کاتلین به زبان ساده

برای دریافت فایل‌ها و آپدیت‌های جدید این کتاب به سایت **www.w3-farsi.com** مراجعه فرمایید.

راه‌های ارتباط با نویسنده

وب سایت: www.w3-farsi.com

لینک تلگرام: https://telegram.me/ebrahimi_younes

ID تلگرام: @ebrahimi_younes

پست الکترونیکی: younes.ebrahimi.1391@gmail.com

تقدیم به

همسر و پسر عزیزم

کاتلین چیست

کاتلین یک زبان برنامه نویسی جدید و قدرتمند است که توسط شرکت JetBrains ساخته شده است و یک زبان منبع باز محسوب می‌شود که ویژگی‌های برنامه نویسی شی گرا و ماژولار را با یک دیگر ترکیب کرده است .نام کاتلین برگرفته از جزیره ای به همین نام در اطراف سن پترزبوگ گرفته شده است. کاتلین از سال ۲۰۱۱ معرفی شده و در حقیقت زبان جدیدی محسوب نمی‌شود اما پس از آنکه در کنفرانس Google I/O در سال ۲۰۱۷ به عنوان یک زبان رسمی برای توسعه اندروید معرفی شد، باعث شد کاتلین بر سر زبان‌ها بیفتد و افراد زیادی سراغ یادگیری آن بروند. در ابتدا برخی از مزایای کاتلین را بررسی می‌کنیم.

مزایا

- یادگیری کاتلین بسیار ساده است: کاتلین یک زبان برنامه نویسی functionalبوده که یادگیری آن به سادگی امکان پذیر است. ساختار کاتلین بسیار شبیه به زبان جاواست که این موضوع به خصوص برای برنامه نویسان جاوا خبر خوبی است، زیرا می‌توانند به سرعت آن را فراگرفته و در پروژه‌های بعدی خود از آن استفاده کنند. استفاده از کاتلین می‌تواند کد شما را خواناتر و قابل فهم تر کند.

- کاتلین می‌تواند به JavaScript یا JVM bytecode کامپایل شود. دقیقاً مشابه جاوا، کاتلین نیز می‌تواند به فرمت bytecode کامپایل شود. روش کار به این صورت است که ابتدا کد ما به بایت کد که یک زبان قابل فهم برای ماشین مجازی جاوا (JVM) است کامپایل شده و سپس JVM برنامه را در ماشین مقصد اجرا می‌کند. با این کار برنامه می‌تواند مستقل از سیستم عامل اجرا شود.

- کاتلین می‌تواند از تمام کتابخانه‌ها و فرم ورک های جاوا استفاده کند: شما می‌توانید از تمام کتابخانه‌ها و فرم ورک های زبان جاوا در کاتلین استفاده کنید که این یکی از مهم‌ترین ویژگی‌های کاتلین است.

- جاوا به صورت خودکار قابل تبدیل به کاتلین است: ابزارهایی وجود دارد که می‌توانید به صورت خودکار کدهای جاوای خود را به کاتلین تبدیل کنید و به این ترتیب در وقت خود صرفه جویی کنید.

- بازبینی کدها دیگر کار مشکلی نخواهد بود: کاتلین تمرکز زیادی بر روی خوانایی و قابل فهم بودن کدها دارد که این موضوع فرآیند بازبینی کدها را ساده می‌کند، بنابراین همه اعضای تیم می‌توانند کدها را بازبینی کنند حتی اگر با این زبان آشنایی نداشته باشند.

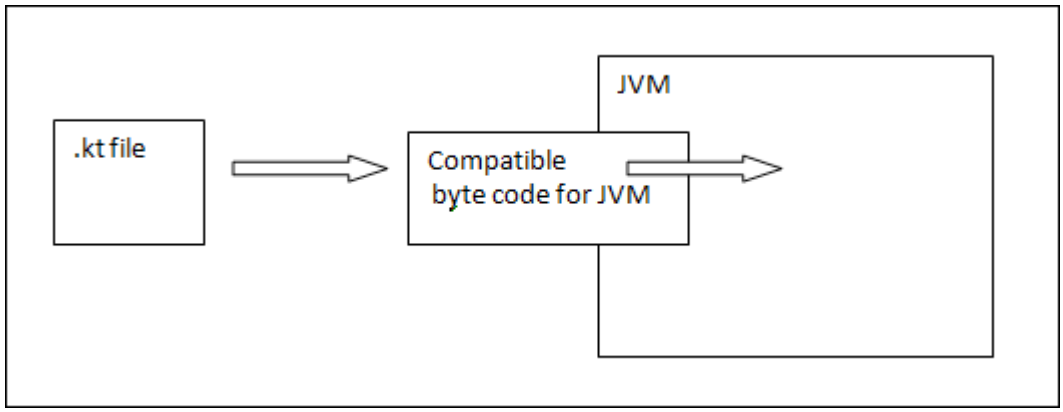
معایب

- عدم پشتیبانی از برخی ویژگی‌های زبان جاوا: کاتلین از برخی ویژگی‌های زبان جاوا نظیر انواع Primitive ،Static Type ها یا انواع داده اولیه، عملگر شرطی سه تایی و ... پشتیبانی نمی‌کند که البته تعداد آن‌ها زیاد نیست.
- سرعت کامپایل: در برخی موارد کاتلین سریع‌تر از جاوا عمل می‌کند اما گاهی اوقات نیز به شکل قابل ملاحظه‌ای کندتر عمل می‌کند.

معماری کاتلین

کاتلین، مانند هر زبان دیگر، معماری خاص خودش را برای اختصاص حافظه و تولید خروجی با کیفیت برای کاربر نهایی دارد. کامپایلر کاتلین، سناریوهای متفاوتی را برای زبان‌های مقصد متفاوت از جمله جاوا یا جاوا اسکریپت دارد.

کامپایلر کاتلین یک بایت کد برای اجرا در JVM تولید می‌کند که دقیقاً همان بایت کدی است که برای یک فایل جاوا با پسوند class. تولید می‌شود. به عبارت دیگر هر دو بایت کدِ تولید شده، بر روی JMV اجرا می‌شوند و این بدین معنی است که یک قابلیت تعاملی بین جاوا و کاتلین وجود دارد.



در صورتی که مقصد کاتلین زبان جاوا اسکریپت باشد، کامپایلر فایل kt. را به ES5.1 تبدیل می‌کند و یک کد سازگار با جاوا اسکریپت تولید می‌شود. به صورت پیش فرض کاتلین مانند جاوا با استفاده از JVM اجرا می‌شود و وابسته به آن است اما با قابلیت Kotlin/Native، کامپایلر کاتلین می‌تواند با استفاده از LLVM، برنامه‌های نوشته شده با کاتلین را بدون نیاز به ماشین مجازی نیز اجرا کند. در حال حاضر با توجه به مستندات JetBrains، ماشین‌های مقصد زیر پشتیبانی می‌شوند:

- Mac OS X 10.10 and later (x86-64)
- x86-64 Ubuntu Linux (14.04, 16.04 and later), other Linux flavours may work as well
- Apple iOS (arm64), cross-compiled on MacOS X host
- Raspberry Pi, cross-compiled on Linux host

JDK و IntelliJ IDEA

IntelliJ IDEA محیط توسعه یکپارچه‌ای است که، دارای ابزارهایی برای کمک به شما برای توسعه برنامه‌های Kotlin می‌باشد. توصیه می‌کنیم که از محیط IntelliJ IDEA برای ساخت برنامه استفاده کنید، چون این محیط دارای ویژگی‌های زیادی برای کمک به شما جهت توسعه برنامه‌های Kotlin می‌باشد. توسط IntelliJ IDEA می‌توان در استانداردهای مختلف جاوا مانند J2SE، J2EE و J2ME برنامه نویسی کرد. همچنین از محیط زبان‌های PHP، HTML، C و نیز Groovy پشتیبانی می‌کند.

JDK که مخفف عبارت Java Development Toolkit می‌باشد ترکیبی از کامپایلر زبان جاوا، کلاس‌های کتابخانه‌ای (Java Class Libraries) و JVM و فایل‌های راهنمای آن‌ها می‌باشد. برای اینکه ما بتوانیم با استفاده از زبان برنامه نویسی Kotlin، برنامه بنویسیم به این مجموعه نیاز داریم. تعداد زیادی از پردازش‌ها که وقت شما را هدر می‌دهند به صورت خودکار توسط IntelliJ IDEA انجام می‌شوند. یکی از این ویژگی‌ها اینتلی سنس (Intellisense) است که شما را در تایپ سریع کدهایتان کمک می‌کند .

IntelliJ IDEA برنامه شما را خطایابی می‌کند و حتی خطاهای کوچک (مانند بزرگ یا کوچک نوشتن حروف) را برطرف می‌کند. با این برنامه‌های قدرتمند بازدهی شما افزایش می‌یابد و در وقت شما با وجود این ویژگیهای شگفت انگیز صرفه جویی می‌شود. IntelliJ IDEA رایگان است و می‌توان آن را دانلود و از آن استفاده کرد. این برنامه ویژگیهای کافی را برای شروع برنامه نویسی Kotlin در اختیار شما قرار می‌دهد. در آموزش‌ها از IntelliJ IDEA نسخه ۲۰۱۸ استفاده شده است و استفاده از این نسخه برای انجام تمرینات این سایت کافی می‌باشد. برای دانلود نرم افزارهای مورد نیاز بر روی لینک زیر کلیک کنید:

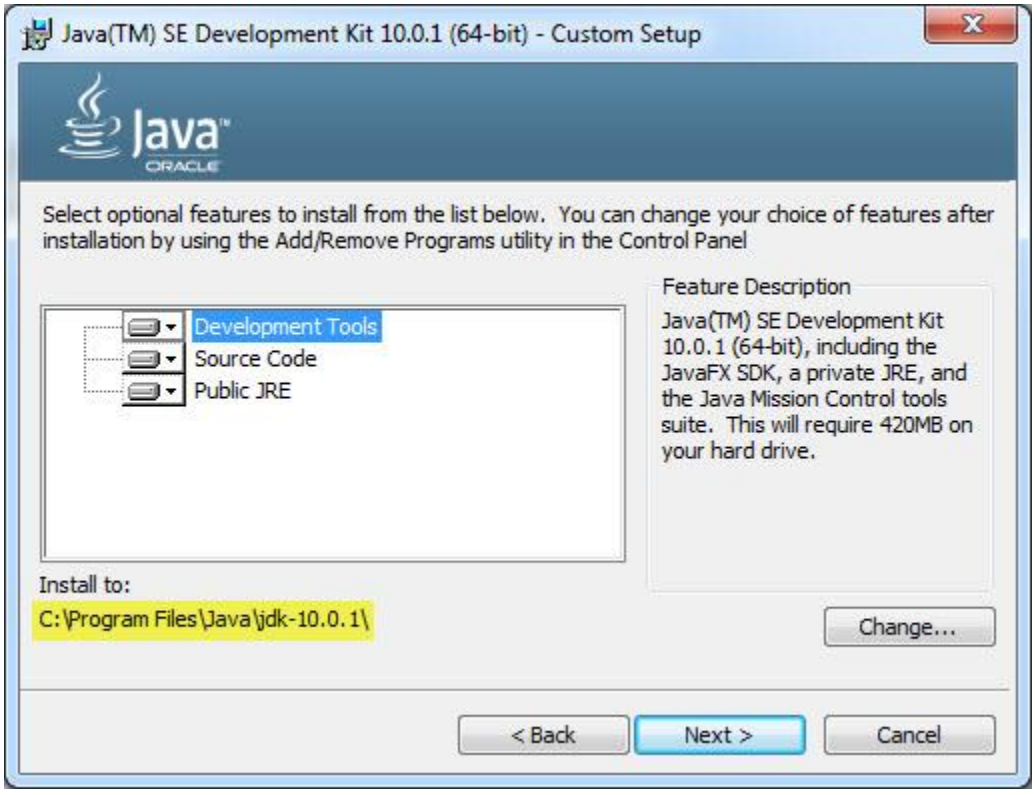
http://dl.w3-farsi.com/Software/Kotlin/Intellij-IDEA-and-JDK.rar

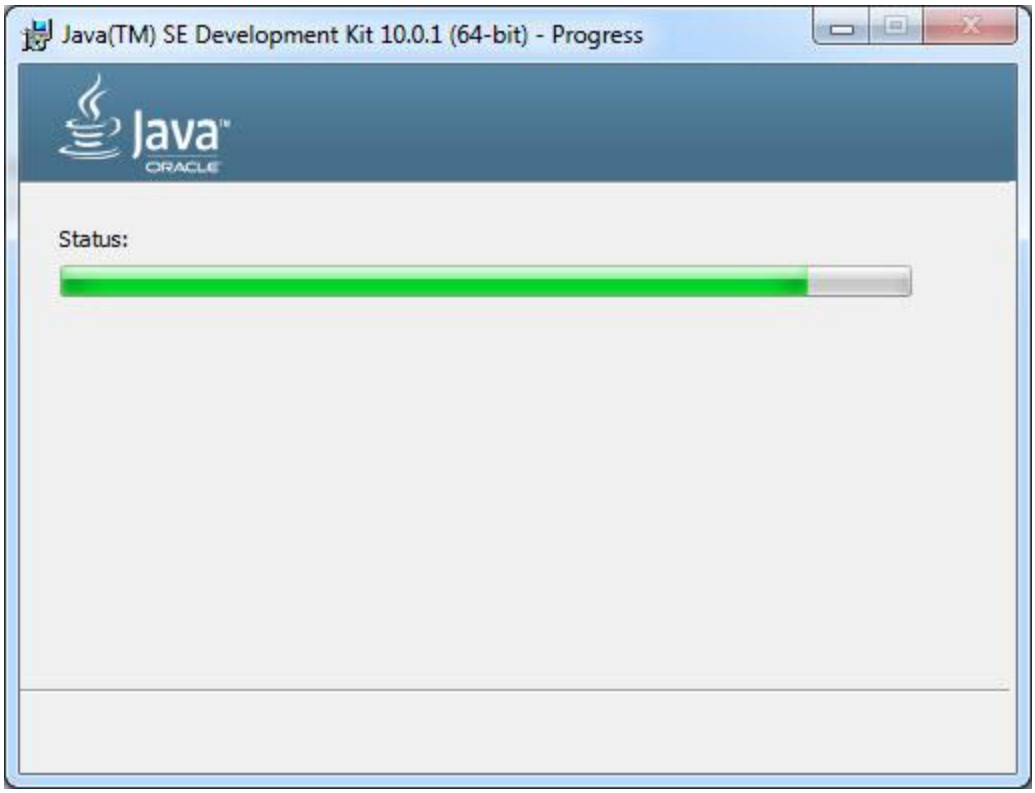
در درس آینده مراحل نصب و راه اندازی دو نرم افزار JDK و IntelliJ IDEA را توضیح می‌دهیم.

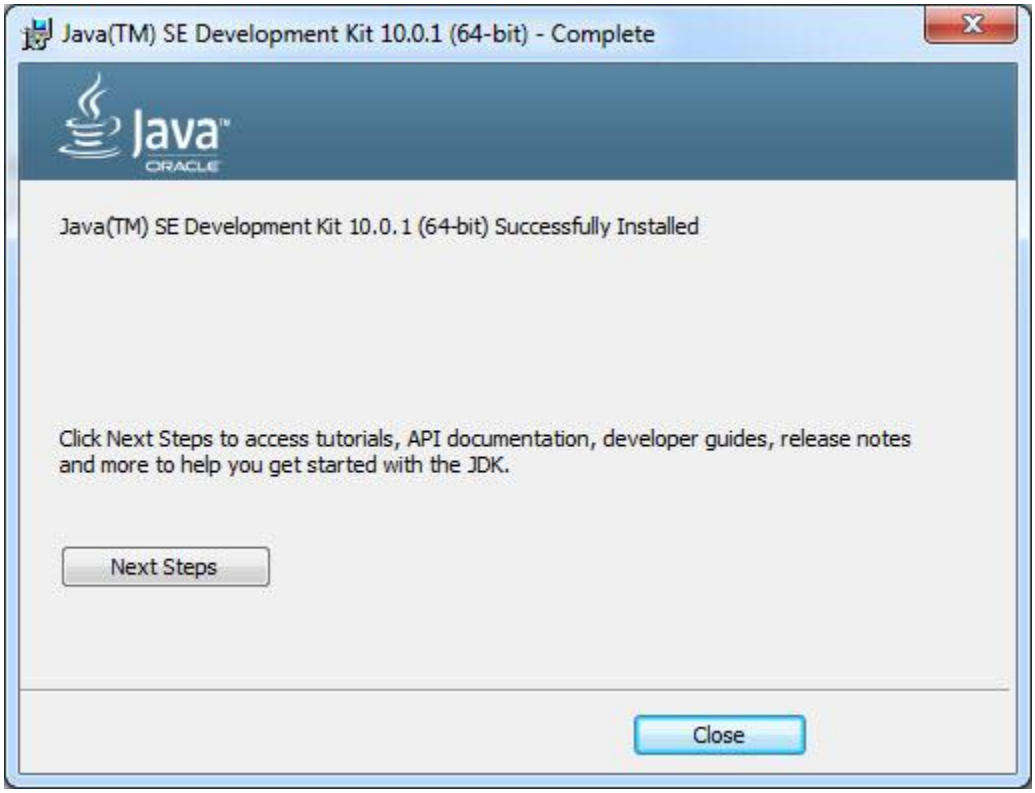
نصب JDK و IntelliJ IDEA

در درس قبل در مورد نرم افزارهای IntelliJ IDEA و JDK توضیحات مختصری ارائه دادیم. در این درس می خواهیم شما را با نحوه نصب این دو نرم افزار آشنا کنیم. نصب این نرم افزارها مانند اکثر نرم افزارهای دیگر بسیار آسان بود و بعد از زدن چند دکمه Next نصب می شوند. در لینکی که در درس قبل دانلود کردیه اید، دو نسخه ۳۲ و ۶۴ بیت از JDK وجود دارد که شما بسته به نوع ویندوز خود باید یکی از این دو را نصب کنید. در این درس چون که ویندوز ما ۶۴ بیت است، پس مراحل نصب نسخه ۶۴ بیتی JDK را به شما آموزش می دهیم. البته مراحل نصب هر دو نسخه دقیقا شبیه هم است. در در زیر مراحل تصویری نصب این دو نرم افزار نشان داده شده است .

