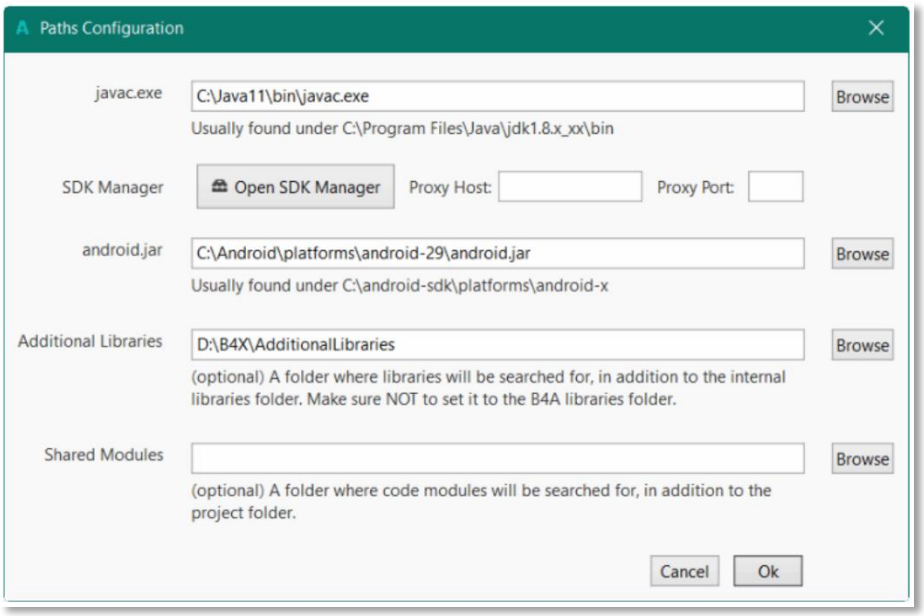
- یک دستگاه واقعی متصل از طریق *B4A Bridge*
- یک دستگاه واقعی متصل از طریق کابل *USB*
- یا یک شبیه‌ساز *Android*.

برای شروع کار ابتدا باید نرم‌افزار *B4A* را از اینترنت دانلود کنید و نصب کنید. زمانی که نرم‌افزار را اجرا می‌کنید با تصویر زیر روبرو خواهید شد.



2-1- پیکربندی مسیرهای *B4A* در *IDE*

نرم‌افزار را باز کنید. روی گزینه *Configure Paths* در منوی *Tools* کلیک کنید. پس از آن تصویر زیر را مشاهده خواهید کرد.



اگر از قبل جاوا روی سیستم شما نصب باشد (فرض می‌کنیم که جاوا نصب است)، مقدار فیلد *javac.exe* پر شده می‌باشد. در تمام این مراحل فرض شده است که *Android SDK* به طور کامل روی سیستم شما نصب شده است. مسیر را برای "*android.jar*" با عبارت مقابل تنظیم کنید:

C:\Android\platforms\android-28\android.jar

توصیه می‌شود یک پوشه خاص برای کتابخانه‌های اضافی ایجاد کنید. *B4A* از دو نوع کتابخانه استفاده می‌کند: کتابخانه‌های استاندارد که همراه *B4A* هستند و در پوشه *Libraries* نرم‌افزار *B4A* قرار دارند. هنگام نصب نسخه جدید *B4A*، این کتابخانه‌ها به طور خودکار به‌روز می‌شوند.

کتابخانه های اضافی، که بخشی از *B4A* نیستند و بیشتر توسط برنامه‌نویسان نوشته می‌شوند. این کتابخانه‌ها باید در یک پوشه خاص متفاوت از پوشه کتابخانه‌های استاندارد ذخیره شوند.

2-2- اتصال یک دستگاه واقعی

روش‌های مختلفی برای اتصال یک دستگاه واقعی وجود دارد:

USB، به پشتیبانی دستگاه از *ADB debugging* نیاز دارد و باید *USB Debugging* را روی دستگاه فعال کنید.

B4A Bridge ، از طریق *WiFi* .

2-2-1- اتصال از طریق *USB*

درایور *USB Google* قبلاً به طور پیش‌فرض روی گوشی شما بارگیری شده است. اگر این درایور کار نمی‌کند، باید به دنبال دستگاه دیگری باشید. برای اتصال دستگاه از طریق *USB* باید *USB debugging* را روی دستگاه فعال کنید. در صورت استفاده از شبیه‌ساز، این مورد نیز مورد نیاز است. روی تنظیمات گوشی کلیک کنید. وارد قسمت
Developper options شوید و *USB debugging* را فعال کنید.

