آموزش کاتلین

آموزش کاتلین

کاتلین یک زبان برنامه نویسی متن باز مثل جاوا ، جاوا اسکریپت و غیره است. یک زبان نگارشی سطح بالا و به شدت آماری است که بخش های تکنیکی و عملی را در یک مکان ارائه می دهد. در حال حاضر هدف کاتلین، جاوا و جاوا اسکریپت است. کاتلین روی JVM اجرا می شود. با آموزش زبان برنامه نویسی کاتلین همراه ما باشید.

کاتلین از سایر زبان های برنامه نویسی مثل جاوا، , Scala, Groovy, Gosu و غیره تاثیر پذیرفته است. نحوه ی نوشتن کاتلین ممکن است دقیقا مثل جاوا نباشد، ولی، کاتلین از درون به کتابخانه ی رده ی جاوا متکی است تا نتایج خارق العاده ای را برای برنامه نویس به ارمغان بیاورد. کاتلین قابلیت انتقال اطلاعات، امنیت کدنویسی، و شفافیت را برای توسعه دهنده ها بوجود می آورد.

نکات مثبت و منفی

در ادامه بخشی از نکات مثبت استفاده از کاتلین برای توسعه ی نرم افزار خود را می بینید.

الف) زبان ساده : کاتلین یک زبان کاربردی و بسیار راحت برای یادگیری است. نحوه ی نوشتن آن به شدت شبیه جاواست، در نتیجه بسیار راحت به خاطر سپرده می شود. کاتلین بیان بیشتری دارد و از این جهت کد شما قابل فهم تر و قابل خواندن می شود.

ب) مختصر: کاتلین بر اساس JVM است و زبانی کاربردی است. در نتیجه تعداد کد های بی کاربرد که در زبان های برنامه نویسی دیگر استفاده می شوند، کم می شود.

پ) زمان اجرا و عملکرد : عملکرد بهتر و زمان اجرای کم

ت): قابلیت انتقال اطلاعات : کاتلین به حدی پیشرفته است که بتوان با آن یک نرم افزار انتقال اطلاعات با راهی نه چندان پیچیده ساخت.

ث) جدید : کاتلین یک زبان جدید است که به توسعه دهنده ها شروعی تازه می دهد. جایگزینی برای جاوا نیست، از آنجا که روی JVM توسعه داده شده است. این زبان به عنوان اولین زبان رسمی توسعه ی اندروید مورد قبول واقع شده است. کاتلین را می شود به شکل زیر تعریف کرد :

کاتلین = جاوا + ویژگی های جدید و آپدیت شده.

در ادامه تعدادی از معایب کاتلین را می بینیم.

ج) فضای نام : کاتلین به توسعه دهنده ها اجازه می دهد که عملکرد ها را در سطح بالا اعلام کنند. اما، هر وقت همان عملکرد در بخش های مختلفی از نرم افزار درخواست شود، آنوقت فهمیدن این که کدام درخواست خواسته شده است سخت می شود.

چ) عدم اعلام ثابت : تغییر دهنده ی بررسی کننده ی ثابتی مثل جاوا ندارد، که می تواند برای کسی که توسعه دهنده ی جاوا است مشکل ساز شود.

آموزش نصب کاتلین

نصب روی سیستم محلی

آموزش نصب کاتلین

نصب روی سیستم محلی

به هر طریق، اگر هنوز می خواهید از کاتلین آفلاین رو سیستم محلی تان استفاده کنید، باید مراحل زیر را طی کنید تا سیستم کحل کارتان را پیکر بندی کنید.

مرحله ی ۱: نصب جاوا ۸

کاتلین روی JVM کار می کند، در نتیجه بسیار ضروری است که از JDK8 برای توسعه کاتلین تان استفاده کنید. برای دانلود و نصب JDK8 به وبسایت رسمی oracle مراجعه کنید.
Environment Variable را برای جاوا به گونه ای تنظیم کنید که به خوبی کار کند.برای تایید نصب در سیستم عامل ویندوزتان عبارت : java-version را در command prompt وارد کنید که در خروجی به شما نسخه ی جاوایی که روی سیستمتان نصب است را می دهد.