نصب JDK

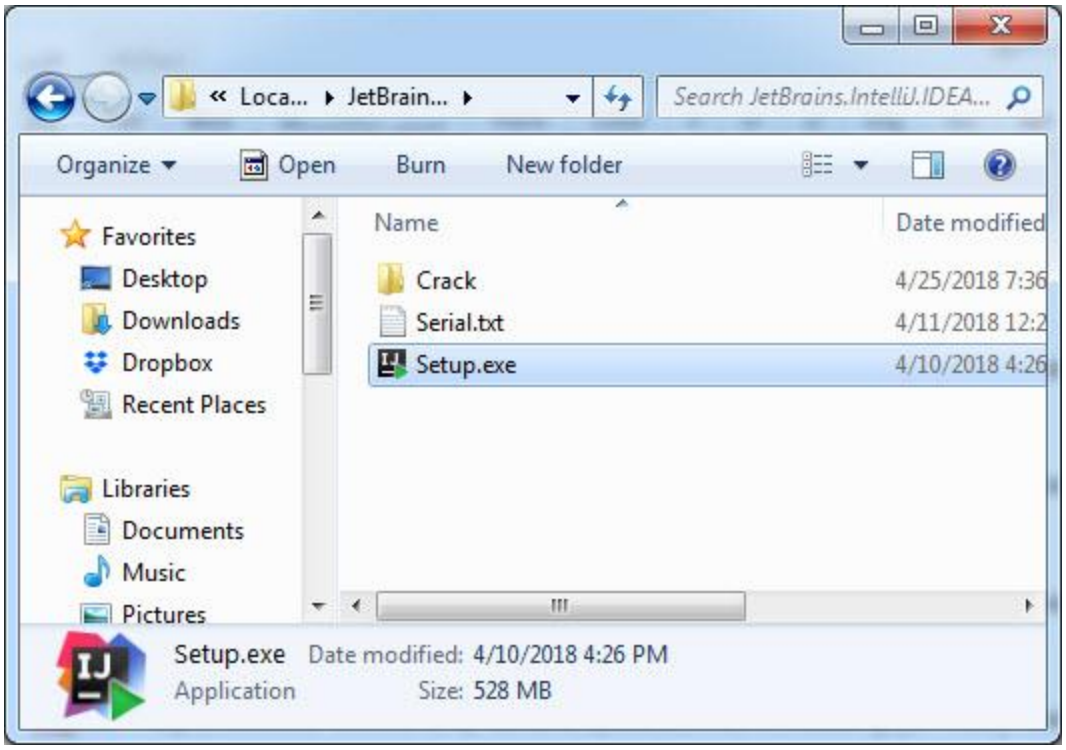






نصب IntelliJ IDEA

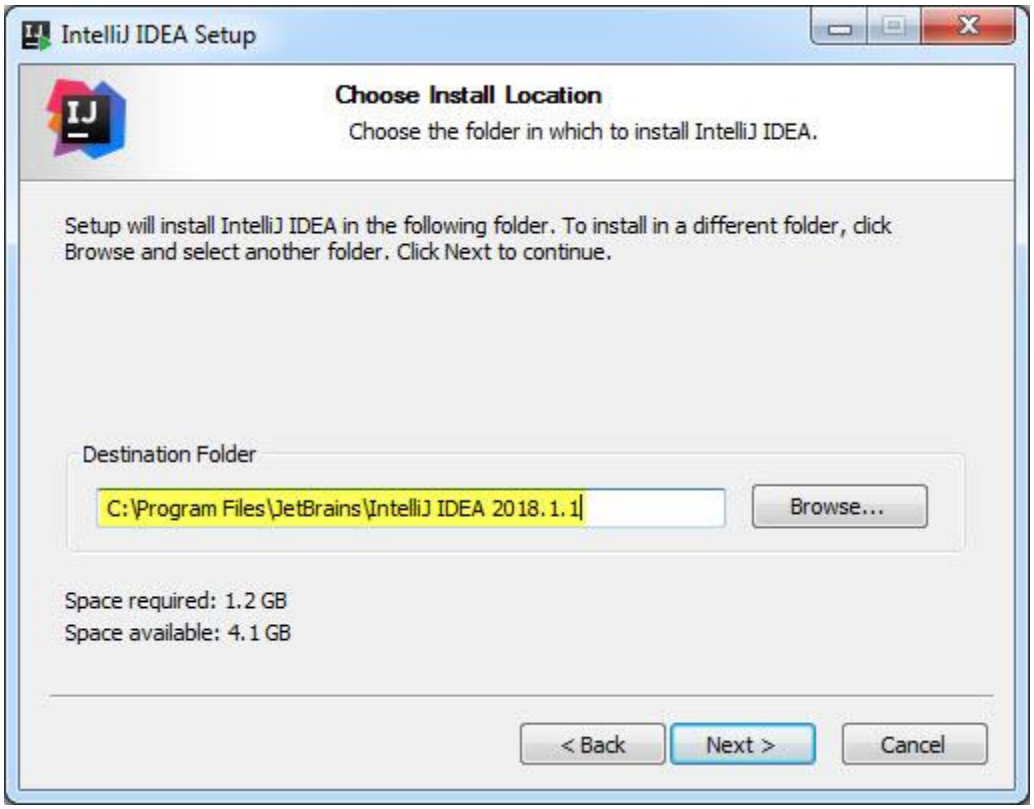
نصب IntelliJ IDEA کمی نیاز به دقت دارد. ابتدا به پوشه نرم افزار بروید:



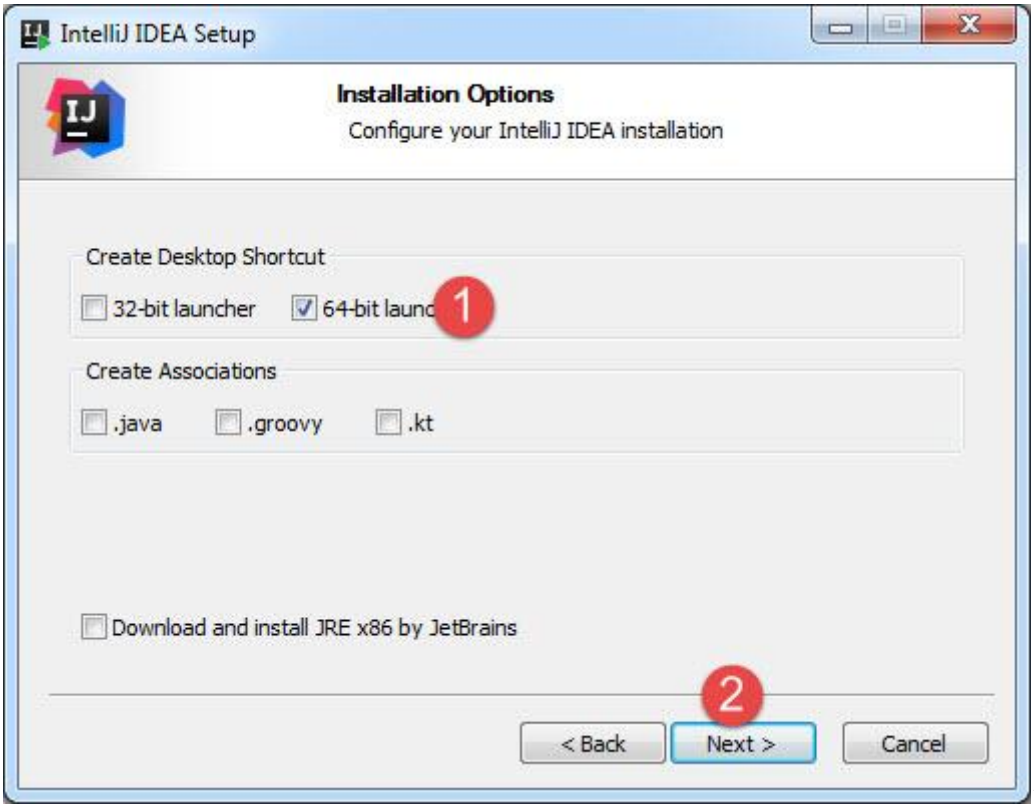
فایل Setup.exe را اجرا کرده تا وارد مراحل نصب نرم افزار شوید. در شکل زیر دکمه Next را کلیک کنید:

در صفحه زیر مسیری که قرار است برنامه نصب شود، نشان داده شده است. این مسیر ممکن است در کامپیوتر شما متفاوت باشد.

در هر صورت این مسیر مهم است، و در مراحل بعدی با آن سر و کار داریم. آن را دستکاری نکنید و دکمه `Next` را بزنید:

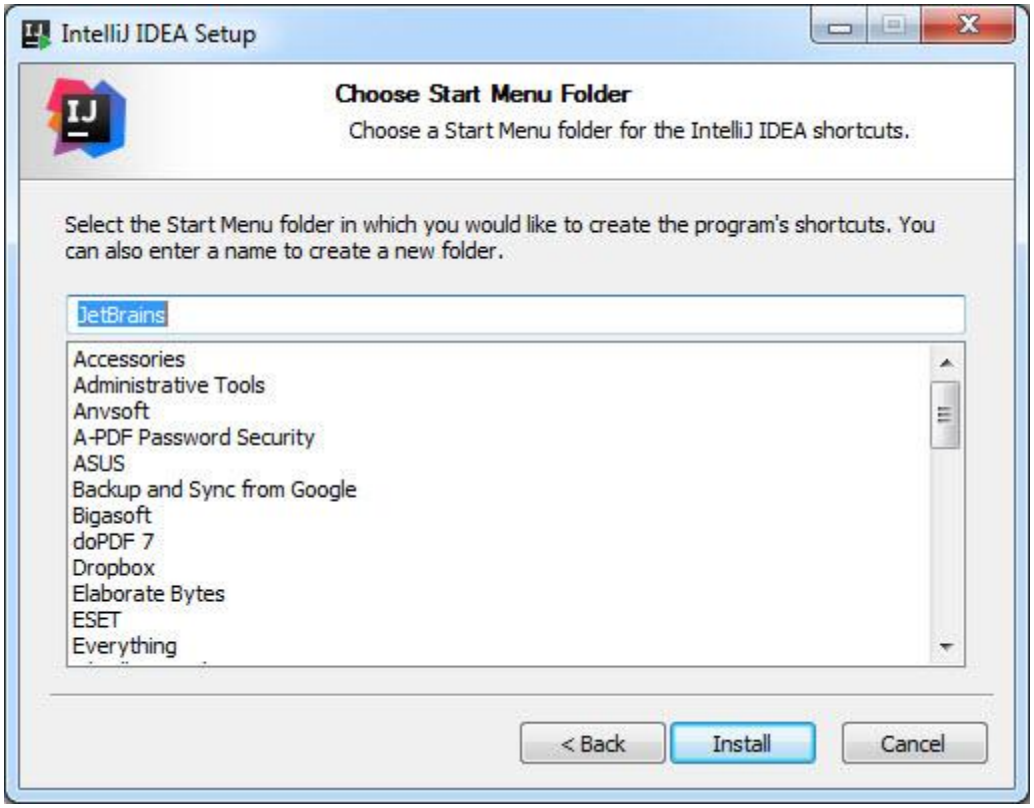


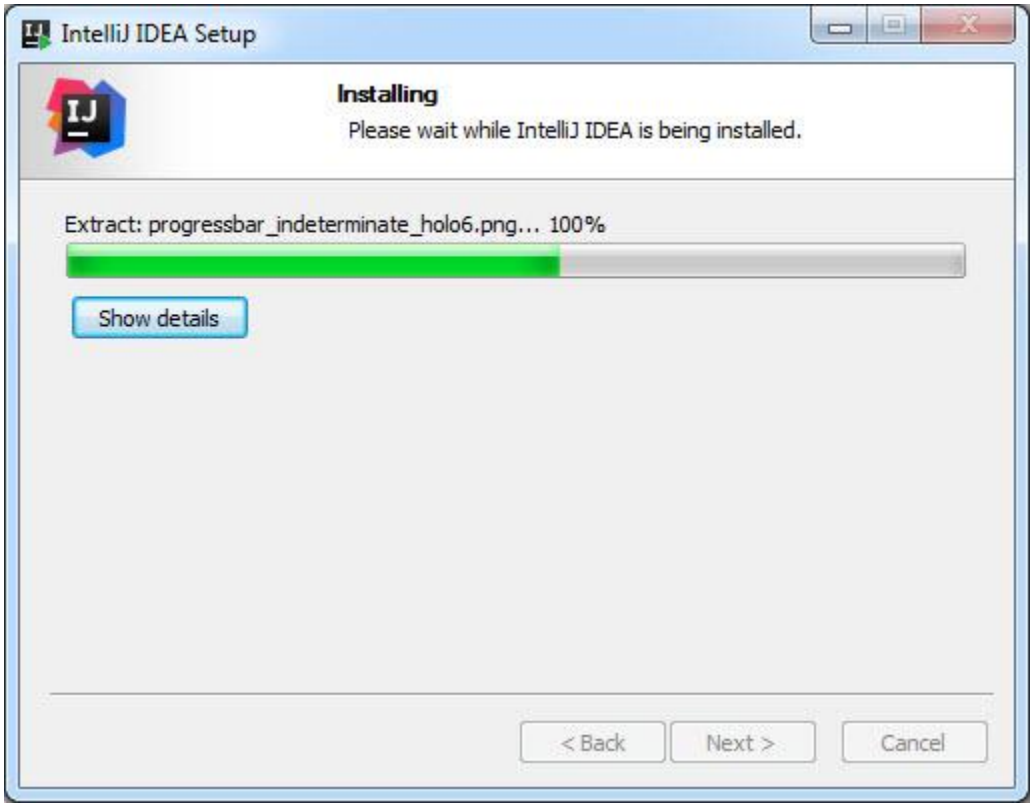
در صفحه زیر بسته به اینکه ویندوزتان ۳۲ یا ۶۴ بیتی است، گزینه مرتبط را تیک زده و سپس دکمه Next را بزنید:



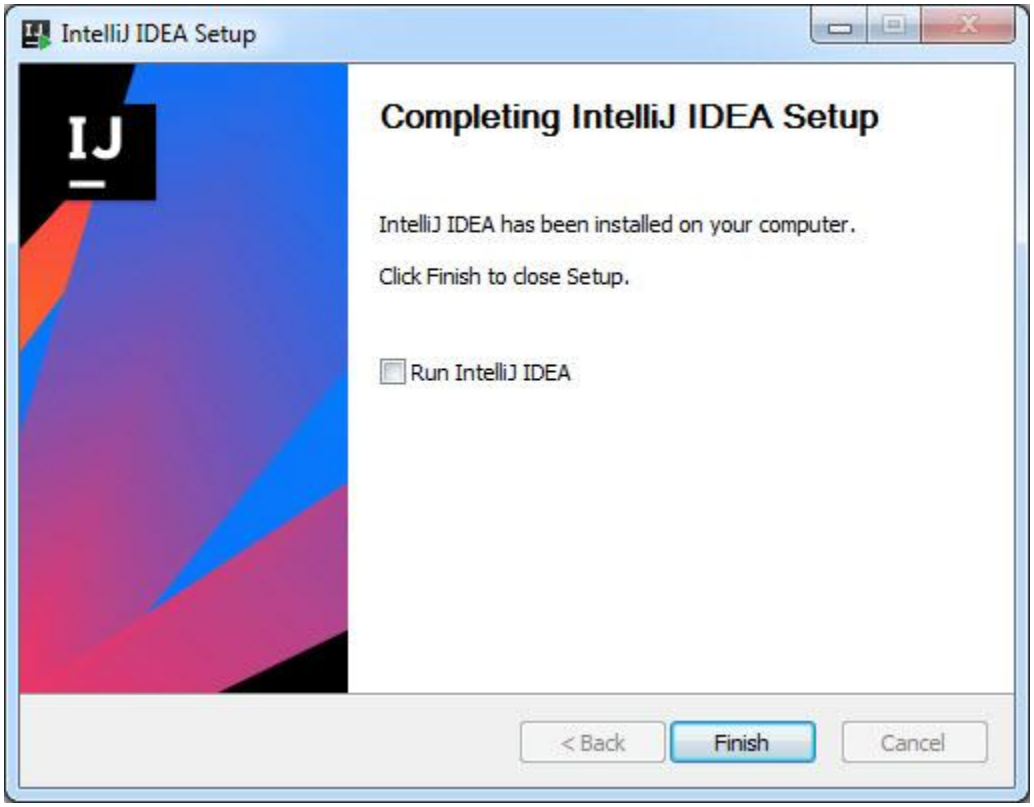
۳

در شکل زیر دکمه `Install` را زده تا وارد مرحله پایانی نصب شوید.