2-2-2- اتصال از طریق *B4A-Bridge*

همیشه توصیه می‌شود به جای شبیه‌ساز اندروید از یک دستگاه واقعی استفاده کنید چون شبیه‌ساز در مقایسه با دستگاه واقعی بسیار کند است (به خصوص با نصب برنامه ها). با این وجود همه دستگاه‌ها از *ADB debugging* پشتیبانی نمی‌کنند. این دلیل استفاده از ابزار *B4A-Bridge* است. *B4A-Bridge* از دو جز ساخته شده است. یک مؤلفه بر روی دستگاه اجرا می‌شود و به مؤلفه دوم که بخشی از *IDE* است امکان اتصال و ارتباط با دستگاه را می‌دهد. اتصال از طریق شبکه انجام می‌شود (*B4A-Bridge* در صورت عدم وجود شبکه نمی‌تواند کار کند). پس از اتصال، *B4A-Bridge* از تمام ویژگی‌های *IDE* پشتیبانی می‌کند که شامل: نصب برنامه‌ها ، مشاهده *LogCat* و طراح بصری و ... می‌باشد.

3- شروع برنامه‌نویسی *B4A* با ساخت برنامه *Hello World!*

بیایید با یک "*Hello World*" ساده در *Basic4Android* شروع کنیم. روی گزینه *new* کلیک کنید و سپس پروژه خود را در جایی ذخیره کنید که بتوانید دوباره آن را پیدا کنید. برای شروع، ما قصد داریم یک برچسب ایجاد کنیم. ما می‌توانیم این کار را به صورت بصری انجام دهیم اما در این قسمت به صورت کدنویسی این کار را انجام می‌دهیم. این کار سریع و آسان است.

ابتدا یک برچسب را در *Globals* تعریف می‌کنیم. خط زیر را اضافه کنید:

```
Sub Globals
Dim label1 As Label
End Sub
```

در اینجا *sub* مانند یک متد رفتار می‌کند. اکنون می‌توانید از هرجای کد به *label1* ارجاع دهید.

در ادامه باید یک متد به نام *Activity_Create* بنویسیم. *Activity_Create* متدی است که هنگام شروع *activity* برای اولین بار اجرا می‌شود. در اینجا قصد داریم مقدار اولیه *label1* را مقداردهی کنیم و بگوییم که می‌خواهیم به چه صورت باشد باشد. از کد زیر استفاده کنید:

```
Sub Activity_Create(FirstTime as Boolean)
label1.Initialize("Label1")
Activity.AddView(label1, 0%x, 0%y, 100%x, 100%y)
```

ما از این طریق برچسب را ایجاد کردیم و سپس آن را به *activity* (بخشی از برنامه در حال اجرا) اضافه کردیم در حالی که مکان و اندازه آن را نیز تعیین کردیم. مکان *'0%x, 0%y'* به این معنی است که در گوشه بالا سمت چپ صفحه است. عرض و ارتفاع *'0%x, 0%y'* به این معنی است که 100٪ عرض و 100٪ ارتفاع را اشغال می‌کند. اکنون برچسب ما دقیقاً همان اندازه صفحه است که البته نامرئی است. از آنجا که ما از درصد استفاده می‌کنیم، اندازه برچسب متناسب با اندازه نمایشگری که در حال اجرا است تغییر می‌کند. این خطوط پیام سلام ما را اضافه می‌کنند و اطمینان حاصل می‌کنند که در مرکز برچسب شناور (*float*) است:

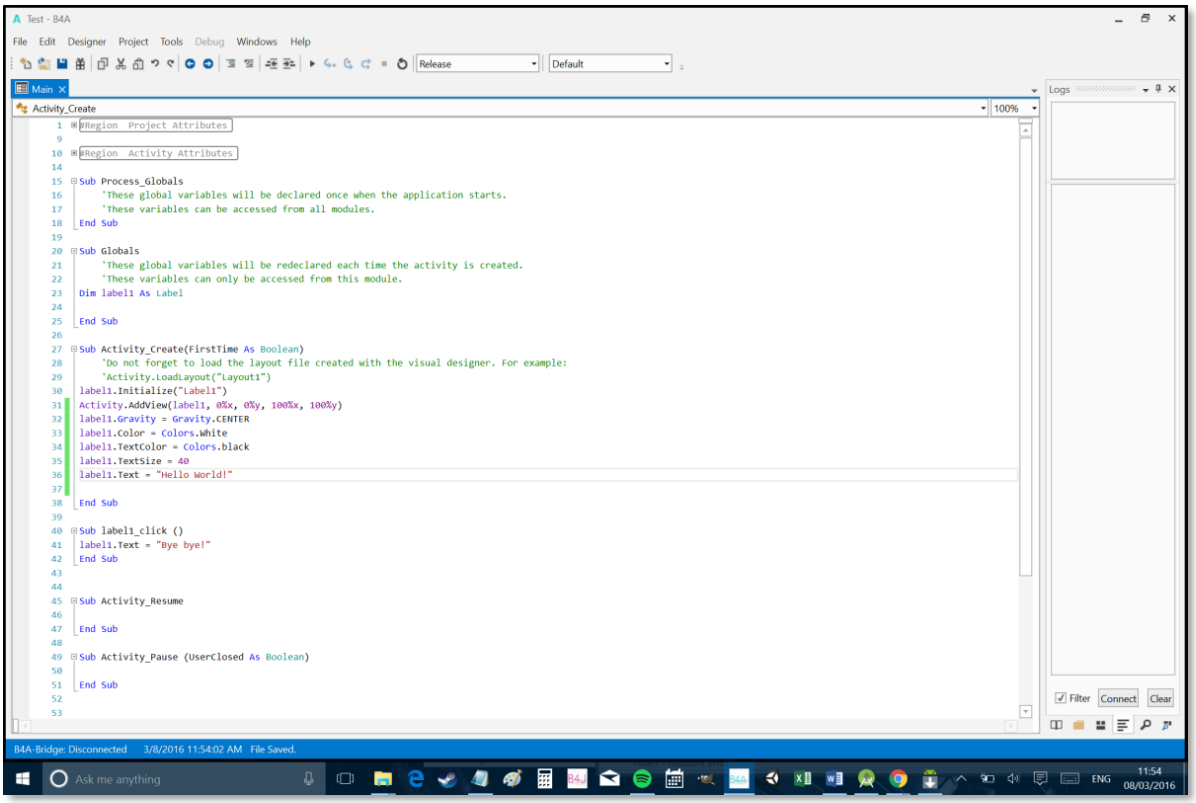
همچنین می‌توانیم خطوط زیر را اضافه کنیم:

این کار کمی سبک (*style*) به برچسب می‌دهد.در این قسمت "*Hello World!*" به پایان رسیده است اما برای ایجاد تعاملی بیشتر در کارها، می‌توانیم متد دیگری ایجاد کنیم:

با این کار به ازای کلیک روی برچسب عبارت *Hello World!* به *Bye bye!* تغییر میابد.

```
label1.Color = Colors.White

label1.TextColor = Colors.black

label1.TextSize = 40

Sub label1_click ()

label1.Text = "Bye bye!"

End Sub
```

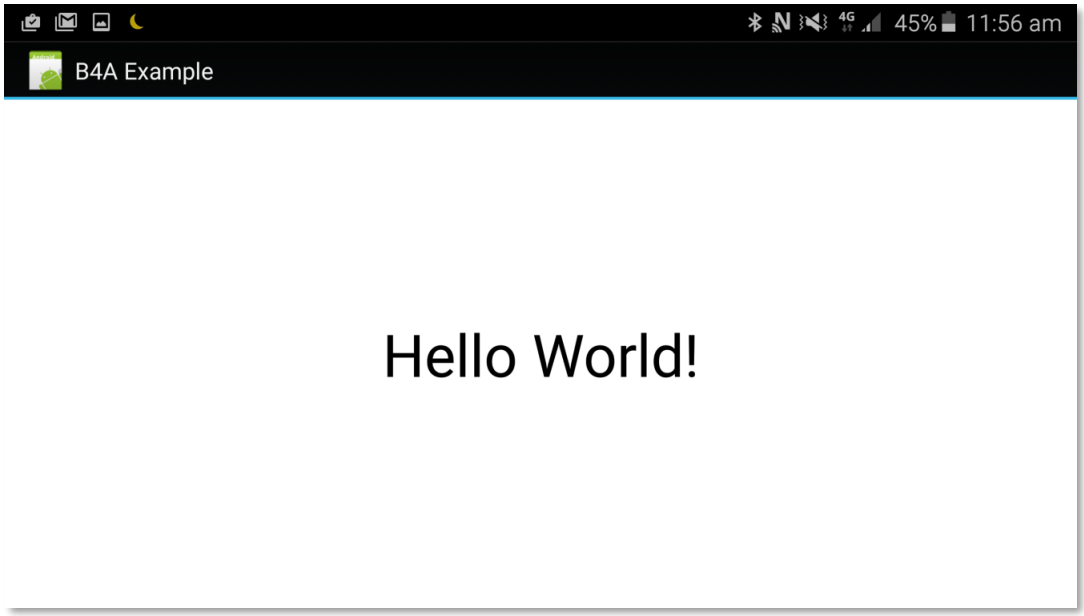
اگر کار را درست انجام داده باشید، چیزی شبیه به عکس زیر خواهد بود:



با اجرای آن روی دستگاه واقعی یا شبیه‌ساز با تصویر زیر روبرو می‌شوید:

4- شرح بیشتر برنامه‌نویسی *B4A* با ساخت برنامه ریاضی ساده

در زیر عکس برنامه‌ای که در ادامه اقدام به ساخت آن خواهیم کرد آمده است:



در عکس بالا، ما موارد زیر را خواهیم داشت:

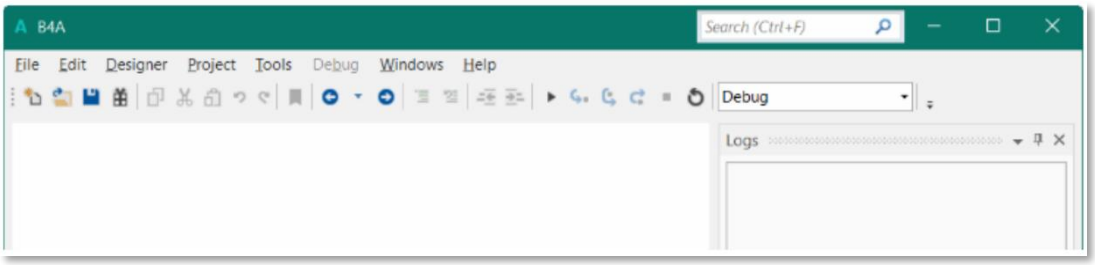
- 2 برچسب که اعداد ایجاد شده به طور تصادفی را نمایش می‌دهند (بین 1 تا 9).
- 1 برچسب با علامت ریاضی (+).
- 1 نمای *EditText* که در آن کاربر باید نتیجه را وارد کند.