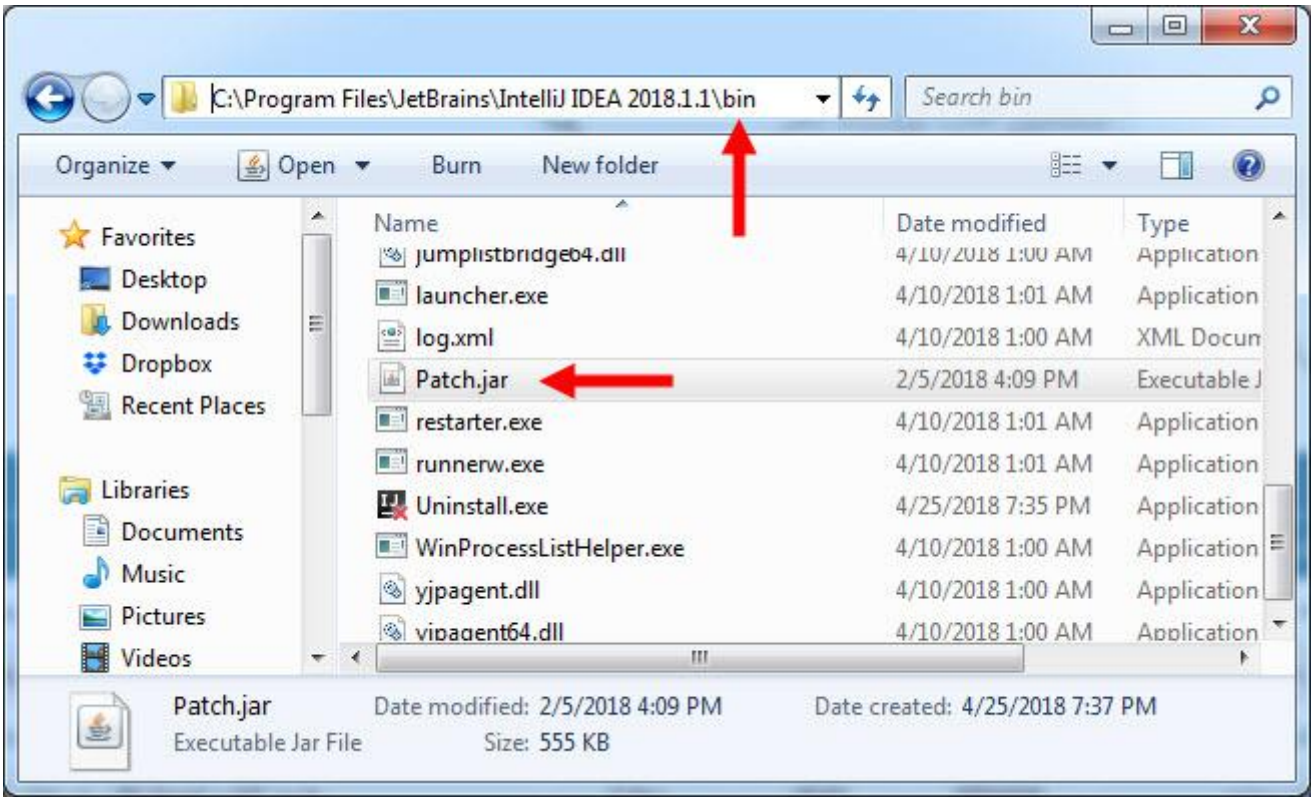




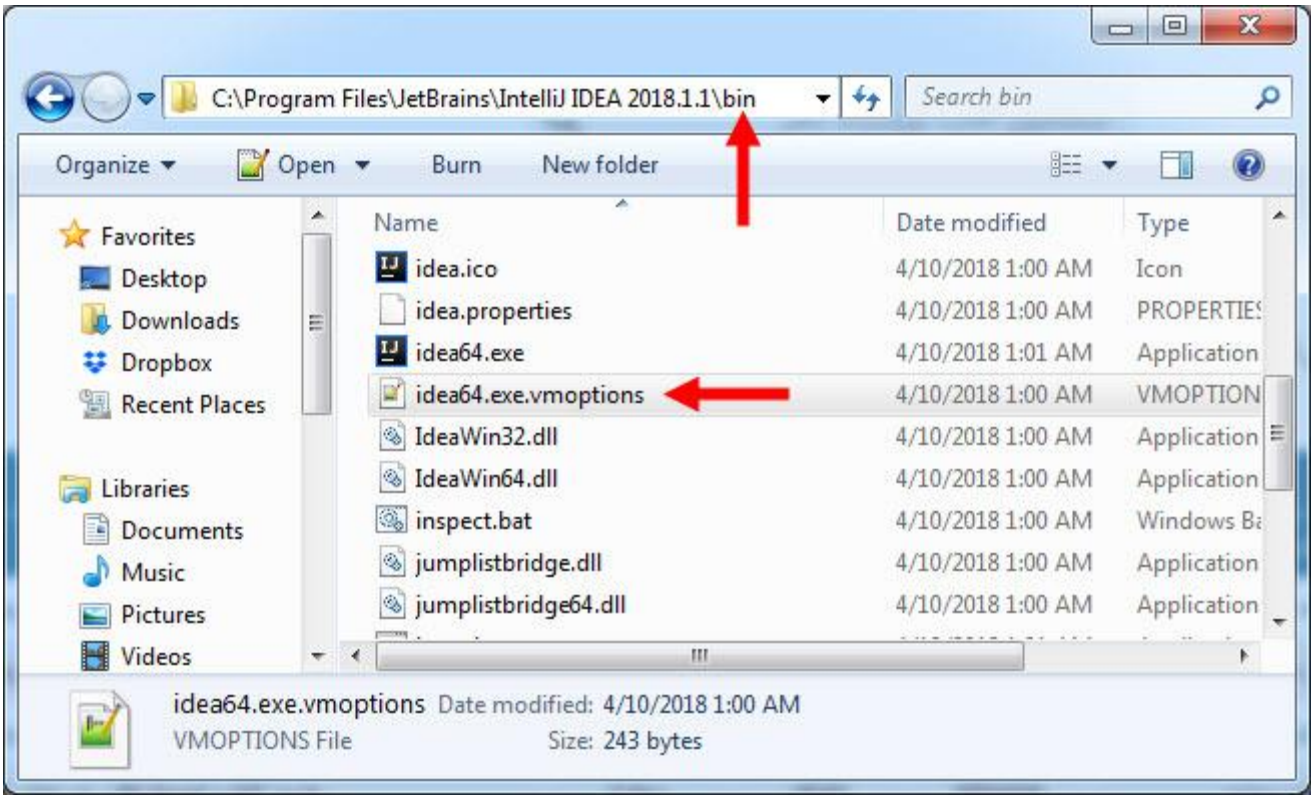
با زدن دکمه Close در شکل زیر، نصب برنامه به اتمام می رسد. و نوبت به کرک کردن آن می رسد:



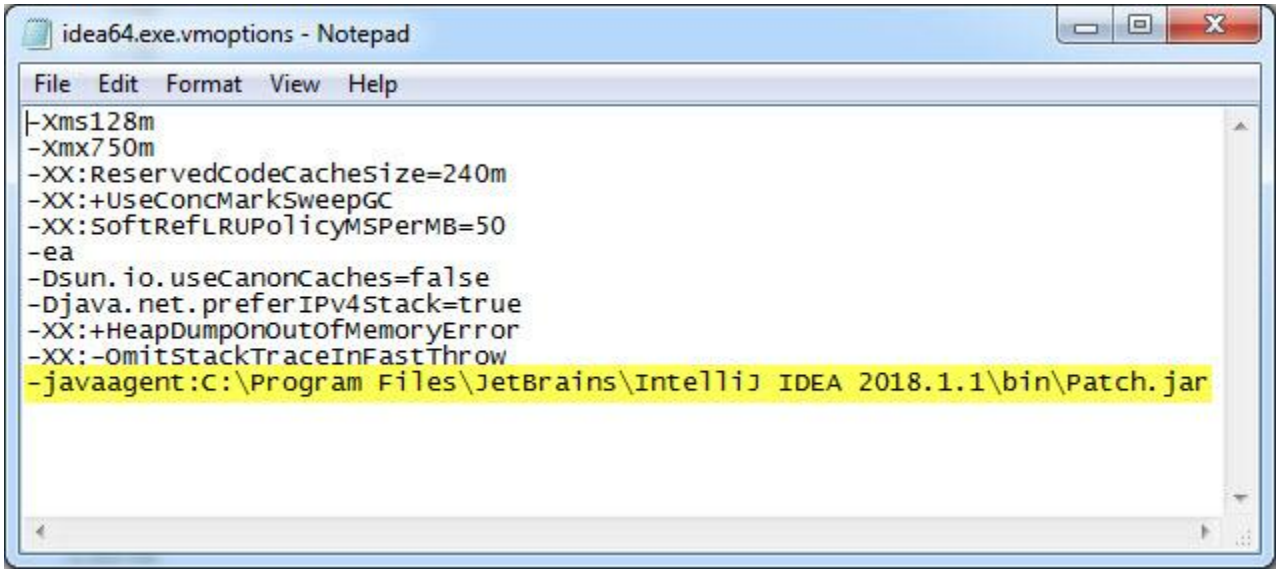
حال به پوشه برنامه که در شکل اول نشان داده شد، بروید و در داخل پوشه Crackآن فایل Patch.jar را Copy و در مسیری که نرم افزار IntelliJ IDEA نصب شده، در داخل پوشه bin، Paste کنید:



حال در داخل همین پوشه `bin` دو فایل با پسوند `vmoptions` هستند که یکی برای ویندوزهای ۶۴ و دیگری برای ویندوزهای ۳۲ می باشد. بسته به نوع ویندوزتان یکی از این دو را با `NotePad` باز کرده:

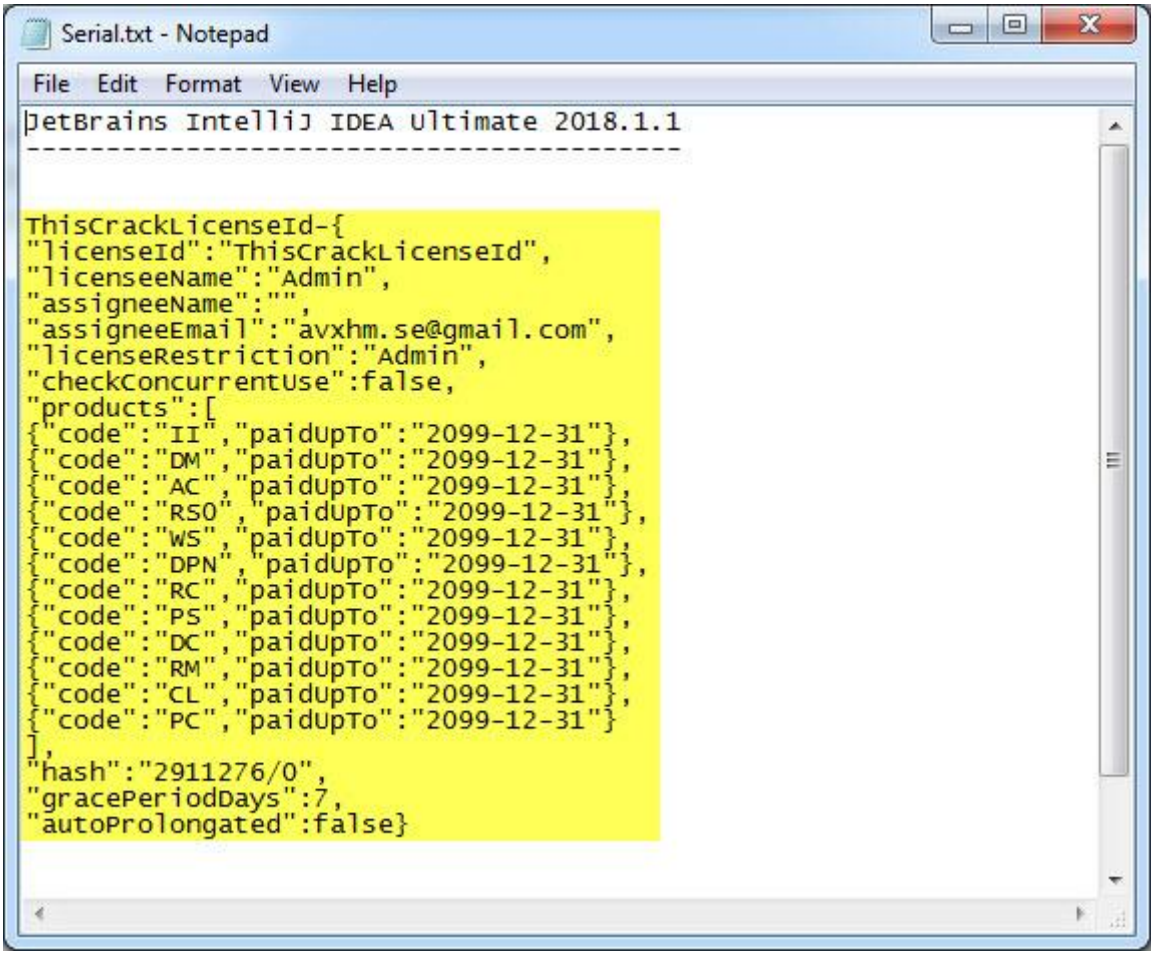


و مسیر فایل `Patch.jar` را در داخل آن به صورت زیر نوشته و ذخیره کنید:

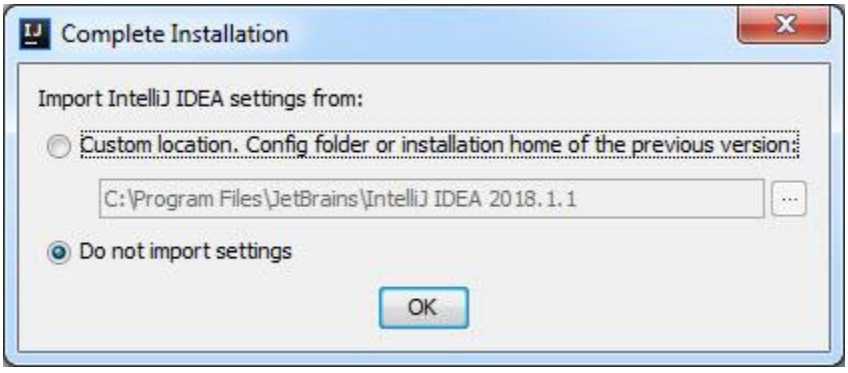


دوباره به پوشه برنامه که در شکل اول نشان داده شده است رفته و فایل `Serial.txt` را باز کرده و کدهایی که در شکل زیر نشان

داده شده است را کپی کرده:

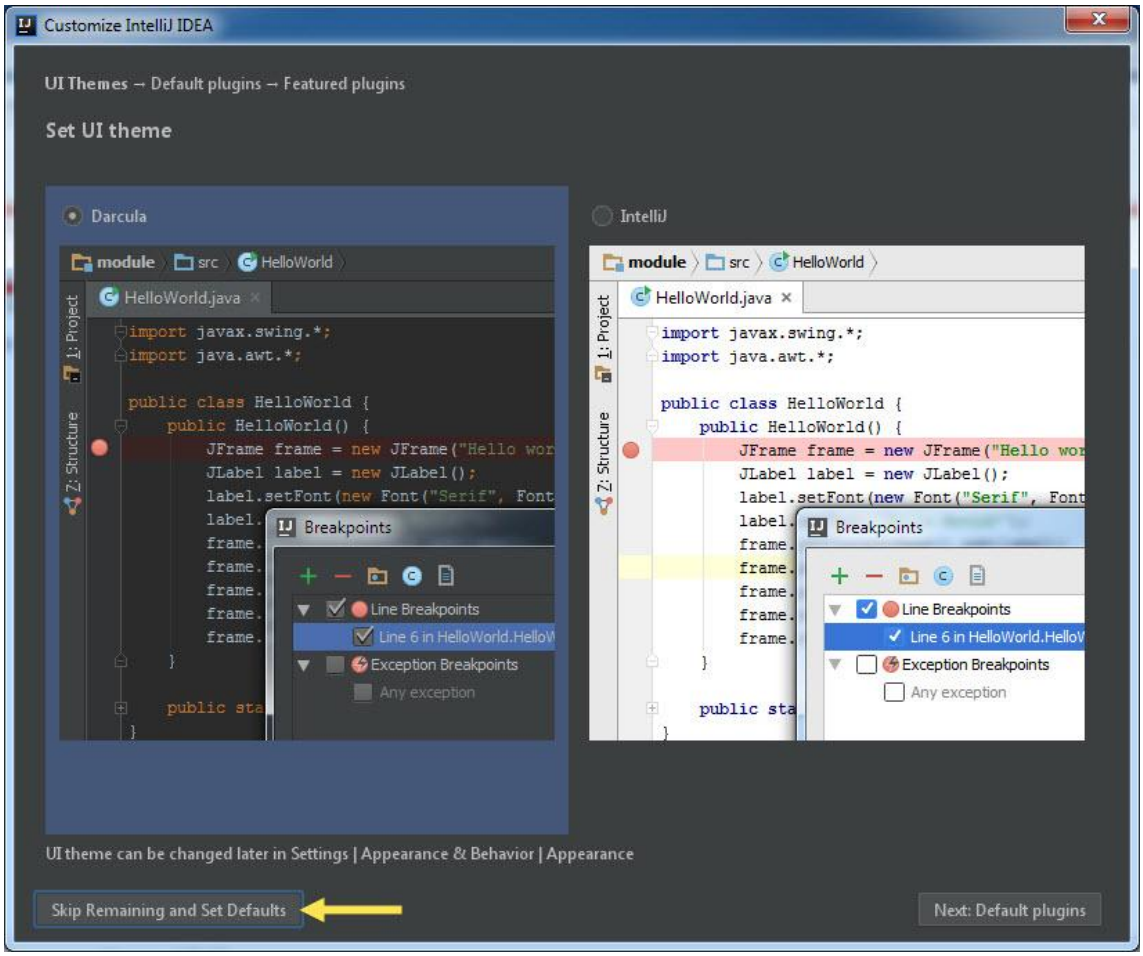


سپس برنامه را اجرا و دکمه Ok را در شکل زیر بزنید:

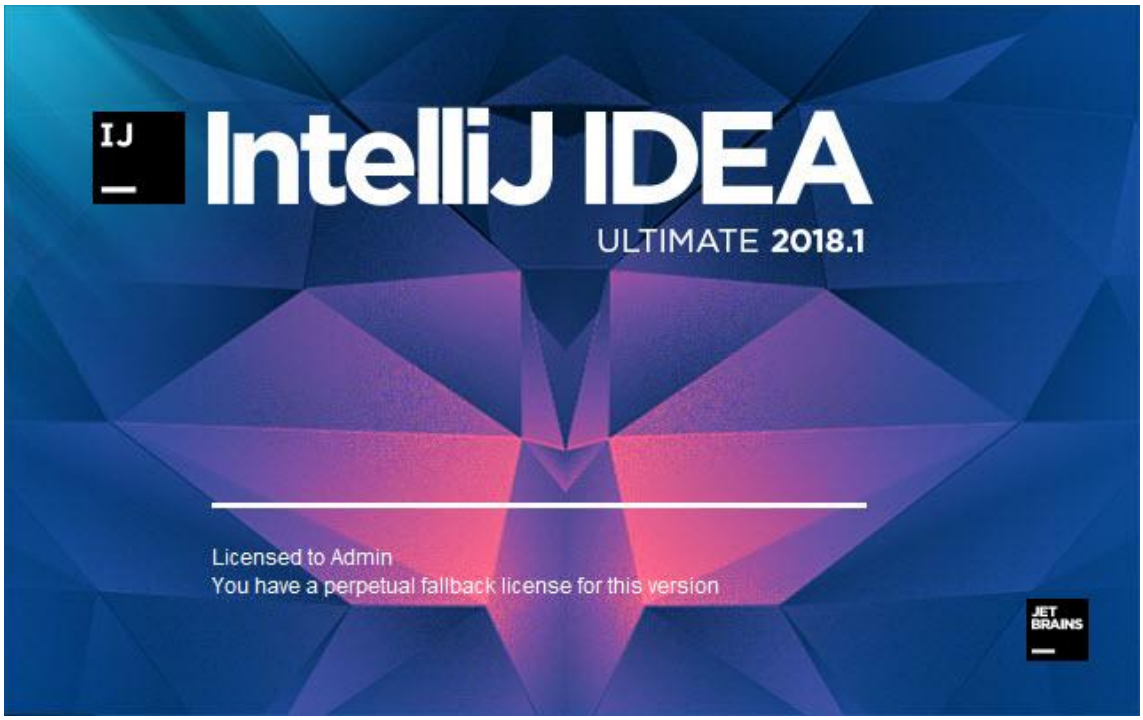


در شکل زیر سریالی را که کپی کرده اید، به صورت زیر Patse نمایید و دکمه OK را فشار دهید:

بعد از دکمه OK در شکل بالا شکل زیر طاهر می شود. در این شکل دکمه Skip Remaning and Set Defaults را بفشارید:

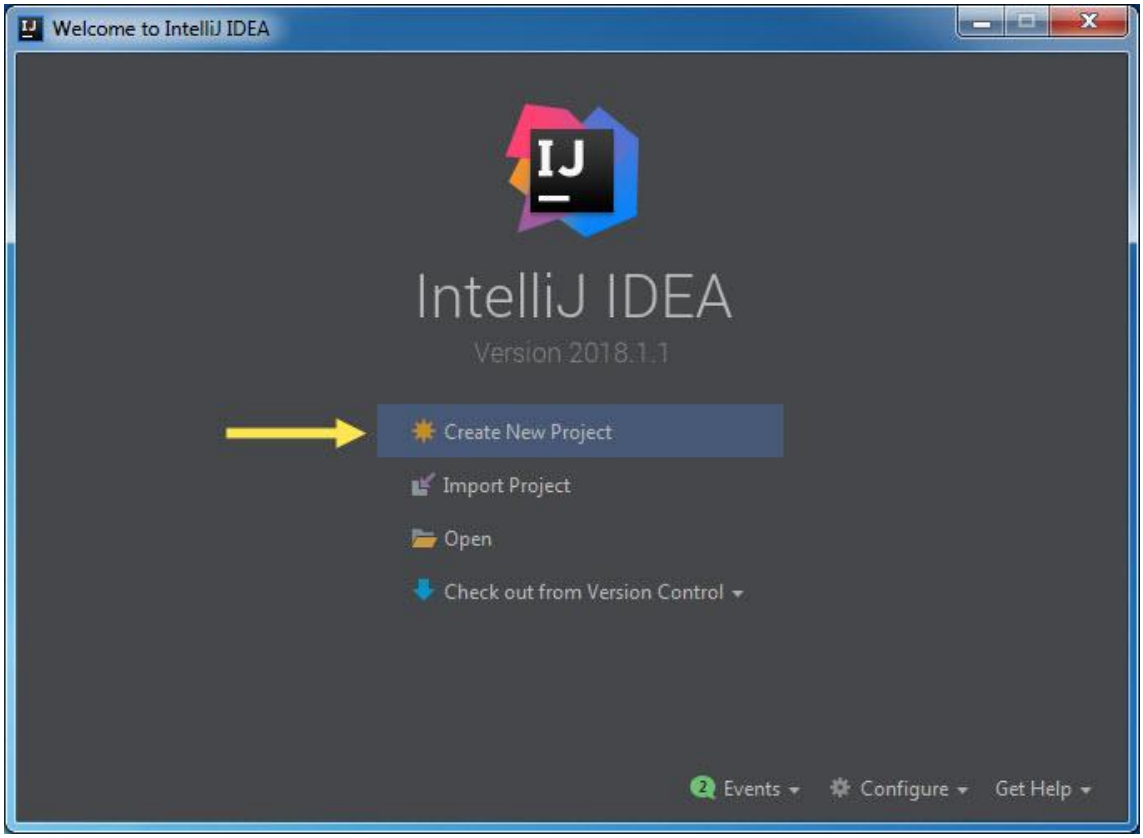


با ظاهر شدن شکل زیر نصب و کرک نرم افزار تمام می شود و شما می توانید با خیال راحت در آن کدنویسی کنید:

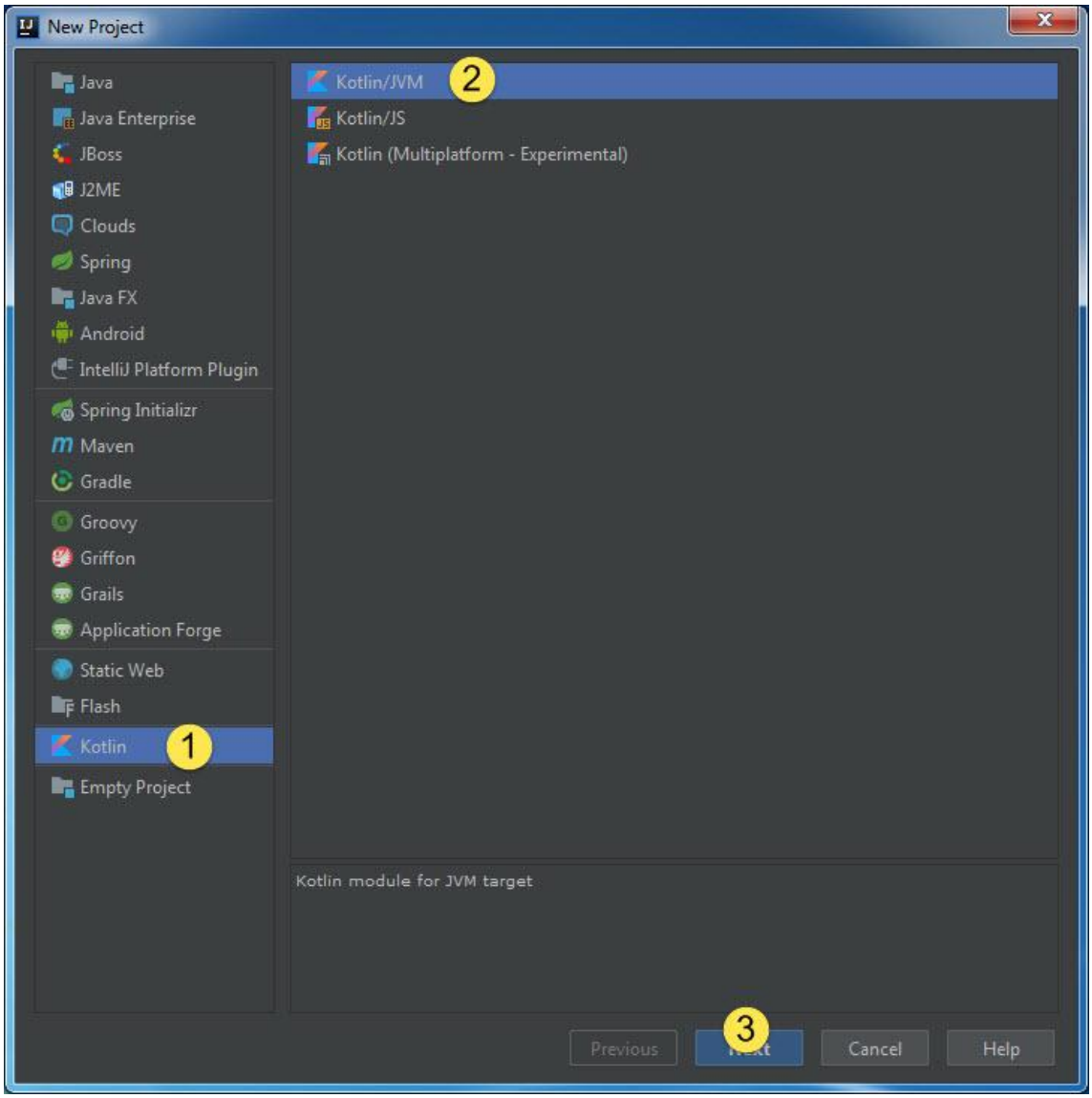


ساخت یک برنامه ساده در Kotlin

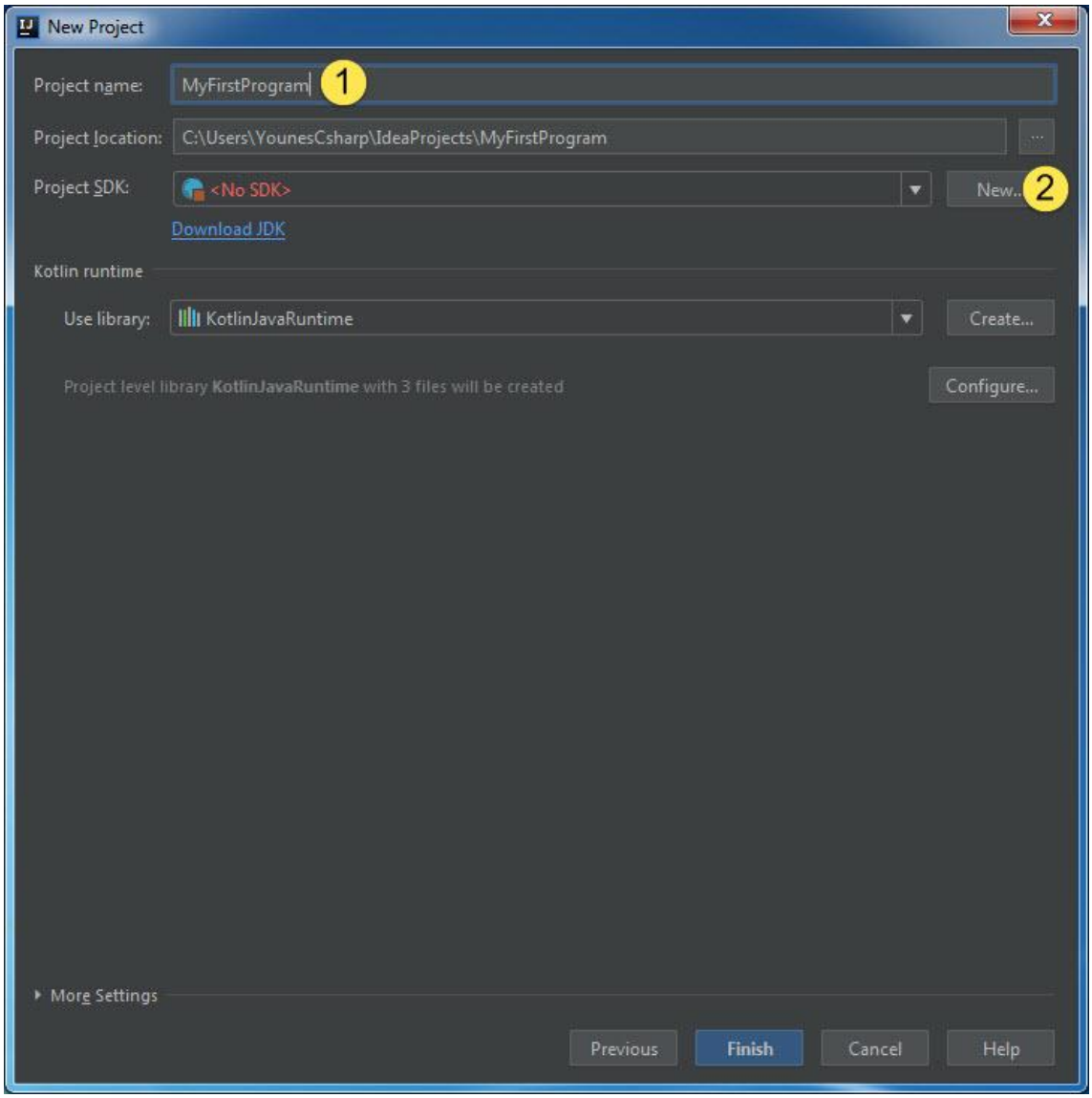
اجازه بدهید یک برنامه بسیار ساده به زبان Kotlin بنویسیم. این برنامه یک پیغام را نمایش می‌دهد. در این درس می‌خواهم ساختار و دستور زبان یک برنامه ساده Kotlin را توضیح دهم. ابتدا برنامه Intellij IDEA را اجرا کنید:



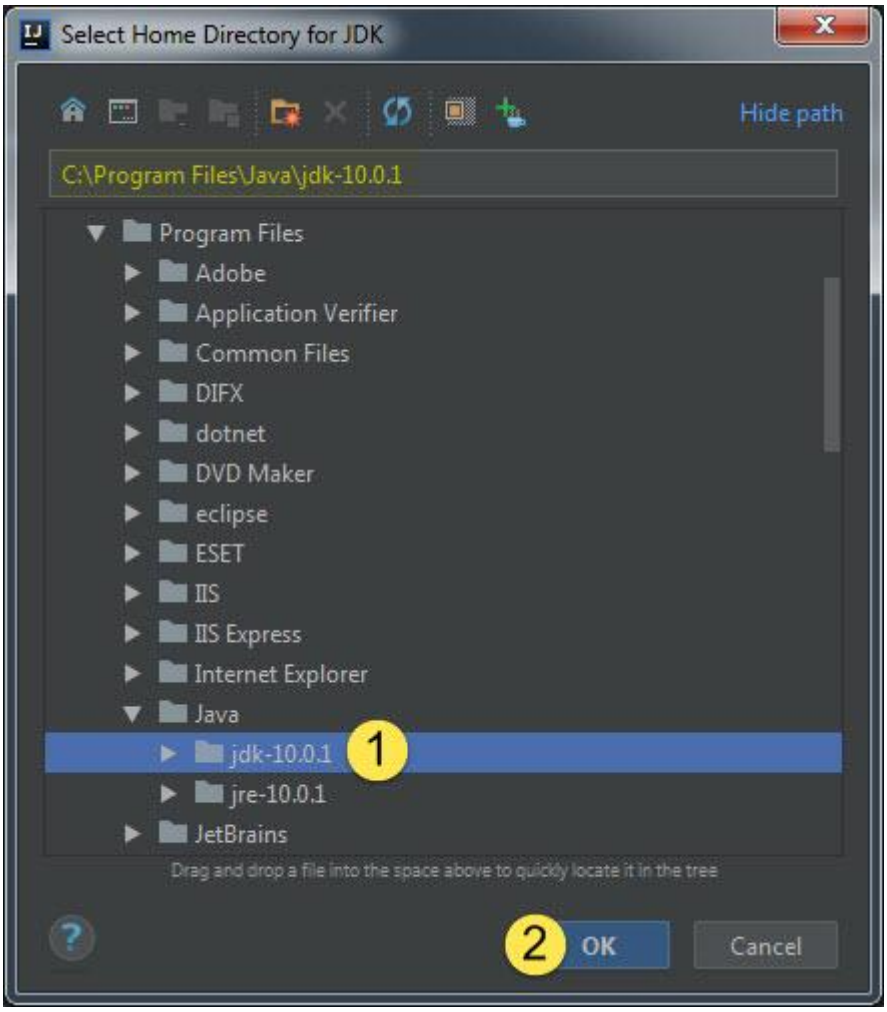
در شکل بالا، بر روی گزینه `Create New Project` کلیک کرده و در صفحه‌ای که به صورت زیر نمایش داده می‌شود هم، مراحل ۱ تا ۳ شکل زیر را طی نمایید:



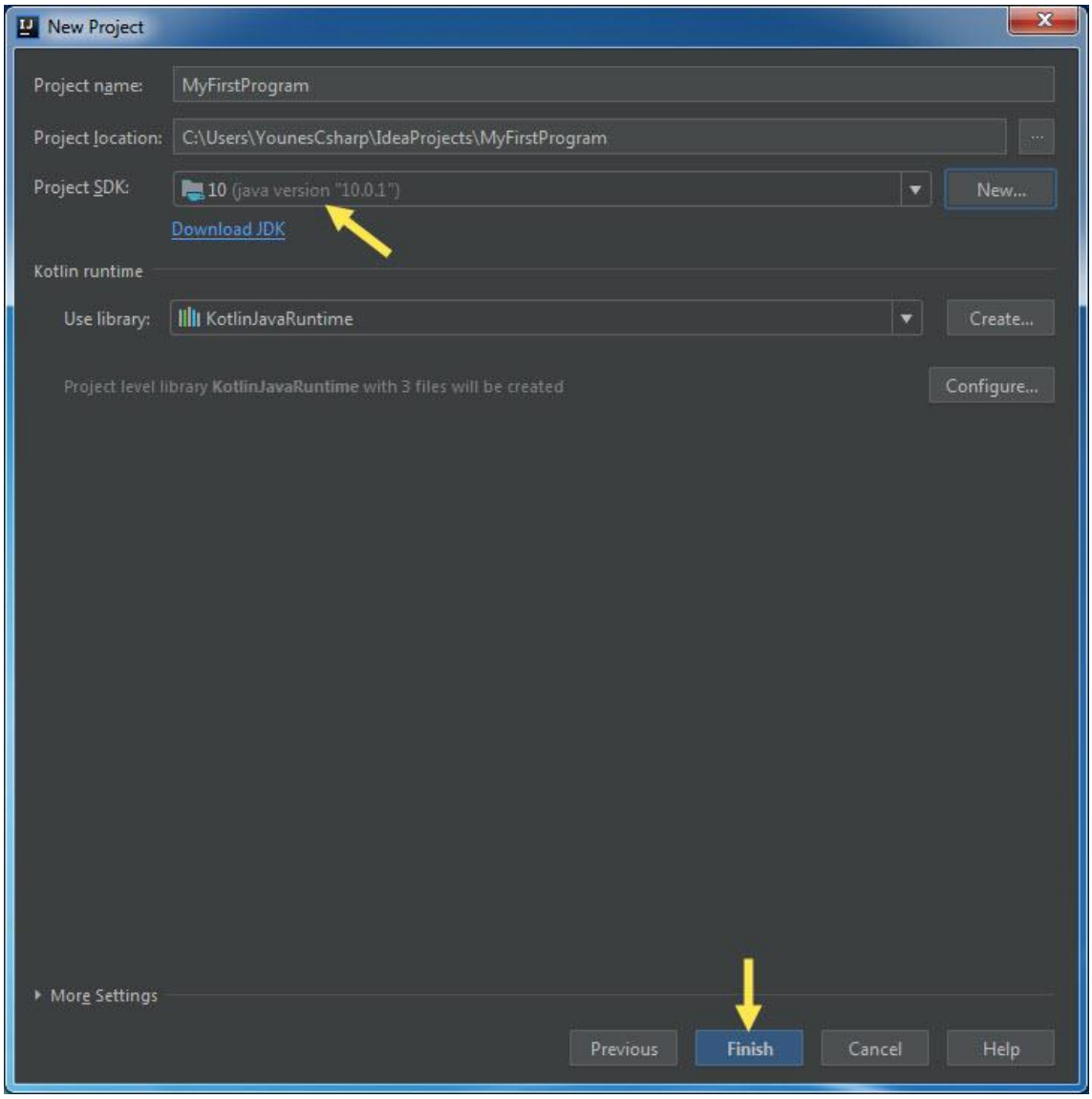
در شکل زیر یک نام برای پروژه‌تان انتخاب کرده و سپس بر روی گزینه New کلیک نمایید:



حال صفحه‌ای به صورت زیر نمایش داده می‌شود که از شما مسیر نصب فایل JDK را می‌خواهد. در درایو که آن را نصب کرده‌اید مسیر پوشه JDK را پیدا کرده و دکمه OK را کلیک کنید:

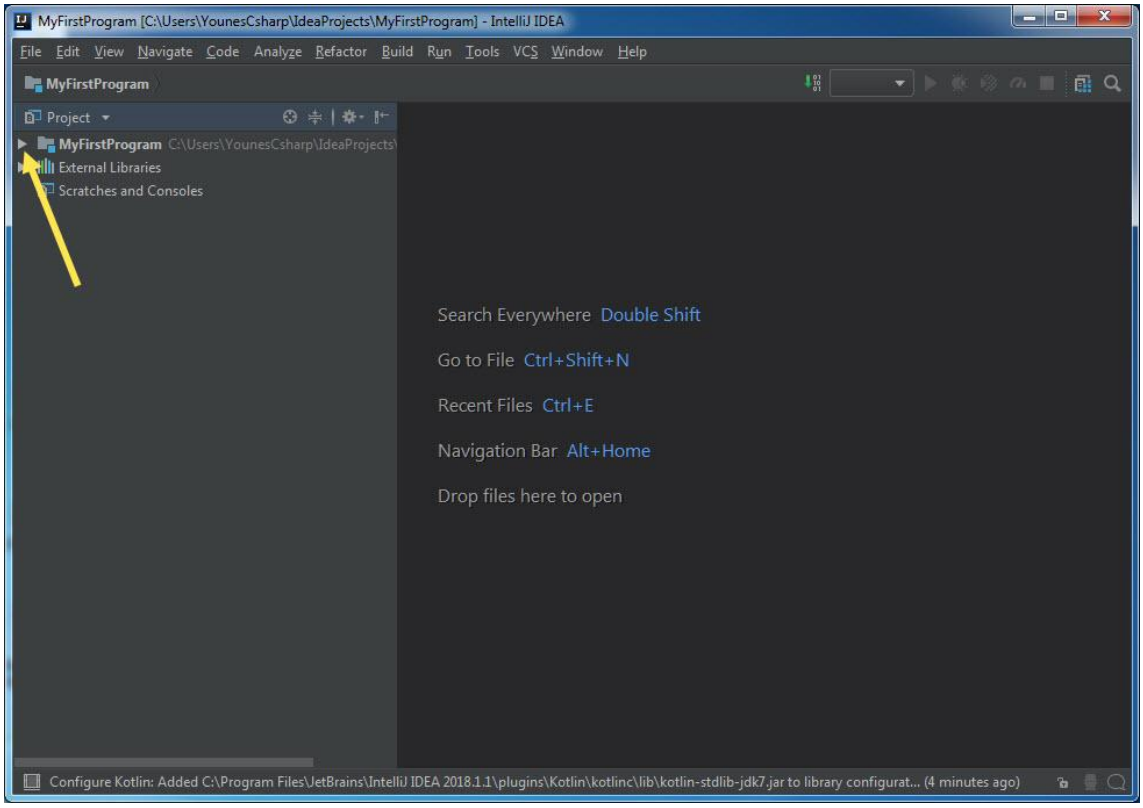


مشاهده می‌کنید که نام و نسخه JDK در کادر مربوطه نمایش داده می‌شود. سپس بر روی دکمه Finish کلیک نمایید:

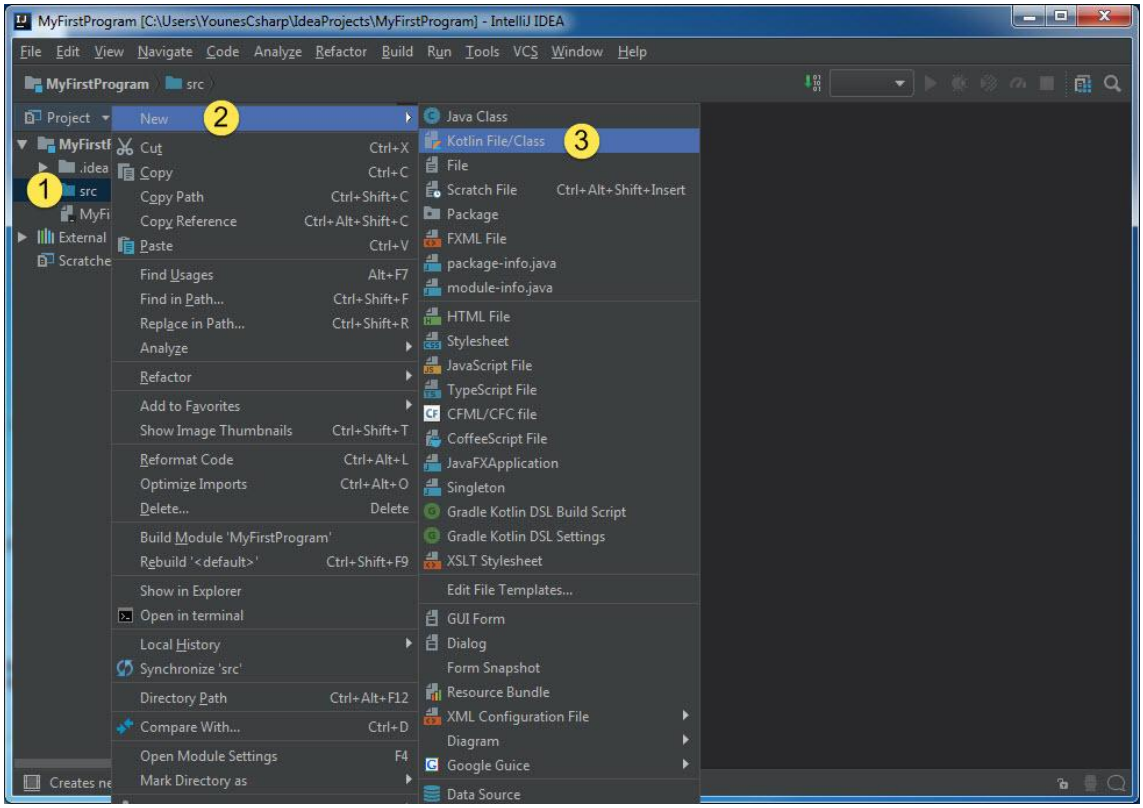


اگر صفحه‌ای به صورت زیر نمایش داده شد، تیک را بر داشته و دکمه Close را بزنید:

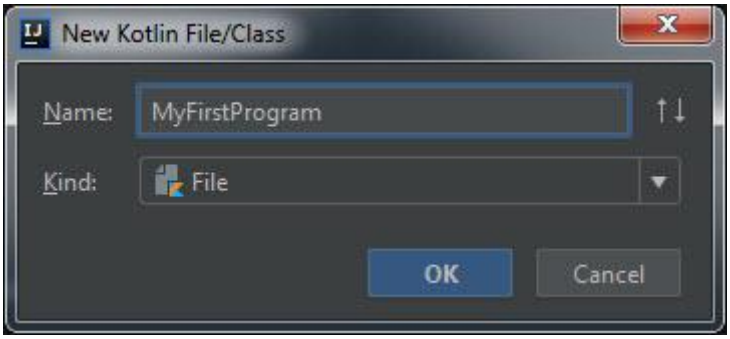
در شکل زیر بر روی فلش کوچک کنار نام پروژه کلیک کرده تا زیر مجموعه‌های آن نمایش داده شوند:



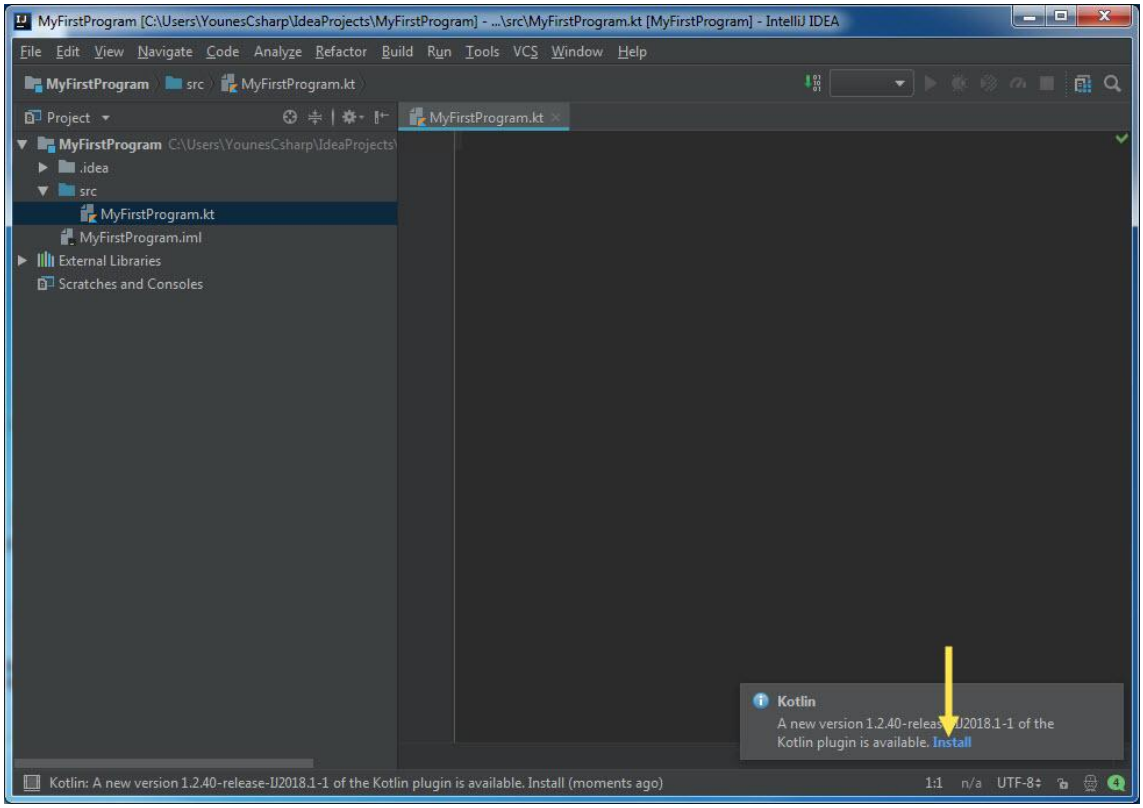
سپس بر روی پوشه src راست کلیک کرده و مراحل زیر را طی کنید:



در شکل زیر یک نام برای فایلی که قرار است ایجاد نمایید، انتخاب و سپس بر روی دکمه OK کلیک کنید:



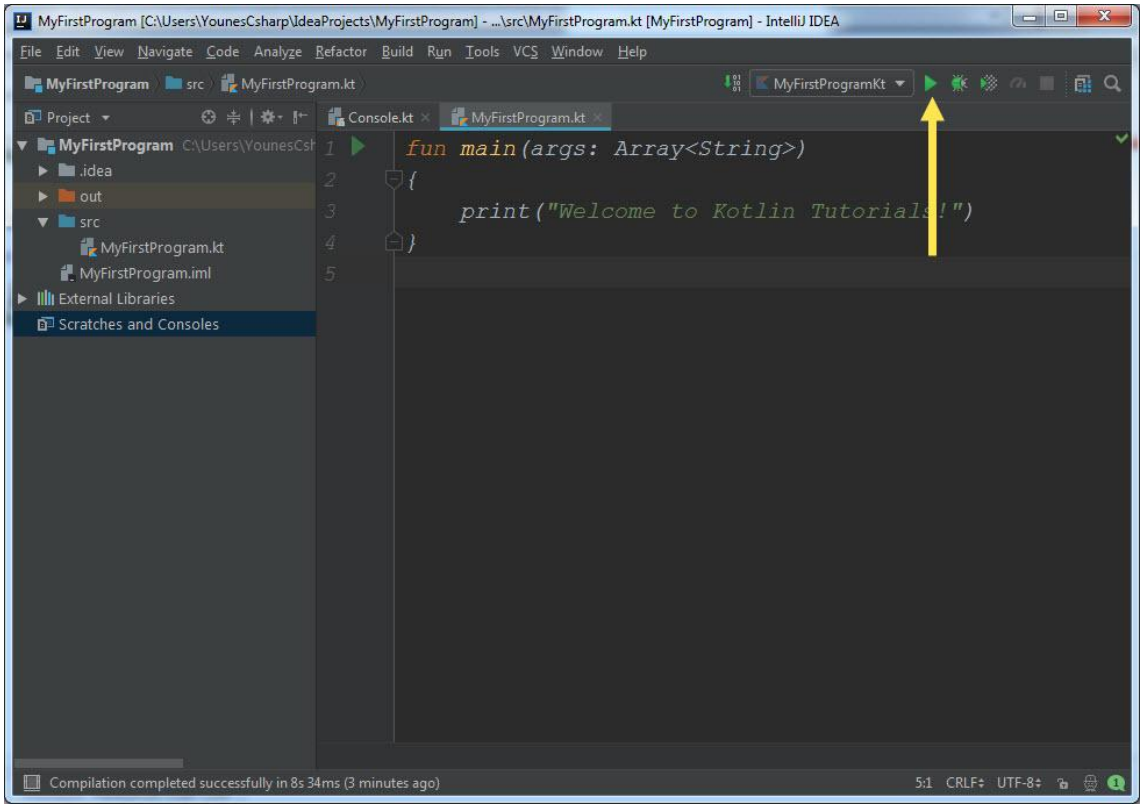
اگر پیغامی به صورت زیر مشاهده کردید بر روی دکمه Install کلیک کنید:



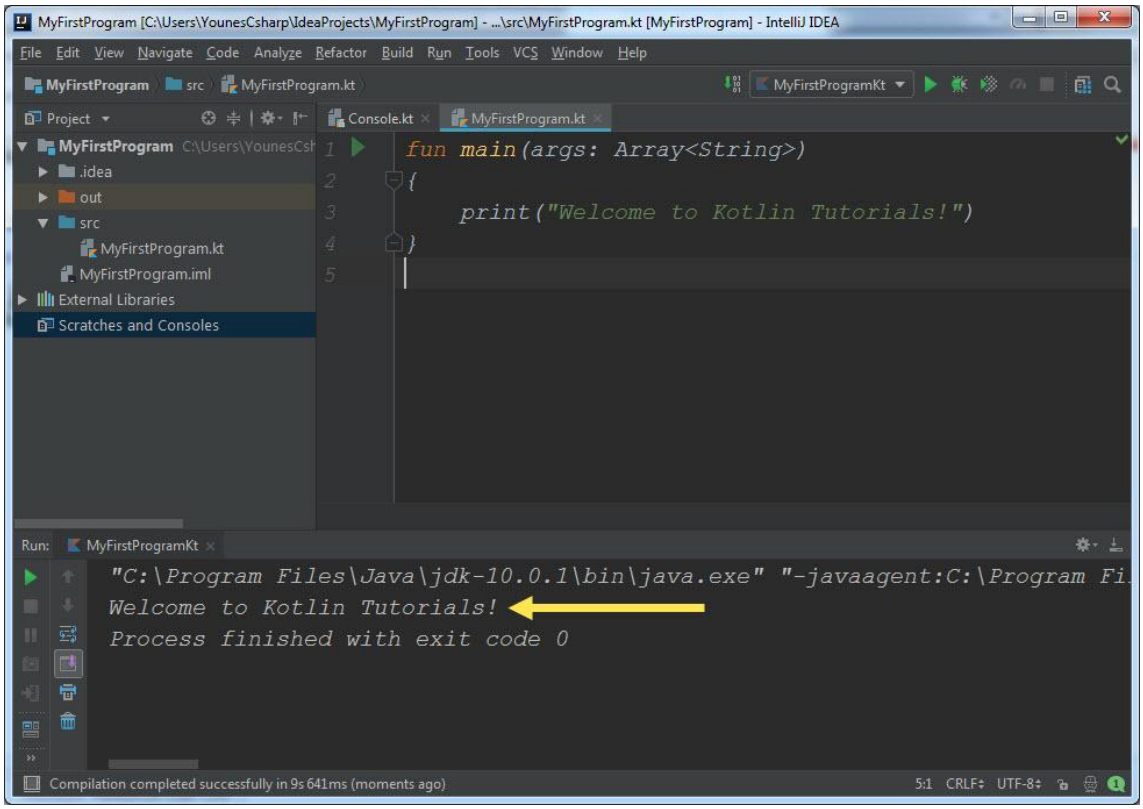
حال برنامه آماده کدنویسی است. کدهای زیر را داخل محیط کدنویسی بنویسید:

```
fun main(args: Array<String>)
{
   print("Welcome to Kotlin tutorials!")
}
```

محیط کدنویسی یا IDE جایی است که ما کدها را در آن تایپ می‌کنیم. کدها در محیط کدنویسی به صورت رنگی تایپ می‌شوند در نتیجه تشخیص بخشهای مختلف کد را راحت می‌کند. برای اجرای برنامه بر روی فلش کوچک و سبز رنگ کلیک کرده:



برنامه اجرا شده و پیغام Welcome to Kotlin Tutorials! نمایش داده می‌شود.



ساختار یک برنامه در کاتلین

مثال بالا ساده‌ترین برنامه‌ای است که شما می‌توانید در Kotlin بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است:

```
fun main(args: Array<String>)
{
   print("Welcome to Kotlin tutorials!")

}
```

هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. اجازه بدهید هر خط کد را در مثال بالا توضیح بدهیم. خط ۱ تابع main() یا تابع اصلی نامیده می‌شود. هر تابع شامل یک سری کد است که وقتی اجرا می‌شوند که تابع را صدا بزنیم. درباره تابع و نحوه صدا زدن آن در فصول بعدی توضیح خواهیم داد. تابع main() نقطه آغاز اجرای برنامه است. این بدان معناست که ابتدا تمام کدهای داخل تابع main() و سپس بقیه کدها اجرا می‌شود. درباره تابع main() در فصول بعدی توضیح خواهیم داد. تابع

main() و سایر توابع دارای آکولاد و کدهایی در داخل آن‌ها می‌باشند و وقتی کدها اجرا می‌شوند که توابع را صدا بزنیم. در خط ۲ آکولاد ({) نوشته شده است. آکولاد برای تعریف یک بلوک کد به کار می‌رود. کاتلین یک زبان ساخت یافته است که شامل کدهای زیاد و ساختارهای فراوانی می‌باشد. هر آکولاد باز ({) در کاتلین باید دارای یک آکولاد بسته (}) نیز باشد. همه کدهای نوشته شده از خط ۲ تا خط ۴ یک بلوک کد است. مثالی از یک خط کد در کاتلین به صورت زیر است:

```
print("Welcome to Kotlin tutorials!")
```

این خط کد پیغام Welcome to Kotlin Tutorials! را در صفحه نمایش نشان می‌دهد. از تابع print() برای چاپ یک رشته استفاده می‌شود. یک رشته گروهی از کاراکترها است که به وسیله دابل کوتیشن (") محصور شده است. مانند:

```
"Welcome to Visual Kotlin Tutorials!"
```

یک کاراکتر می‌تواند یک حرف، عدد، علامت یا .... باشد. در کل مثال بالا نحوه استفاده از تابع print() نشان داده شده است. کاتلین فضای خالی و خطوط جدید را نادیده می‌گیرد. بنابراین شما می‌توانید همه برنامه را در یک خط بنویسید به شرط اینکه بین هر دو دستور علامت سمیکالن بگذارید:

```
print("Welcome to "); print("Kotlin Tutorials!")
```

اما اینکار خواندن و اشکال زدایی برنامه را مشکل می‌کند. همیشه به یاد داشته باشید که کاتلین به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال MAN و man در کاتلین با هم فرق دارند. رشته‌ها و توضیحات از این قاعده مستثنی هستند که در درسهای آینده توضیح خواهیم داد. مثلاً کدهای زیر با خطا مواجه می‌شوند و اجرا نمی‌شوند:

```
Print("Welcome to Kotlin Tutorials!")
PRINT("Welcome to Kotlin Tutorials!")
pRinT("Welcome to Kotlin Tutorials!")
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می‌کند. اما کد زیر کاملاً بدون خطا است:

```
print("Welcome to Kotlin Tutorials!")
```

همیشه کدهای خود را در داخل آکولاد بنویسید.

```
{
  statement1
}
```

این کار باعث می‌شود که کدنویسی شما بهتر به چشم بیاید و تشخیص خطاها راحت‌تر باشد. یکی از ویژگیهای مهم Intellij IDEA نشان دادن کدها به صورت تو رفتگی است بدین معنی که کدها را به صورت تو رفتگی از هم تفکیک می‌کند و این در خوانایی برنامه بسیار مؤثر است.

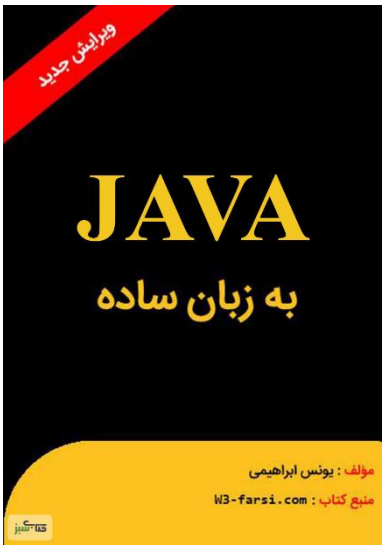
مقایسه با جاوا

کسانی که با جاوا آشنایی دارند، می‌دانند که برای پیاده سازی کد همین درس در جاوا باید به صورت زیر عمل شود:

```
class MyFirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

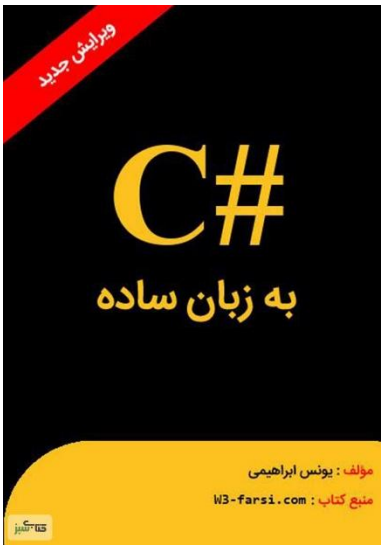
در بیان تفاوت بین کدهای جاوا و کاتلین به همین نکات زیر بسنده می‌کنیم:

- کدهای کاتلین نسب به جاوا خلاصه‌تر هستند.
- در برنامه‌های جاوا لازم است که یک کلاس ایجاد شود، در حالی که در کاتلین این کار توسط کامپایلر به صورت خودکار انجام می‌شود.
- برای استفاده از برخی از توابع مانند print() لازم نیست که کلاسی که این تابع در داخل آن قرار دارد، ذکر شود.



برای دانلود به لینک زیر مراجعه نمایید:

https://goo.gl/oD1AmH



برای دانلود به لینک زیر مراجعه نمایید:

https://goo.gl/2snnQM

کاراکترهای کنترلی

کاراکترهای کنترلی کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آنها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد:

```
print("Hello\nWorld!")
```

```
Hello
World
```

مشاهده کردید که کامپایلر بعد از مواجهه با کاراکتر کنترلی \n نشانگر ماوس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می‌دهد. متد Println() هم مانند کاراکتر کنترلی \n یک خط جدید ایجاد می‌کند، البته بدین صورت که در انتهای رشته یک کاراکتر کنترلی \n اضافه می‌کند:

```
println("Hello World!")
```

کد بالا و کد زیر هیچ فرقی با هم ندارند:

```
print("Hello World!\n")
```

متد Print() کارکردی شبیه به Println() دارد با این تفاوت که نشان گر ماوس را در همان خط نگه می‌دارد و خط جدید ایجاد نمی‌کند. جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می‌دهد:

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی از بک اسلش (\\) استفاده می‌کنیم. از آنجاییکه \ معنای خاصی به رشته‌ها می‌دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
println("We can print a \\ by using the \\\\ escape sequence.")
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\، نشان دادن مسیر یک فایل در ویندوز است:

```
println("C:\\Program Files\\Some Directory\\SomeFile.txt")
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (“) برای نشان دادن رشته‌ها استفاده می‌کنیم برای چاپ آن از \” استفاده می‌کنیم:

```
println("I said, \"Motivate yourself!\".")
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (‘) از \’ استفاده می‌کنیم:

```
println("The programmer\'s heaven.")
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می‌شود:

```
println("Left\tRight")
```

```
Left	Right
```

هر تعداد کاراکتر که بعد از کاراکتر کنترلی \r بیایند به اول سطر منتقل و جایگزین کاراکترهای موجود می‌شوند:

```
println("Mitten\rK")
```

```
Kitten
```

مثلاً در مثال بالا کاراکتر K بعد از کاراکتر کنترلی \r آمده است. کاراکتر کنترلی حرف K را به ابتدای سطر برده و جایگزین Mitten می‌کند. برای مشاهده لیست مقادیر مبنای ۱۶ برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید:

http://www.ascii.cl/htmlcodes.htm

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می‌دهد. بیشترین خطا زمانی اتفاق می افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می‌کند.

توضیحات

وقتی که کدی تایپ می‌کنید شاید بخواهید که متنی جهت یادآوری وظیفه آن کد به آن اضافه کنید. در Kotlin (و بیشتر زبانهای برنامه نویسی) می‌توان این کار را با استفاده از توضیحات انجام داد. توضیحات متونی هستند که توسط کامپایلر نادیده گرفته می‌شوند و به عنوان بخشی از کد محسوب نمی‌شوند.

هدف اصلی از ایجاد توضیحات، بالا بردن خوانایی و تشخیص نقش کدهای نوشته شده توسط شما، برای دیگران است. فرض کنید که می‌خواهید در مورد یک کد خاص، توضیح بدهید، می‌توانید توضیحات را در بالای کد یا کنار آن بنویسید. از توضیحات برای مستند سازی برنامه هم استفاده می‌شود. در برنامه زیر نقش توضیحات نشان داده شده است:

```
fun main(args: Array<String>)
{
    // This line will print the message hello world
    println("Hello World!")
```

```
}
```

```
Hello World!
```

در کد بالا، خط ۷ یک توضیح درباره خط ۸ است که به کاربر اعلام می‌کند که وظیفه خط ۸ چیست؟ با اجرای کد بالا فقط جمله Hello World چاپ شده و خط ۷ در خروجی نمایش داده نمی‌شود چون کامپایلر توضیحات را نادیده می‌گیرد. توضیحات بر سه نوعند:

- توضیحات تک خطی

```
// single line comment
```

- توضیحات چند خطی

```
/* multi

line

comment */
```

توضیحات تک خطی همانگونه که از نامش پیداست، برای توضیحاتی در حد یک خط به کار می‌روند. این توضیحات با علامت // شروع می‌شوند و هر نوشته‌ای که در سمت راست آن قرار بگیرد جز توضیحات به حساب می‌آید. این نوع توضیحات معمولاً در بالا یا کنار کد قرار می‌گیرند. اگر توضیح درباره یک کد به بیش از یک خط نیاز باشد از توضیحات چند خطی استفاده می‌شود. توضیحات چند خطی با /* شروع و با */ پایان می‌یابند. هر نوشته‌ای که بین این دو علامت قرار بگیرد جز توضیحات محسوب می‌شود.

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید. می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید.محتویات این ظرف می‌تواند پاک شود یا تغییر کند.هر متغیر دارای یک نام نیز هست.که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد. همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد. مقدار متغیر می‌تواند تهی نیز باشد. متغیر دارای نوع نیز هست بدین

معنی که نوع آن با نوع داده‌ای که در آن ذخیره می‌شود یکی است. متغیر دارای عمر نیز هست که از روی آن می‌توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد. در نهایت متغیر دارای محدوده استفاده نیز هست، که به شما می‌گوید، که متغیر در چه جای برنامه برای شما قابل دسترسی است. ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می‌کنیم. هنگامی که یک برنامه ایجاد می‌کنیم احتیاج به یک مکان برای ذخیره داده، مقادیر یا داده‌هایی که توسط کاربر وارد می‌شوند داریم. این مکان، همان متغیر است. برای این از کلمه متغیر استفاده می‌شود چون ما می‌توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می‌گیرند که برنامه در حال اجراست و وقتی شما برنامه را می‌بندید، محتویات متغیرها نیز پاک می‌شود. قبلاً ذکر شد که به وسیله نام متغیر می‌توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

۱. نام متغیر باید با یکی از حروف الفبا و یا علامت زیر خط (_) شروع شود.
۲. حرف اول نمی تواند عدد باشد.
۳. نمی‌تواند شامل کاراکترهای غیرمجاز مانند #، ?، ^، $ و . باشد.
۴. نمی‌توان از کلمات رزرو شده در کاتلین برای نام متغیر استفاده کرد.
۵. نام متغیر نباید دارای فضای خالی (spaces) باشد.
۶. اگر اسم انتخاب شده برای متغیر چند کلمه ای باشد طبق قرارداد کلمه اول را با حروف کوچک نوشته و کلمات بعدی را با حرف بزرگ شروع کنید. مانند myFirstVariable.
۷. اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در کاتلین دو حرف مانند a و A دو کاراکتر مختلف به حساب می‌آیند.

دو متغیر با نامهای myNumber و MyNumber دو متغیر مختلف محسوب می‌شوند چون یکی از آنها با حرف کوچک mو دیگری با حرف بزرگ Mشروع می‌شود. شما نمی‌توانید دو متغیر را که دقیق شبیه هم هستند را در یک scope (محدوده) تعریف کنید. Scope به معنای یک بلوک کد است که متغیر در آن قابل دسترسی و استفاده است. در مورد Scope در فصلهای آینده بیشتر توضیح خواهیم داد. متغیر دارای نوع هست که نوع داده‌ای را که در خود ذخیره می‌کند را نشان می‌دهد. معمول‌ترین انواع داده Float

، Double، Int، Short، Byte و Long می‌باشند. برای مثال شما برای قرار دادن یک عدد صحیح در متغیر باید از نوع Int استفاده کنید. لیست کلمات کلیدی کاتلین، که نباید از آنها در نامگذاری متغیرها استفاده کرد، در زیر آمده است:

```
as          break       class       continue        do          else
false       for         fun         if              in          interface
is          null        object      package         return          super
thi         throw       true        try             typealias       typeof
val         var         when        while
```

انواع داده

انواع ساده، انواعی از داده‌ها هستند که شامل، اعداد، کاراکترها و مقادیر بولی می‌باشند. به انواع ساده، انواع اصلی نیز گفته می‌شود چون از آنها برای ساخت انواع پیچیده‌تری مانند کلاس‌ها و ... استفاده می‌شود. انواع ساده، دارای مجموعه مشخصی از مقادیر هستند و محدوده خاصی از اعداد را در خود ذخیره می‌کنند. در جدول زیر انواع ساده و محدود آنها آمده است:

نوع	دامنه
Byte	اعداد صحیح بین ۱۲۸- تا ۱۲۷
Short	اعداد صحیح بین ۳۲۷۶۸- تا ۳۲۷۶۷
Int	اعداد صحیح بین ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷
Long	اعداد صحیح بین ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۸- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۸۰۷

جدول زیر انواعی که مقادیر با ممیز اعشار را می‌توانند در خود ذخیره کنند را نشان می‌دهد:

نوع	دامنه تقریبی	دقت
Float	±1.5E-45 to ±3.4E38	۷ رقم
Double	±5.0E-324 to ±1.7E308	۱۶-۱۵ رقم

برای به خاطر سپردن آنها باید از نماد علمی استفاده شود. نوع دیگری از انواع ساده برای ذخیره داده‌های غیر عددی به کار می‌روند و در جدول زیر نمایش داده شده‌اند:

نوع	مقادیر مجاز
Char	کاراکترهای یونیکد
Boolean	مقدار true یا false

نوع Char برای ذخیره کاراکترهای یونیکد استفاده می‌شود. کاراکترها باید داخل یک کوتیشن ساده قرار بگیرند مانند ('a'). نوع Boolean فقط می‌تواند مقادیر درست (true) یا نادرست (false) را در خود ذخیره کند و بیشتر در برنامه‌هایی که دارای ساختار تصمیم گیری هستند مورد استفاده قرار می‌گیرد.

استفاده از رشته‌ها

کاتلین دارای نوعی به نام رشته یا String است. از رشته برای ذخیره گروهی از کاراکترها مانند یک پیغام استفاده می‌شود. مقادیر ذخیره شده در یک رشته باید داخل دابل کوتیشن قرار گیرند تا توسط کامپایلر به عنوان یک رشته در نظر گرفته شوند، مانند ("massage"). دلیل اینکه در این قسمت درباره رشته‌ها مختصری توضیح دادیم این است که قرار است، در آینده به طور مفصل در مورد رشته‌ها توضیح دهیم.

استفاده از متغیرها

در مثال زیر نحوه تعریف و مقدار دهی متغیرها نمایش داده شده است:

```
1  fun main(args: Array<String>)
2  {
3      //Declare variables
4      val num1: Int
5      val num2: Int
```

```
    val num3: Double
    val num4: Double
    val boolVal: Boolean
    val myChar: Char

    //Assign values to variables
    num1 = 1
    num2 = 2
    num3 = 3.54
    num4 = 4.12
    boolVal = true
    myChar = 'R'

    //Show the values of the variables
    println("num1    = $num1")
    println("num3    = $num3")
    println("num4    = $num4")
    println("boolVal = $boolVal")
    println("num2    = $num2")
    println("myChar  = $myChar")
}
```

```
num1    = 1
num3    = 3.54
num4    = 4.12
boolVal = true
num2    = 2
myChar  = R
```

تعریف متغیر

برای تعریف متغیر از دو کلمه var و یا val به صورت زیر استفاده می شود:

```
val identifier
//or
var identifier
```

Identifier نامی است که برای متغیر انتخاب می کنیم. تفاوت بین var و val را در ادامه توضیح می دهیم. به تعریف دو متغیر توجه کنید:

```
val myString = "Hello World!"
val myNumber = 10
```

در کد بالا myString یک متغیر از نوع رشته و myNumber یک متغیر از نوع صحیح یا Int است. ما در کد بالا نوع متغیرها را صراحتا اعلام نکرده ایم و این کار را بر عهده کامپایلر گذاشته ایم. روش دیگر تعریف متغیر این است که صراحتا نوع متغیر را ذکر کنیم:

```
val myString: String = "Hello World!"
val myNumber: Int = 10
```

حال اجازه دهید که کد بالا را توضیح دهیم.

تعریف متغیر

در خطوط ۹-۴ متغیرهایی با نوع و نام متفاوت تعریف شده‌اند. ابتدا باید یکی از کلمات var یا val، سپس نام متغیر و علامت : و در نهایت نوعی که این متغیرها قرار است در خود ذخیره کنند را مشخص کنیم . همیشه به یاد داشته باشید که قبل از مقدار دهی و استفاده از متغیر باید آن را تعریف کرد.

```
val num1: Int
val num2: Int
val num3: Double
val num4: Double
val boolVal: Boolean
val myChar: Char
```

نحوه تعریف متغیر به صورت زیر است:

```
var/val identifier: data_type
```

date_type همان نوع داده است مانند Int، Double و ..... identifier نیز نام متغیر است که به ما امکان استفاده و دسترسی به مقدار متغیر را می‌دهد.

نامگذاری متغیرها

- نام متغیر باید با یک حرف یا زیرخط و به دنبال آن حرف یا عدد شروع شود.

- نمی‌توان از کاراکترهای خاص مانند #، %، & یا عدد برای شروع نام متغیر استفاده کرد مانند 2numbers.
- نام متغیر نباید دارای فاصله باشد. برای نام‌های چند حرفی می‌توان به جای فاصله از علامت زیرخط یا _ استفاده کرد.

نامهای مجاز:

```
num1  myNumber  studentCount  total       first_name    _minimum
num2  myChar    average       amountDue   last_name     _maximum
name  counter   sum           isLeapYear  color_of_car  _age
```

نامهای غیر مجاز:

```
123         #numbers#  #ofstudents  1abc2
123abc      $money     first name   ty.np
my number   this&that  last name    1:00
```

اگر به نامهای مجاز در مثال بالا توجه کنید متوجه قراردادهای به کار رفته در نامگذاری آن‌ها خواهید شد. یکی از روشهای نامگذاری، نامگذاری کوهان شتری است. در این روش که برای متغیرهای دو کلمه‌ای به کار می‌رود، اولین حرف اولین کلمه با حرف کوچک و اولین حرف دومین کلمه با حرف بزرگ نمایش داده می‌شود مانند : myNumber توجه کنید که اولین حرف کلمه Number با حرف بزرگ شروع شده است. مثال دیگر کلمه numberOfStudents است. اگر توجه کنید، بعد از اولین کلمه، حرف اول سایر کلمات با حروف بزرگ نمایش داده شده است.

محدوده متغیر

متغیرهای ابتدای درس در داخل متد main() تعریف شده‌اند. در نتیجه، این متغیرها فقط در داخل متد main() قابل دسترسی و استفاده هستند. محدوده یک متغیر مشخص می‌کند که متغیر در کجای کد قابل دسترسی است. هنگامیکه برنامه به پایان متد main() می‌رسد متغیرها از محدوده خارج و بدون استفاده می‌شوند تا زمانی که برنامه در حال اجراست. محدوده متغیرها انواعی دارد که در درسهای بعدی با آنها آشنا می‌شوید. تشخیص محدوده متغیر بسیار مهم است چون به وسیله آن می‌فهمید که در کجای کد می‌توان از متغیر استفاده کرد. باید یاد آور شد که دو متغیر در یک محدوده نمی‌توانند دارای نام یکسان باشند. مثلاً کد زیر در برنامه ایجاد خطا می‌کند:

```
val num1: Int
val num1: Int
```

از آنجاییکه کاتلین به بزرگی و کوچک بودن حروف حساس است، می‌توان از این خاصیت برای تعریف چند متغیر هم نام ولی با حروف متفاوت (از لحاظ بزرگی و کوچکی) برای تعریف چند متغیر از یک نوع استفاده کرد مانند:

```
val num1: Int
val Num1: Int
val NUM1: Int
```

مقداردهی متغیرها

می‌توان فوراً بعد از تعریف متغیرها مقادیری را به آنها اختصاص داد. این عمل را مقداردهی می‌نامند. در زیر نحوه مقداردهی متغیرها نشان داده شده است:

```
var/val identifier: data_type = value
```

به عنوان مثال:

```
val myNumber: Int = 7;
```

تعریف متغیر با مقدار دهی متغیرها متفاوت است. تعریف متغیر یعنی انخاب نوع و نام برای متغیر ولی مقدار دهی یعنی اختصاص یک مقدار به متغیر.

اختصاص مقدار به متغیر

در زیر نحوه اختصاص مقادیر به متغیرها نشان داده شده است:

```
num1 = 1
num2 = 2
num3 = 3.54
num4 = 4.12
boolVal = true
myChar = 'R'
```

به این نکته توجه کنید که شما به متغیری که هنوز تعریف نشده، نمی‌توانید مقدار بدهید. شما فقط می‌توانید از متغیرهایی استفاده کنید که هم تعریف و هم مقدار دهی شده باشند. مثلاً متغیرهای بالا همه قابل استفاده هستند. در این مثال num1 و num2 هر دو تعریف شده‌اند و مقادیری از نوع صحیح به آنها اختصاص داده شده است. اگر نوع داده با نوع متغیر یکی نباشد برنامه پیغام خطا می‌دهد. به خطوط ۲۵-۲۰ می باشد. وجود علامت $ قبل از نام متغیر در اصل به منزله مقداری است که متغیر در خود ذخیره کرده است . مثلا $num1 در خط ۲۰، یعنی مقدار این متغیر که همان عدد ۱ در ۱۲ است. حال همین علامت $ در خطوط ۲۵-۲۰ را حذف کرده و برنامه را اجرا و نتیجه را مشاهده نمایید. البته از این علامت زمانی استفاده می شود که شما بخواهید در داخل رشته یعنی علامت "" مقدار متغیر را چاپ کنید:

```
println("$num1")
```

ولی اگر بخواهید خارج از علامت "" مقدار یک متغیر را چاپ کنید، دیگر احتیاجی به علامت $ نیست:

```
println(num1)
```

تفاوت var و val

تفاوت این دو کلمه در این است که اگر یک متغیر را با کلمه val تعریف و مقدار دهی کنیم، دیگر نمی توانیم در طول برنامه مقدار متغیر را تغییر دهیم و اگر این کار را انجام دهیم برنامه با خطا مواجه می شود:

```
val number = 10
    number = 5
```

ولی اگر از کلمه var استفاده کنیم، می توانیم بعدا مقدار متغیر را تغییر دهیم.