- 1 دکمه، برای تأیید زمان پایان کار کاربر هنگام وارد کردن نتیجه استفاده می‌شود یا محاسبه جدیدی را ایجاد می کند.
- 1 برچسب با نظر درباره نتیجه.

ما طرح رابط کاربری را با *VisualDesigner* طراحی می‌کنیم و مرحله به مرحله کل مراحل را طی می‌کنیم.

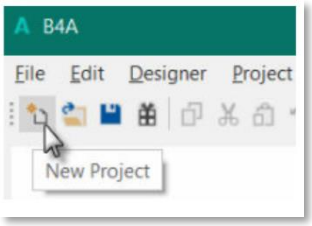
1-4- *IDE* را اجرا کنید

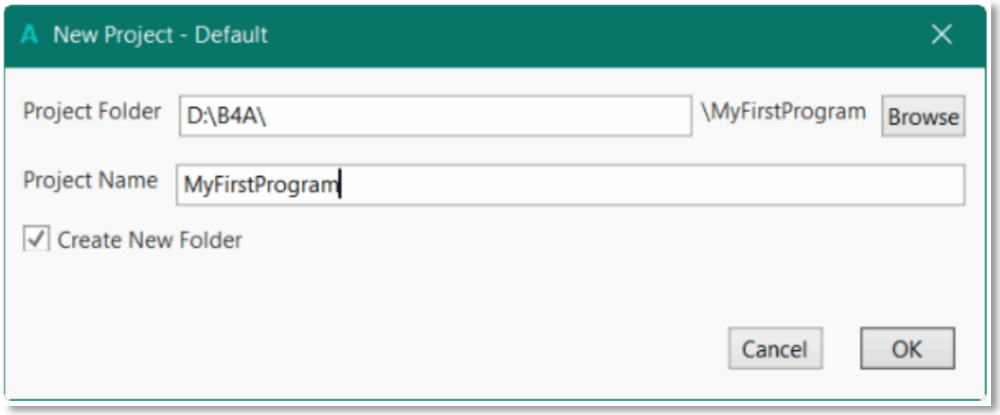
وقتی *IDE* را باز می‌کنید همه چیز خالی است.



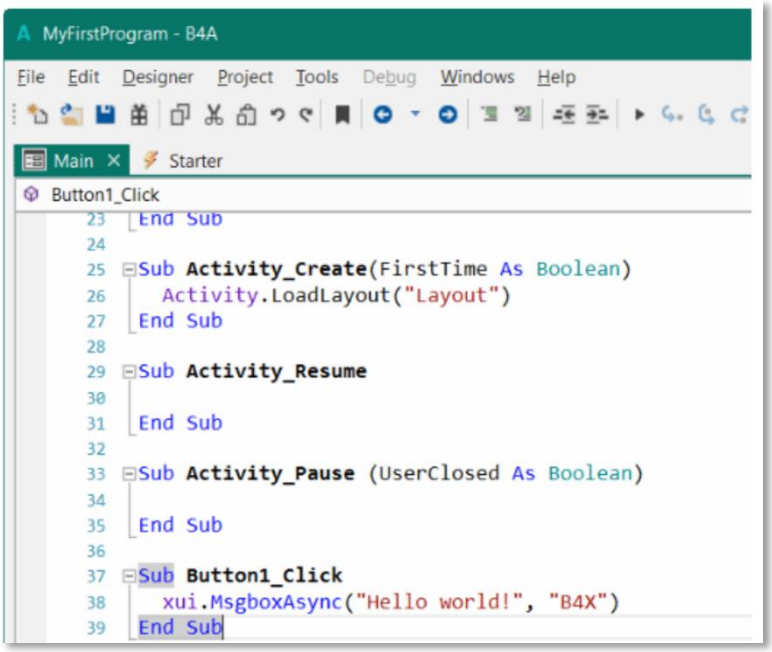
روی دکمه *New Button* کلیک کنید.

در ادامه از شما درخواست می‌شود که پروژه خود را ذخیره کنید:



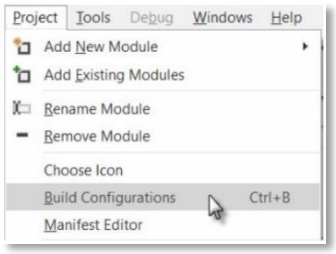


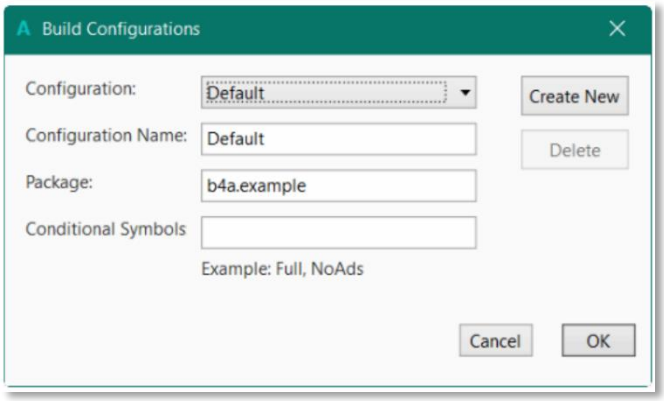
در قسمت *Project Name*، *MyFirstProgram* را وارد کنید. همچنین مسیر *Project Folder* را تعیین کنید. برای این کار می‌توانید هر پوشه‌ای را وارد کنید. ما از *D:\B4A* به عنوان پوشه عمومی برای پروژه‌های *B4A* استفاده می‌کنیم. روی *ok* کلیک کنید و راهی مرحله بعد شوید. در ادامه با یک فایل *main* مواجه خواهید شد که قالب پیش‌فرض یک فایل *B4A* هست.

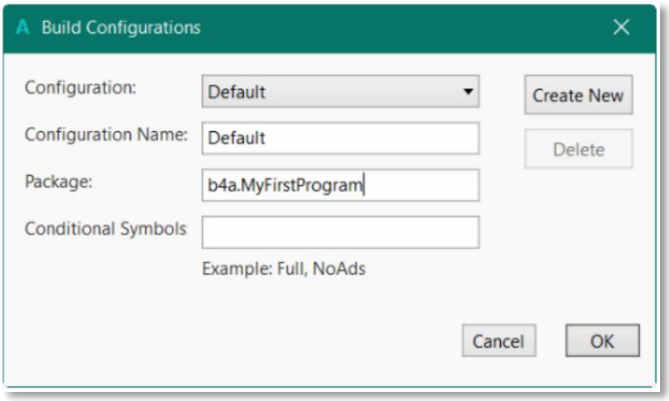


4-2- تعیین نام بسته (*package*)

هر برنامه به یک نام بسته نیاز دارد. از منو روی گزینه *Project* کلیک کنید و پس از آن روی *Build Configuration* کلیک کنید. در پنجره باز شده، نام پیش‌فرض بسته *b4a.example* می‌باشد که ما آن را تغییر داده و *b4a.MyFirstProgram* را جایگزین آن می‌کنیم و روی دکمه *ok* می‌زنیم.

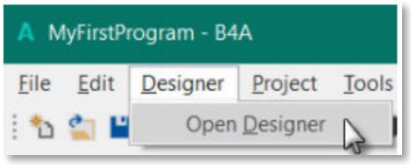


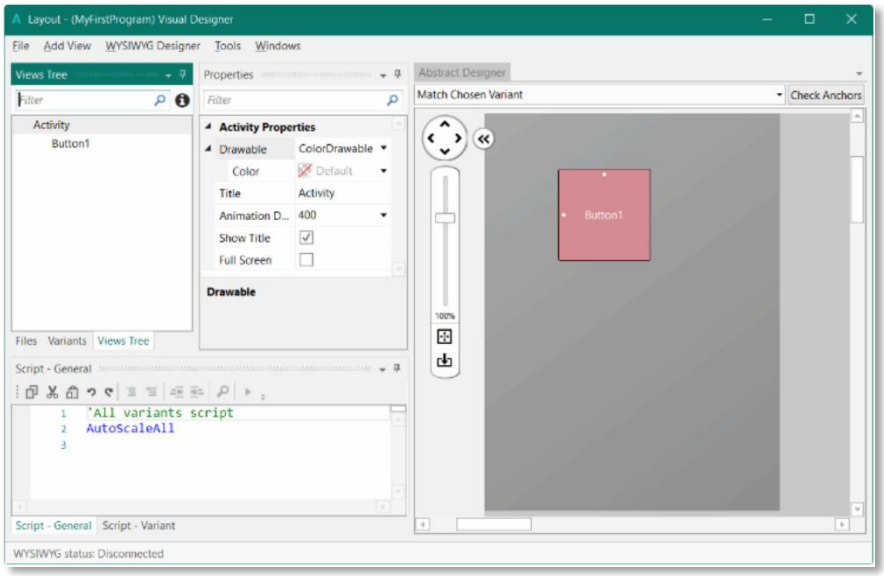




4-3- در *IDE*، *designer* را اجرا کنید و کد را بنویسید

از منو *designer* ا اجرا کنید. پنجره باز شده باید به شکل زیر باشد:

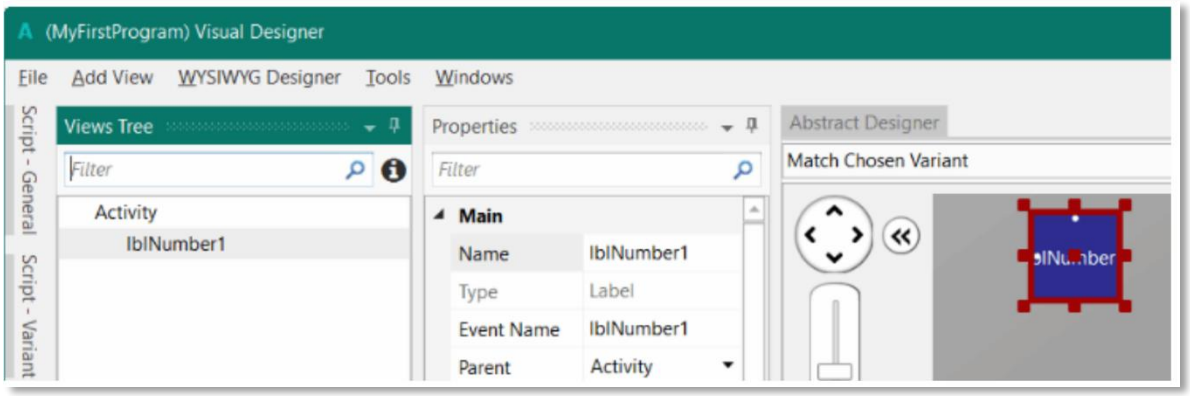




پنجره های مختلفی وجود دارد:

- *Views Tree*: تمام نماها را به صورت درخت نشان می‌دهد.
- *Properties*: تمام خصوصیات نمای انتخاب شده را نشان می‌دهد.
- *Abstract Designer* نماها را روی صفحه نمایش می‌دهد.
- *Script - General*: اجازه می‌دهد تا تنظیمات دقیق را روی لایه‌ها انجام دهید.

در ادامه نمای *Button1* که به صورت پیش‌فرض در لایه موجود بود را حذف می‌کنیم. از قسمت *Add View* 2 تا برچسب یا *label* اضافه می‌کنیم. این دو برچسب به ازای 2 عدد توضیح داده شده در بالا می‌باشد. می‌بینیم که برچسب در صفحه ایجاد شده که قابلیت تغییر اندازه را نیز داراست. حال بیایید کمی تغییرات در آن ایجاد کنیم. این تغییرات را می‌توان از قسمت *properties* انجام داد. به طور پیش‌فرض، نام *Label* با یک عدد است که در اینجا *Label1* است. بگذارید نام آن را به *lblNumber1* تغییر دهیم. سه حرف "*lbl*" در ابتدا به معنای "*Label*" است و "*Number1*" به معنی شماره اول است. توصیه می‌شود از نام‌های معنی‌دار برای نماها استفاده کنید تا مستقیماً بدانیم هدف آن چیست.



روی برچسب ایجاد شده کلیک راست کنید و گزینه *duplicate* را بزنید تا یک برچسب دیگر تکرار شود و نام آن را *lblNumber2* بگذارید. در ادامه باز هم مثل قبل عمال کرده و یک برچسب دیگر ایجاد کرده ویژگی *text* آن را + و نام آن را *lblMathSign* قرار می‌دهیم. در مرحله بعدی از قسمت *Add View* یک *EditText* انتخاب

می‌کنیم و مکان آن را زیر برچسب‌ها قرار می‌دهیم و نام آن را *edtResult* می‌گذاریم. در قسمت بعدی یک دکمه یا *Button* انتخاب می‌کنیم و نام آن را *btnAction* می‌گذاریم و همچنین ویژگی *Text* آن را روی *ok* تنظیم می‌کنیم.

یک برچسب دیگر اضافه می‌کنیم و مکان آن را در زیر دیگر نماها قرار می‌دهیم و نام آن را *lblComments*

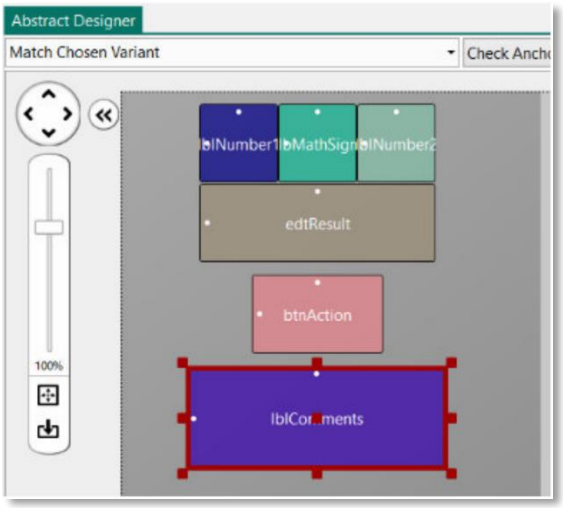
> **توجه: ویژگی‌ها یا Properties دیگری مانند رنگ متن، فونت و ... هم برای تغییر وجود دارند که به دلیل سادگی آن توضیح داده نشده است.**

می‌گذاریم.