```
var number = 10
    number = 5
```

تبدیل انواع داده

توصیه می کنیم قبل از شروع این درس ابتدا دو مطلب زیر را بخوانید:

http://www.w3-farsi.com/?p=5698

http://www.w3-farsi.com/?p=5710

در کاتلین، انواع عددی نمی توانند به طور خودکار یا ضمنی به هم تبدیل شوند. مثلا در جاوا کد زیر قابل قبول است:

```
byte number1 = 5;

int number2 = number1;
```

در کد بالا نوع byte به نوع int تبدیل و این کار به صورت خودکار توسط کامپایلر انجام می شود. حال اگر همین کد را در کاتلین پیاده سازی کنیم با خطا مواجه می شویم:

```
val number1: Byte = 5

val number2: Int = number1
```

برای تبدیل یک نوع به نوع دیگر باید صریحا نوع داده ای که قرار است تبدیل شود را مشخص کنیم و این کار با استفاده از توابعی در کاتلین امکان پذیر است که در جدول زیر به آنها اشاره شده است:

تابع	کاربرد
toByte()	مقدار یک متغیر را به نوع Byte تبدیل می کند.
toChar()	مقدار یک متغیر را به نوع Char تبدیل می کند.
toFloat()	مقدار یک متغیر را به نوع Float تبدیل می کند.
toInt()	مقدار یک متغیر را به نوع Int تبدیل می کند.
toLong()	مقدار یک متغیر را به نوع Long تبدیل می کند.
toShort()	مقدار یک متغیر را به نوع Short تبدیل می کند.
toString()	مقدار یک متغیر را به نوع String تبدیل می کند.

با وجود این توابع تبدیل انواع داده به هم امکان پذیر است. در نتیجه کد کاتلین بالا به صورت زیر قابل انجام است:

```
fun main(args: Array<String>)
{
    val number1: Byte = 5
    val number2: Int = number1.toInt()

    print(number2)
}
```

```
5
```

در خط ۲ کد بالا بعد از نام متغیر اول، علامت نقطه و نام تابع را ذکر کرده ایم. این بدین معناست که متغیر number1 را به نوع Int تبدیل کن و در داخل متغیر number2 قرار بده. در مثال نوع داده‌ای Byte می‌تواند مقادیر ۰ تا ۲۵۵ را در خود ذخیره کند و نوع داده‌ای int مقادیر ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷ را شامل می‌شود. پس می‌توانید بدون هیچ مشکلی یک متغیر از نوع Byte را به یک نوع Int تبدیل کنید. مقدار ۵ متغیر number1 در محدوده مقادیر Byte یعنی اعداد بین ۲۵۵-۰ قرار دارد اما متغیری از نوع بایت حافظه کمتری نسبت به متغیری از نوع صحیح اشغال می‌کند. نوع Byte شامل ۸ بیت یا ۸ رقم دودویی است در حالی که نوع Int شامل ۳۲ بیت یا رقم باینری است. یک عدد باینری عددی متشکل از ۰ و ۱ است. برای مثال عدد ۵ در کامپیوتر به عدد باینری ۱۰۱ ترجمه می‌شود. بنابراین وقتی ما عدد ۵ را در یک متغیر از نوع بایت ذخیره می‌کنیم عددی به صورت زیر نمایش داده می‌شود:

```
00000101
```

و وقتی آن را در یک متغیر از نوع صحیح ذخیره می‌کنیم به صورت زیر نمایش داده می‌شود:

```
00000000000000000000000000000101
```

بنابراین قرار دادن یک مقدار Int، در یک متغیر Byte، درست مانند این است که ما سعی کنیم که یک توپ فوتبال را در یک سوراخ کوچک گلف جای دهیم. حال به مثال زیر توجه کنید:

```
fun main(args: Array<String>)
{
    val number1: Int = 300
    val number2: Byte = number1.toByte()

    print(number2)
}
```

```
44
```

در کد بالا یک number1 یک نوع صحیح با مقدار ۳۰۰ است که آن را با استفاده از تابع ()toByte به نوع Byte تبدیل کرده و در متغیر number2 قرار داده ایم. پس چرا خروجی ۴۴ است؟ دلیل آن این است که نوع Byte اعداد بین ۲۵۵-۰ را در خود ذخیره می کند و عدد ۳۰۰ جز این محدود نیست. اما چرا عدد ۴۴؟ این کار به تعداد بیتها بستگی دارد. یک Byte دارای ۸ بیت است درحالی که Int دارای ۳۲ بیت است. حال اگر به مقدار باینری ۲ عدد توجه کنید متوجه می‌شوید که چرا خروجی عدد ۴۴ است.

```
300 = 00000000000000000000000100101100
255 =                         11111111
-------------------------------------
 44 =                         00101100
```

خروجی بالا نشان می‌دهد که بیشترین مقدار byte که عدد ۲۵۵ است می‌تواند فقط شامل ۸ بیت باشد (۱۱۱۱۱۱۱۱) بنابراین فقط ۸ بیت اول مقدار Int به متغیر Byte انتقال می‌یابد که شامل (۰۰۱۰۱۱۰۰) یا عدد ۴۴ در مبنای ۱۰ است.

عبارات و عملگرها

ابتدا با دو کلمه آشنا شوید:

- عملگر: نمادهایی هستند که اعمال خاص انجام می‌دهند.
- عملوند: مقادیری که عملگرها بر روی آنها عملی انجام می‌دهند.

مثلاً X+Y: یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می‌آیند.

زبانهای برنامه نویسی جدید دارای عملگرهایی هستند که از اجزاء معمول زبان به حساب می‌آیند. کاتلین دارای عملگرهای مختلفی از جمله عملگرهای ریاضی، تخصیصی، مقایسه‌ای، منطقی و بیتی می‌باشد. از عملگرهای ساده ریاضی می‌توان به عملگر جمع و تفریق اشاره کرد. دو نوع عملگر در کاتلین وجود دارد :

- یگانی - (Unary) به یک عملوند نیاز دارد
- دودویی - (Binary) به دو عملوند نیاز دارد

انواع مختلف عملگر که در این بخش مورد بحث قرار می‌گیرند عبارتند از:

- عملگرهای ریاضی
- عملگرهای تخصیصی
- عملگرهای مقایسه‌ای
- عملگرهای منطقی
- عملگرهای بیتی
- عملگر `in`
- عملگر دسترسی به اندیس
- عملگر `invoke` (درخواست)

عملگرهای ریاضی

کاتلین از عملگرهای ریاضی برای انجام محاسبات استفاده می‌کند. جدول زیر عملگرهای ریاضی کاتلین را نشان می‌دهد:

عملگر	مثال	نتیجه
+	var1 = var2 + var3	Var1 برابر است با حاصل جمع var2 و var3
-	var1 = var2 - var3	Var1 برابر است با حاصل تفریق var2 و var3
*	var1 = var2 * var3	Var1 برابر است با حاصلضرب var2 در var3
/	var1 = var2 / var3	Var1 برابر است با حاصل تقسیم var2 بر var3
%	var1 = var2 % var3	Var1 برابر است با باقیمانده تقسیم var2 و var3

حال می‌توانیم با ایجاد یک برنامه نحوه عملکرد عملگرهای ریاضی در کاتلین را یاد بگیریم:

```
1  fun main(args: Array<String>)
2  {
3      //Variable declarations
```

```
4       val num1: Int
5       val num2: Int
6
7       //Assign test values
8       num1 = 5
9       num2 = 3
10
11      println("The sum of $num1 and $num2 is "        + (num1 + num2))
12      println("The difference of $num1 and $num2 is " + (num1 - num2))
13      println("The product of $num1 and $num2 is "    + (num1 * num2))
14      println("The quotient of $num1 and $num2 is "   + (num1 / num2))
15      println("The remainder of $num1 and $num2 is "  + (num1 % num2))
16  }
```

```
The sum of 5 and 3 is 8
The difference of 5 and 3 is 2
The product of 5 and 3 is 15
The quotient of 5 and 3 is 1
The remainder of 5 divided by 3 is 2
```

برنامه بالا نتیجه هر عبارت را نشان می‌دهد. در این برنامه از متد println() برای نشان دادن نتایج در سطرهای متفاوت استفاده شده است. خطوط ۱۵-۱۱ کد بالا را به صورت زیر هم می توان نوشت:

```
println("The sum of $num1 and $num2 is ${num1 + num2}")
println("The difference of $num1 and $num2 is ${num1 - num2}")
println("The product of $num1 and $num2 is ${num1 * num2}")
println("The quotient of $num1 and $num2 is ${num1 / num2}")
println("The remainder of $num1 and $num2 is ${num1 % num2}")
```

دیگر عملگرهای کاتلین عملگرهای کاهش و افزایش هستند. این عملگرها مقدار ۱ را از متغیرها کم یا به آنها اضافه می‌کنند. از این متغیرها اغلب در حلقه‌ها استفاده می‌شود:

عملگر	مثال	نتیجه
++	var1 = ++var2	مقدار var1 برابر است با var2 بعلاوه ۱
--	var1 = --var2	مقدار var1 برابر است با var2 منهای ۱
++	var1 = var2++	مقدار var1 برابر است با var2 به متغیر var2 یک واحد اضافه می‌شود
--	var1 = var2--	مقدار var1 برابر است با var2 از متغیر var2 یک واحد کم می‌شود

به این نکته توجه داشته باشید که، محل قرار گیری عملگر در نتیجه محاسبات تأثیر دارد. اگر عملگر قبل از متغیر var2 بیاید، افزایش یا کاهش var1 اتفاق می افتد. چنانچه عملگرها بعد از متغیر var2 قرار بگیرند، ابتدا var1 برابر var2 می‌شود و سپس متغیر var2 افزایش یا کاهش می‌یابد. به مثال‌های زیر توجه کنید:

```
fun main(args: Array<String>)
{
    var x = 0
    var y = 1

    x = ++y

    println("x = $x")
    println("y = $y")
}
```

```
X = 2
Y = 2
```

```
fun main(args: Array<String>)
{
    var x = 0
    var y = 1

    x = --y

    println("x = $x")
    println("y = $y")
}
```

```
X = 0
Y = 0
```

همانطور که در دو مثال بالا مشاهده می‌کنید، درج عملگرهای -- و ++ قبل از عملوند y باعث می‌شود که ابتدا یک واحد از y کم و یا یک واحد به y اضافه شود و سپس نتیجه در عملوند x قرار بگیرد. حال به دو مثال زیر توجه کنید:

```
fun main(args: Array<String>)
{
    var x = 0
    var y = 1

    x = y--

    println("x = $x")
    println("y = $y")
}
```

```
X = ۱
Y = 0
```

```
fun main(args: Array<String>)
{
    var x = 0
    var y = 1

    x = y++

    println("x = $x")
    println("y = $y")
}
```

```
X = ۱
Y = ۲
```

همانطور که در دو مثال بالا مشاهده می‌کنید، درج عملگرهای -- و ++ بعد از عملوند y باعث می‌شود که ابتدا مقدار y در داخل متغیر x قرار بگیرد و سپس یک واحد از y کم و یا یک واحد به ان اضافه شود.

عملگرهای تخصیصی

نوع دیگر از عملگرهای کاتلین عملگرهای جایگزینی نام دارند. این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می‌دهند. جدول زیر انواع عملگرهای تخصیصی در کاتلین را نشان می‌دهد:

عملگر	مثال	نتیجه
=	var1 = var2	مقدار var1 برابر است با مقدارvar2
+=	var1 += var2	مقدار var1 برابر است با حاصل جمع var1 و var2
-=	var1 -= var2	مقدار var1 برابر است با حاصل تفریق var1 و var2
*=	var1 *= var2	مقدار var1 برابر است با حاصل ضرب var1 در var2
/=	var1 /= var2	مقدار var1 برابر است با حاصل تقسیم var1 بر var2
%=	var1 %= var2	مقدار var1 برابر است با باقیمانده تقسیم var1 بر var2

از عملگر =+ برای اتصال دو رشته نیز می‌توان استفاده کرد. استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلاً شکل اصلی کد var1 += var2 به صورت var1 = var1 + var2 می‌باشد. این حالت کدنویسی زمانی کارایی خود را نشان می‌دهد که نام متغیرها طولانی باشد. برنامه زیر چگونگی استفاده از عملگرهای تخصیصی و تأثیر آنها را بر متغیرها نشان می‌دهد.

```
fun main(args: Array<String>)
{
    var number: Int

    println("Assigning 10 to number...")
    number = 10
    println("Number = $number")

    println("Adding 10 to number...")
    number += 10
    println("Number = $number")

    println("Subtracting 10 to number...")
    number -= 10
    println("Number = $number")
}
```

```
Assigning 10 to number...
Number = 10
Adding 10 to number...
Number = 20
Subtracting 10 from number...
Number = 10
```

در برنامه از ۳ عملگر تخصیصی استفاده شده است. ابتدا یک متغیر و مقدار ۱۰ با استفاده از عملگر = به آن اختصاص داده شده است. سپس به آن با استفاده از عملگر =+ مقدار ۱۰ اضافه شده است. و در آخر به وسیله عملگر =- عدد ۱۰ از آن کم شده است.

عملگرهای مقایسه ای

از عملگرهای مقایسه‌ای برای مقایسه مقادیر استفاده می‌شود . نتیجه این مقادیر یک مقدار بولی (منطقی) است. این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد، مقدار `true` و اگر نتیجه مقایسه اشتباه باشد، مقدار `false` را نشان می‌دهند. این عملگرها به طور معمول در دستورات شرطی به کار می‌روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می‌شوند. جدول زیر عملگرهای مقایسه‌ای در کاتلین را نشان می‌دهد:

عملگر	مثال	نتیجه
==	`var1 = var2 == var3`	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر باشد در غیر اینصورت false است.
!=	`var1 = var2 != var3`	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت false است.
<	`var1 = var2 < var3`	var1 در صورتی true است که مقدار var2 کوچک‌تر از var3 مقدار باشد در غیر اینصورت false است.
>	`var1 = var2 > var3`	var1 در صورتی true است که مقدار var2 بزرگ‌تر از مقدار var3 باشد در غیر اینصورت false است.
<=	`var1 = var2 <= var3`	var1 در صورتی true است که مقدار var2 کوچک‌تر یا مساوی مقدار var3 باشد در غیر اینصورت false است.
>=	`var1 = var2 >= var3`	var1 در صورتی true است که مقدار var2 بزرگ‌تر یا مساوی var3 مقدار باشد در غیر اینصورت false است.

برنامه زیر نحوه عملکرد این عملگرها را نشان می‌دهد:

```
fun main(args: Array<String>)
{
    val num1 = 10
    val num2 = 5

    println("$num1 == $num2 : " + (num1 == num2))
    println("$num1 != $num2 : " + (num1 != num2))
    println("$num1  < $num2 : " + (num1 <  num2))
    println("$num1  > $num2 : " + (num1 >  num2))
    println("$num1 <= $num2 : " + (num1 <= num2))
    println("$num1 >= $num2 : " + (num1 >= num2))
}
```

```
10 == 5 : false
10 != 5 : true
10  < 5 : false
10  > 5 : true
10 <= 5 : false
10 >= 5 : true
```

در مثال بالا ابتدا دو متغیر را که می‌خواهیم با هم مقایسه کنیم را ایجاد کرده و به آنها مقادیری اختصاص می‌دهیم. سپس با استفاده از یک عملگر مقایسه‌ای آنها را با هم مقایسه کرده و نتیجه را چاپ می‌کنیم.

به این نکته توجه کنید که هنگام مقایسه دو متغیر از عملگر == به جای عملگر = باید استفاده شود.

عملگر = عملگر تخصیصی است و در عبارتی مانند `x = y` مقدار `y` را به `x` اختصاص می‌دهد. عملگر == عملگر مقایسه‌ای است که دو مقدار را با هم مقایسه می‌کند مانند `x == y` و اینطور خوانده می‌شود `x` برابر است با `y`.

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می‌دهند که شکل باینری انواع داده‌ها را دستکاری کنید. برای درک بهتر این درس توصیه می‌شود که شما سیستم باینری و نحوه تبدیل اعداد دهدهی به باینری را یاد بگیرید. در سیستم باینری (دودویی) که کامپیوتر از آن استفاده می‌کند وضعیت هر چیز یا خاموش است یا روشن. برای نشان دادن حالت روشن از عدد ۱ و برای نشان دادن حالت خاموش از عدد ۰ استفاده می‌شود. بنابراین اعداد باینری فقط می‌توانند صفر یا یک باشند. اعداد باینری را اعداد در مبنای ۲ و اعداد اعشاری را اعداد در مبنای ۱۰ می گویند. یک بیت نشان دهنده یک رقم باینری است و هر بایت نشان دهنده ۸ بیت است. به عنوان مثال برای

یک داده از نوع Int به ۳۲ بیت یا ۸ بایت فضا برای ذخیره آن نیاز داریم، این بدین معناست که اعداد از ۳۲ رقم ۰ و ۱ برای ذخیره استفاده می‌کنند. برای مثال عدد ۱۰۰ وقتی به عنوان یک متغیر از نوع Int ذخیره می‌شود در کامپیوتر به صورت زیر خوانده می‌شود :

```
00000000000000000000000001100100
```

عدد ۱۰۰ در مبنای ده معادل عدد ۱۱۰۰۱۰۰ در مبنای ۲ است. در اینجا ۷ رقم سمت راست نشان دهنده عدد ۱۰۰ در مبنای ۲ است و مابقی صفرهای سمت راست برای پر کردن بیتهایی است که عدد از نوع Int نیاز دارد. به این نکته توجه کنید که اعداد باینری از سمت راست به چپ خوانده می‌شوند. عملگرهای بیتی کوتلین در جدول زیر نشان داده شده‌اند:

عملگر	مثال
and	x = y and z
or	x = y or z
xor	x = y xor z
inv	x = y.inv()

عملگر بیتی and

عملگر بیتی and کاری شبیه عملگر منطقی and انجام می‌دهد با این تفاوت که این عملگر بر روی بیتها کار می‌کند. اگر مقادیر دو طرف آن ۱ باشد مقدار ۱ را بر می‌گرداند و اگر یکی یا هر دو طرف آن صفر باشد مقدار صفر را بر می‌گرداند. جدول درستی عمگر بیتی and در زیر آمده است:

X and Y	Y	X
1	1	1
0	0	1
0	1	0
0	0	0

در زیر نحوه استفاده از عملگر بیتی and آمده است:

```
val result = 5 and 3

println(result)
```

```
1
```

همانطور که در مثال بالا مشاهده می‌کنید نتیجه عملکرد عملگر and بر روی دو مقدار ۵ و ۳ عدد یک می‌شود. اجازه بدهید ببینیم که چطور این نتیجه را به دست می‌آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
---------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد ۵ و ۳ به معادل باینری‌شان تبدیل می‌شوند. از آنجاییکه هر عدد صحیح(Int) ۳۲ بیت است از صفر برای پر کردن بیتهای خالی استفاده می‌کنیم. با استفاده از جدول درستی عملگر بیتی and می‌توان فهمید که چرا نتیجه عدد یک می‌شود.

عملگر بیتی or

اگر مقادیر دو طرف عملگر بیتی or هر دو صفر باشند نتیجه صفر در غیر اینصورت ۱ خواهد شد. جدول درستی این عملگر در زیر آمده است:

X	Y	X or Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی or در صورتی صفر است که عملوندهای دو طرف آن صفر باشند. اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد. به مثال زیر توجه کنید:

```
val result = 7 or 9
```

```
println(result)
```

```
15
```

وقتی که از عملگر بیتی or برای دو مقدار در مثال بالا (۷ و ۹) استفاده می‌کنیم نتیجه ۱۵ می‌شود. حال بررسی می‌کنیم که چرا این نتیجه به دست آمده است؟

```
7: 00000000000000000000000000000111
 9: 00000000000000000000000000001001
---------------------------------
15: 00000000000000000000000000001111
```

با استفاده از جدول درستی عملگر بیتی or می‌توان نتیجه استفاده از این عملگر را تشخیص داد. عدد ۱۱۱۱ باینری معادل عدد ۱۵ صحیح است.

عملگر بیتی xor

جدول درستی این عملگر در زیر آمده است:

X	Y	X xor Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند نتیجه صفر در غیر اینصورت نتیجه یک می‌شود. در مثال زیر تأثیر عملگر بیتی xor را بر روی دو مقدار مشاهده می‌کنید:

```
val result = 5 xor 7

println(result)
```

```
2
```

در زیر معادل باینری اعداد بالا (۵ و ۷) نشان داده شده است.