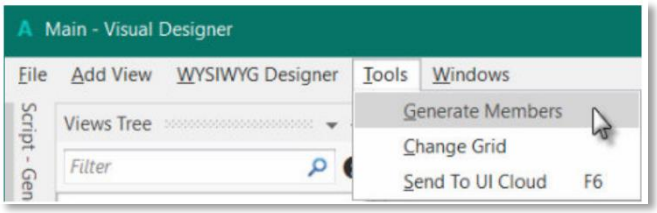
صفحه نهایی در *designer* به شکل زیر خواهد بود:

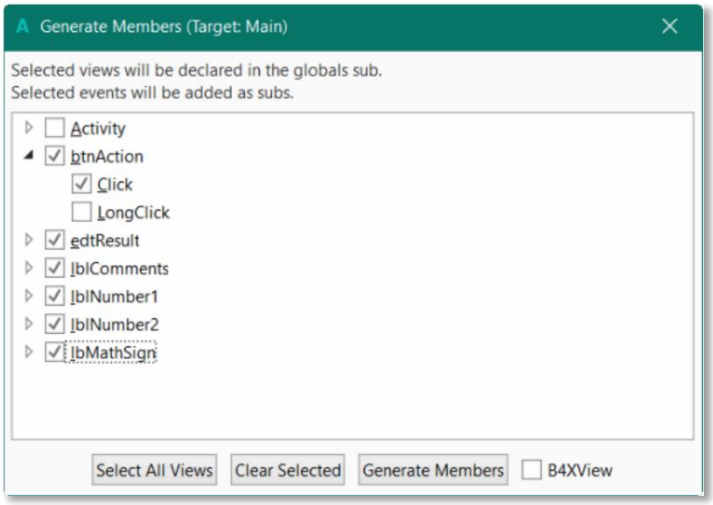
و اگر آن را اجرا کنیم روی شبیه‌ساز یا دستگاه واقعی به شکل زیر خواهد بود:

پروژه را ذخیره کرده و ادامه می‌دهیم. برای نوشتن روال‌های پروژه، باید *View* ها را در کد ارجاع دهیم. این کار را می‌توان با *Generate Memberstool* در *Designer* انجام داد. از منوی *tools* روی *Generate Members* کلیک کنید.



در اینجا تمام نماهای اضافه شده به لایه فعلی را می‌یابیم و چک باکس همه نماها را می‌زنیم. زدن چک باکس نمای *edtResult* باعث ایجاد مرجع آن در زیر روال (*sub*) *Globals* در کد می‌شود. این امر برای شناخته شدن نما توسط سیستم و اجازه دادن به عملکرد تکمیل خودکار، لازم است. روی دکمه *Generate Members* کلیک کرده تا چیزی شبیه به این در کد ظاهر شود.






```
Private btnAction As Button
Private edtResult As EditText
Private lblComments As Label
Private lblMathSign As Label
Private lblNumber1 As Label
Private lblNumber2 As Label

Sub btnAction_Click
End Sub
```

حالا برای وارد کردن کد به *IDE* برمی‌گردیم. ابتدا، برای بارگذاری لایه، به *Activity* خود نیاز داریم. این مورد از قبل در زیرروال "*Activity_Create*" آماده شده است

```
Activity.loadlayout("Layout")
```

ما می‌خواهیم به محض شروع برنامه یک *problem* جدید ایجاد کنیم. بنابراین، ما یک فراخوانی به زیرروال *NewProblem* اضافه می‌کنیم.

```
Sub Activity_Create(FirstTime As Boolean)
  Activity.LoadLayout("Main")
  NewProblem
End Sub
```

توجه داشته باشید که *NewProblem* به رنگ قرمز است به معنی خطا است. این خطا *'Undeclared variable ...'* می‌باشد. در این حالت به این معنی است که سیستم این نام را تشخیص نمی‌دهد. ایجاد یک مسئله جدید به معنای تولید دو مقدار تصادفی جدید بین 1 تا 9 برای *Number1* و *Number2* است، سپس مقادیر را با استفاده از خصیصه *lblNumber1 "Text"* و *lblNumber2* نشان می‌دهیم. برای این کار کد زیر را وارد می‌کنیم: در *Sub Globals* دو متغیر برای دو عدد اضافه می‌کنیم.

```
Public Number1, Number2 As Int
End Sub
```

و زیرروال"*NewProblem*":

```
Sub NewProblem
  Number1 = Rnd(1, 10)       ' Generates a random number between 1 and 9
  Number2 = Rnd(1, 10)       ' Generates a random number between 1 and 9
  lblNumber1.Text = Number1      ' Displays Number1 in label lblNumber1
  lblNumber2.Text = Number2      ' Displays Number2 in label lblNumber2
  lblComments.Text = "Enter the result" & CRLF & "and click on OK"
  edtResult.Text = ""              ' Sets edtResult.Text to empty
End Sub
```

این تابع *Rnd(1, 10)* یک عدد تصادفی از 1 تا 10 ایجاد می‌کند، بنابراین بین 1 و 9 است. خط زیر نظر را در *lblCommentsview* نمایش می‌دهد:

```
lblComments.Text = "Enter the result" & CRLF & "and click on OK"
```

CRLF کاراکتر LineFeed یا خط بعد است.

اکنون کد رویداد کلیک دکمه را اضافه می‌کنیم. ما دو حالت داریم:

- وقتی متن دکمه برابر با "*O K*" باشد (با فاصله‌ای بین *O* و *K*) ، به این معنی است که یک مشکل جدید نمایش داده می‌شود و برنامه منتظر است کاربر نتیجه‌ای وارد کند و دکمه را فشار دهد
- وقتی متن دکمه برابر با "*NEW*" باشد، به این معنی است که کاربر پاسخ صحیحی را وارد کرده است و هنگامی که کاربر روی دکمه کلیک می‌کند مشکل جدیدی ایجاد می‌شود. در حال حاضر روال رویداد کلیک دکمه در کد وجود دارد.

کد زیر را اضافه کنید:

```
Sub btnAction_Click
  If btnAction.Text = "O K" Then
    If edtResult.Text = "" Then
      Msgbox("No result entered","E R R O R")
    Else
      CheckResult
    End If
  Else
    NewProblem
```

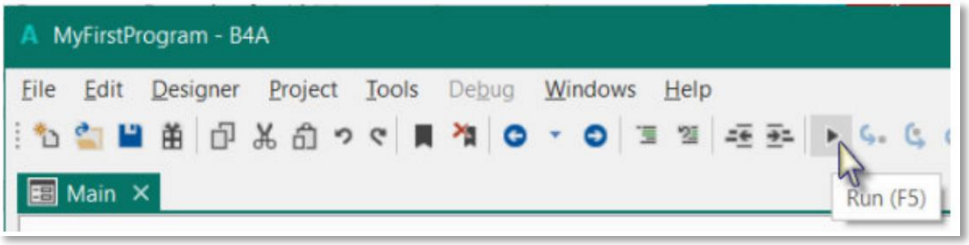
```
    btnAction.Text = "O K"
  End If
End Sub
```

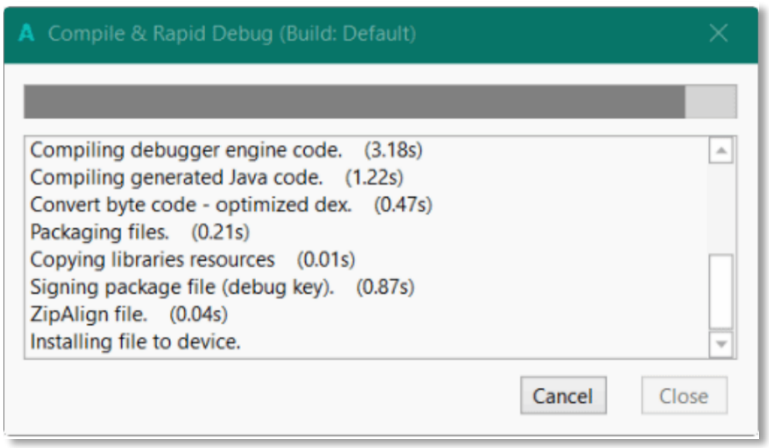
آخرین روال نتیجه را بررسی می‌کند.

```
Sub CheckResult
  If edtResult.Text = Number1 + Number2 Then
    lblComments.Text = "G O O D result" & CRLF & "Click on NEW"
    btnAction.Text = "N E W"
  Else
    lblComments.Text = "W R O N G result" &CRLF & "Enter a new result" & CRLF & "and
    clickOK"
  End If
End Sub
```

در شرط اول بررسی می‌کنیم که آیا نتیجه وارد شده درست است یا خیر.

پروژه را ذخیره می‌کنیم و روی دکمه *Run* کلیک می‌کنیم.





با مشاهده دستگاه، باید چیزی شبیه به تصویر زیر، با شماره‌های مختلف مشاهده کنید.

حال می‌توانید نتیجه جمع خواسته شده را انجام داده و در کادر متن بنویسید و با زدن دکمه نتیجه کار خود را ببینید.

تبریک می‌گویم همین‌که به اینجای این مطلب را خواندید، یعنی احتمال موفقیت شما خیلی بالاست. چون عموم مردم حتی کتابی که می‌خرند را باز نمی‌کنند، چه برسد به اینکه که آن را تا آخر بخوانند. برای همین هم من این کتاب را تا جای ممکن کوتاه نوشتم که بیشتر افراد آن را تا انتها بخوانند و متوجه شوند که برنامه‌نویسی تا چه حد دلچسب است.

در کل دو حالت وجود دارد:

یا تونستید پابه پای ما با این کتابچه به دنیا سلام کنید.

یا نتوانستید

که ما برای هر دو حالت راهکار داریم.

اصلا جای نگرانی نیست.

اگر شما هم توانستید قدم‌به قدم همراه کتاب پیش برید و به دنیا سلام کنید که عالی است. پس وقت آن است که آموزش‌های سطح بالاتر را در سایت من gitiget.com مشاهده کنید.

اگر هم موفق نشده‌اید، هیچ مشکلی نیست. به‌طور کلی بهترین رسانه برای یادگیری برنامه‌نویسی اول ویدیو آموزشی و سپس کتاب است. اگر با مطالعه کتاب مفاهیم را درک نکردید، به‌طور قطع با ویدیوهای سایت gitiget.com که برای شما رایگان است، متوجه می‌شوید.

به این صفحه از سایت من بیاید که نه تنها به دنیا سلام کنیم؛ بلکه آن را بسازیم.

https://gitiget.com/free-android-programming-training