```
5: 00000000000000000000000000000101
7: 00000000000000000000000000000111
---------------------------------
2: 00000000000000000000000000000010
```

با نگاه کردن به جدول درستی عملگر بیتی xor، می‌توان فهمید که چرا نتیجه عدد ۲ می‌شود.

عملگر بیتی inv

این عملگر یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. در زیر جدول درستی این عملگر آمده است:

X	inv()
1	0
0	1

عملگر بیتی inv مقادیر بیتها را معکوس می‌کند. در زیر چگونگی استفاده از این عملگر آمده است:

```
val result = 7.inv()

println(result)
```

```
-8
```

به نمایش باینری مثال بالا که در زیر نشان داده شده است توجه نمایید.

```
 7: 00000000000000000000000000000111
-----------------------------------
-8: 11111111111111111111111111111000
```

عملگر بیتی تغییر مکان

این نوع عملگرها به شما اجازه می‌دهند که بیتها را به سمت چپ یا راست جا به جا کنید. دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می‌کنند. عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جایی بیت ها را نشان می‌دهد.

عملگر	نام	مثال
shr	تغییر مکان به سمت چپ	x = y shl 2
shl	تغییر مکان به سمت راست	x = y shr 2

عملگر تغییر مکان به سمت چپ

این عملگر، بیتهای عملوند سمت چپ را به تعداد n مکان مشخص شده، توسط عملوند سمت راست، به سمت چپ منتقل می‌کند.

به عنوان مثال:

```
val result = 10 shl 2

println(result)
```

```
40
```

در مثال بالا، ما بیتهای مقدار ۱۰ را دو مکان به سمت چپ منتقل کرده‌ایم، حال بیایید تأثیر این انتقال را بررسی کنیم:

```
10: 00000000000000000000000000001010
----------------------------------
40: 00000000000000000000000000101000
```

مشاهده می‌کنید که همه بیت‌ها به اندازه دو واحد به سمت چپ منتقل شده‌اند. در این انتقال دو صفر از صفرهای سمت چپ کم می‌شود و در عوض دو صفر به سمت راست اضافه می‌شود.

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت‌ها را به سمت راست جا به جا می‌کند. به عنوان مثال:

```
val result = 100 shr 4

println(result)
```

```
6
```

با استفاده از عملگرتغییر مکان به سمت راست بیت‌های مقدار ۱۰۰ را به اندازه ۴ واحد به سمت چپ جا به جا می‌کنیم. اجازه بدهید تأثیر این جا به جایی را مورد بررسی قرار دهیم:

```
100: 00000000000000000000000001100100
-----------------------------------
  6: 00000000000000000000000000000110
```

هر بیت به اندازه ۴ واحد به سمت راست منتقل می‌شود، بنابراین ۴ بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می‌شود.

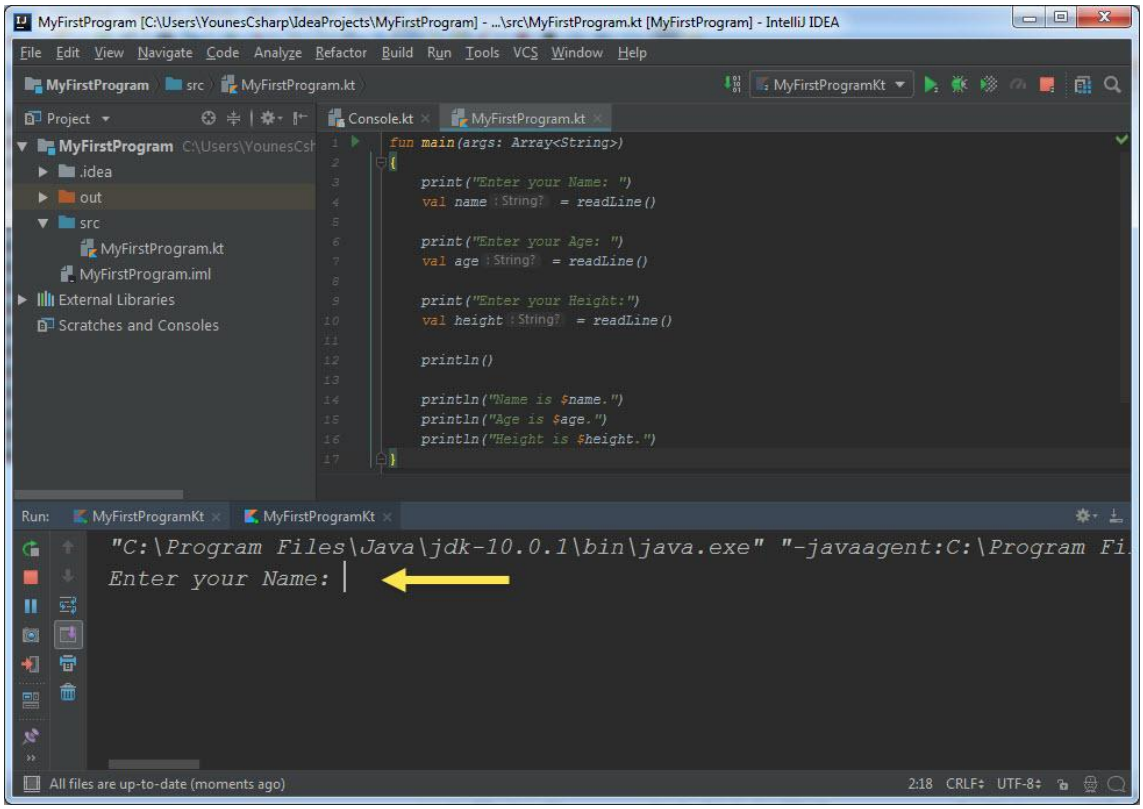
گرفتن ورودی از کاربر

کاتلین از متد `readLine()` برای گرفتن ورودی از کاربر استفاده می کند. این متد، تمام کاراکترهایی را که شما در محیط کنسول تایپ می‌کنید تا زمانی که دکمه `Enter` را می‌زنید، می‌خواند. به برنامه زیر توجه کنید:

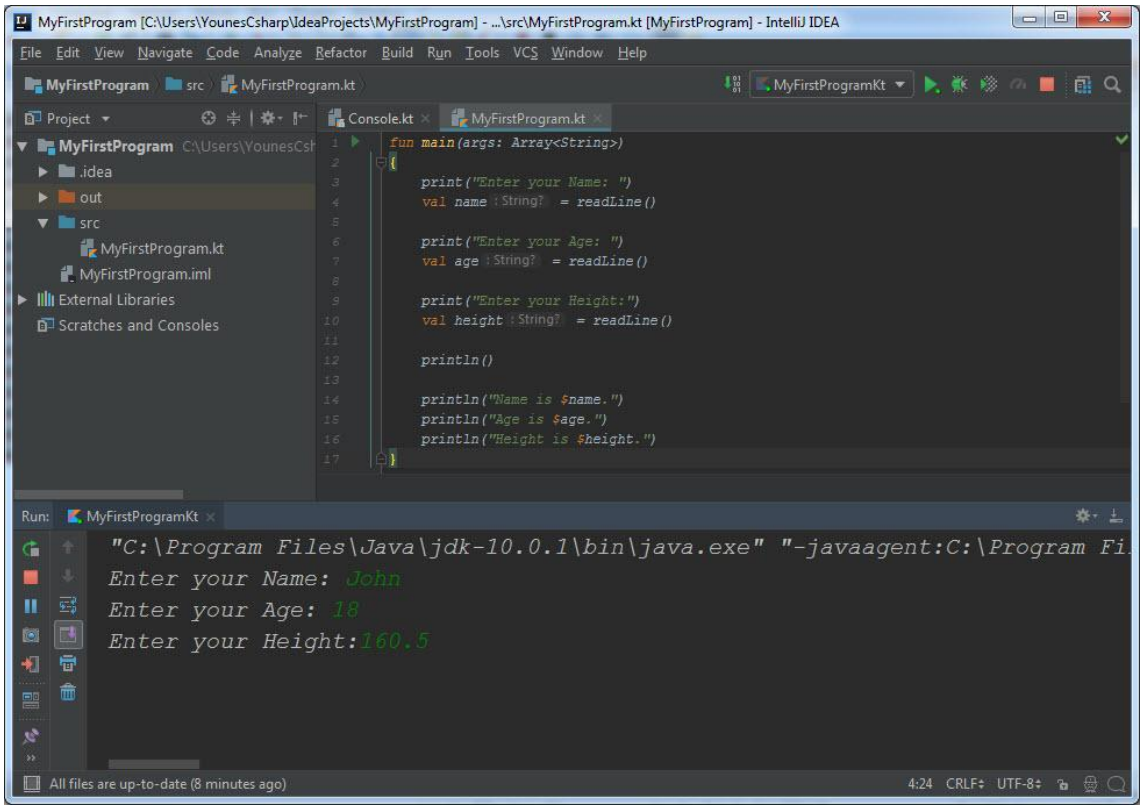
```
1  fun main(args: Array<String>)
2  {
3      print("Enter your Name: ")
4      val name = readLine()
5
6      print("Enter your Age: ")
7      val age = readLine()
8
9      print("Enter your Height: ")
10     val height = readLine()
11
12     println()
13
```

```
14      println("Name is $name.")
15      println("Age is $age.")
16      println("Height is $height.")
17  }
```

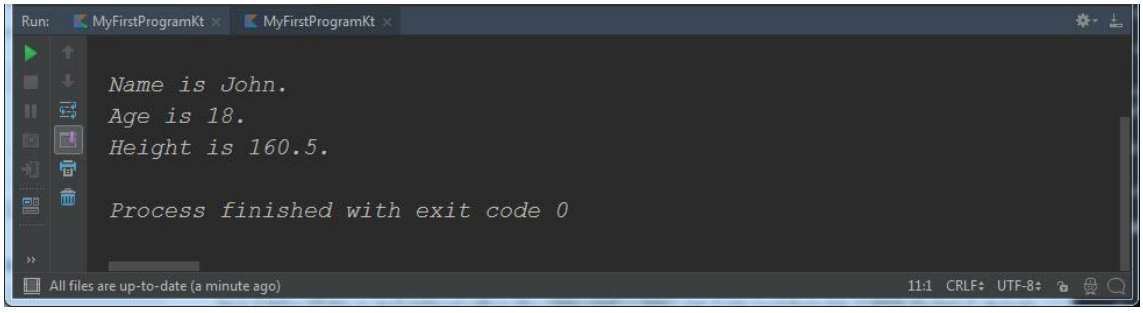
با اجرای برنامه و کلیک بر روی قسمت پایین برنامه Intellj IDEA، که به آن محیط کنسول هم گفته می شود، نشان گر ماوس نمایش داده می شود و برنامه از شما می خواهد که اطلاعات را وارد کنید:



برنامه از شما نام، سن و قدتان را می خواهد و شما باید بعد از وارد کردن اطلاعات هر بخش، دکمه Enter را بزنید:



با زدن دکمه Enter آخر، خروجی به صورت زیر نمایش داده می شود:



در همین حد کافیست که بدانید که متد readLine() ورودی را از کاربر گرفته و آنها را به صورت رشته در متغیرهای age، name و height (خطوط ۴، ۷ و ۱۰) قرار می دهد. قبول دارید که سن و قد باید از نوع عددی باشند؟ اگر بخواهید فقط در محیط کنسول مقادیری را نمایش دهید که عددی یا غیر عددی بود مشکلی به وجود نمی آورد. مانند مثال بالا، که ما فقط از کاربر اطلاعاتی را گرفته و چاپ کرده ایم. ولی اگر بخواهیم دو عدد را با هم جمع کنیم، باید چکار کنیم. به کد زیر توجه کنید:

```
fun main(args: Array<String>)
```

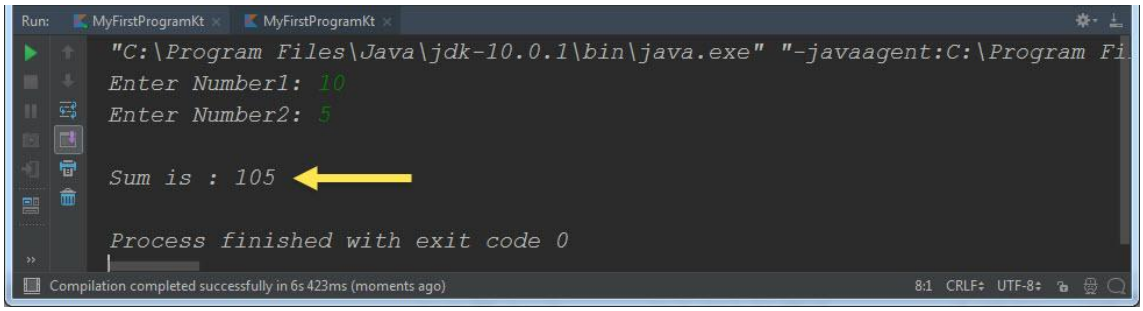
```
{
    print("Enter Number1: ")
    val number1 = readLine()

    print("Enter Number2: ")
    val number2 = readLine()

    println()

    println("Sum is : " + (number1 + number2))
}
```

کد بالا را اجرا کرده و دو عدد را وارد نمایید:



همانطور که مشاهده می کنید بعد از وارد کردن اعداد ۱۰ و ۵ خروجی عدد ۱۰۵ می شود. البته در اصل عدد ۱۰۵ نیست، بلکه برنامه دو عدد را مانند دو رشته در نظر می گیرد و آنها را به هم می چسباند. اگر بخواهیم این دو عدد را واقعا با هم جمع کنیم و عدد ۱۵ را به دست بیاوریم باید بعد زا متد `readLine()` از متدهای تبدیل داده که در درس های قبل ذکر کردیم به صورت زیر استفاده نمایید:

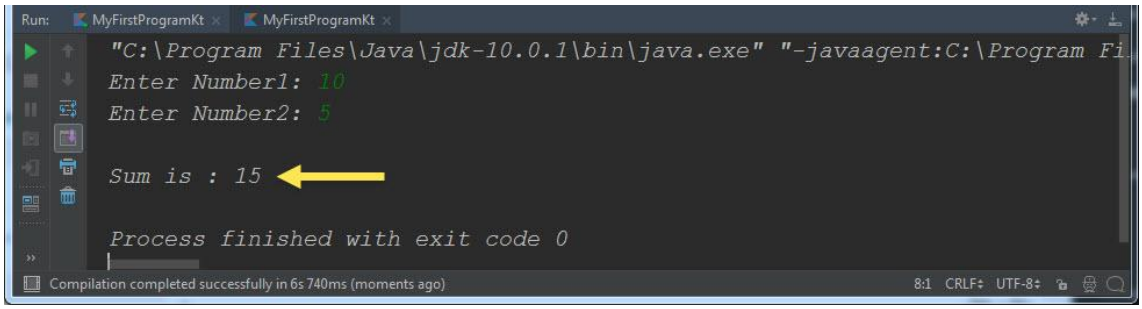
```
fun main(args: Array<String>)
{
    print("Enter Number1: ")
    val number1 = readLine()!!.toInt()

    print("Enter Number2: ")
    val number2 = readLine()!!.toInt()

    println()

    println("Sum is : " + (number1 + number2))
}
```

به خروجی کد بالا توجه کنید:



همانطور که گفتیم، متد `readLine()` رشته در یافت می کند، برای تبدیل این رشته به عدد صحیح از متد `toInt()` در استفاده کرده ایم. اینکه چرا قبل از متد `toInt()` علامت !! را قرار داده ایم در درس های آینده توضیح می دهیم. در نتیجه متد `toInt()` رشته گرفته شده توسط متد `readLine()` را به نوع صحیح تبدیل کرده و در متغیر های `number1` و `number2` می گذارد. با این وجود می توانیم دو متغیر را که عددی هستند با هم جمع کنیم.

برای مشاهده سایر مطالب کتاب کاتلین به زبان ساده به لینک زیر مراجعه فرمایید:

www.w3-farsi.com

راه‌های ارتباط با نویسنده

وب سایت: www.w3-farsi.com

لینک تلگرام: https://telegram.me/ebrahimi_younes

ID تلگرام: @ebrahimi_younes

پست الکترونیکی: younes.ebrahimi.1391@gmail.com