مرحله ی 2: نصب IDE

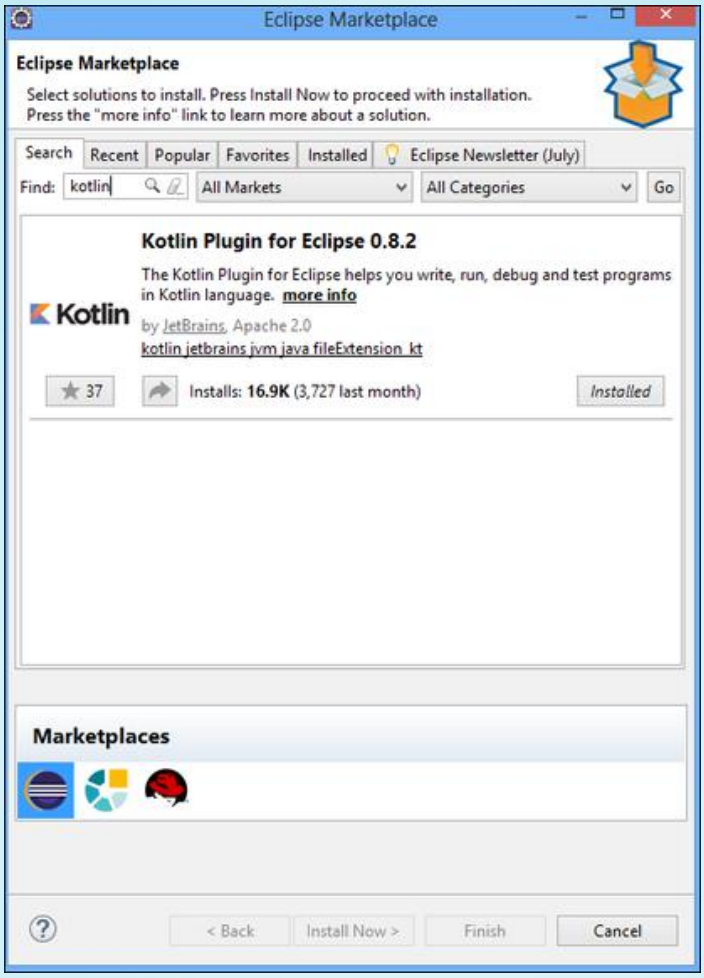
تعداد زیادی IDE روی اینترنت وجود دارد. می توانید از هر کدام که می خواهید استفاده کنید.
لینک دانلود IDE های مختلف در جدول زیر موجود است.

IDE Name	Installation Link
NetBeans	https://netbeans.org/downloads/
Eclipse	https://www.eclipse.org/downloads/
Intellij	https://www.jetbrains.com/idea/download/#section = windows

توصیه می شود که همیشه از آخرین نسخه ی نرم افزار استفاده شود تا بهترین تجربه ی برنامه نویسی را داشته باشید.

مرحله ی 3: راه اندازی و پیکربندی Eclipse

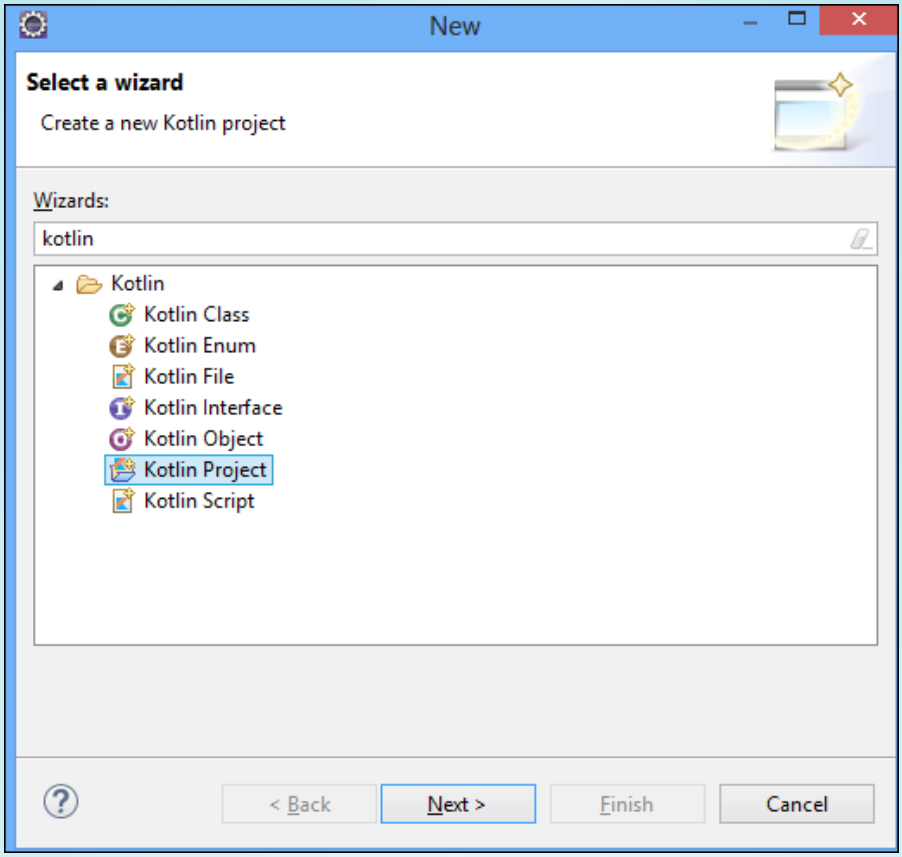
Eclipse را باز کنید و به Eclipse Market Place بروید. صفحه ی زیر را می بینید.



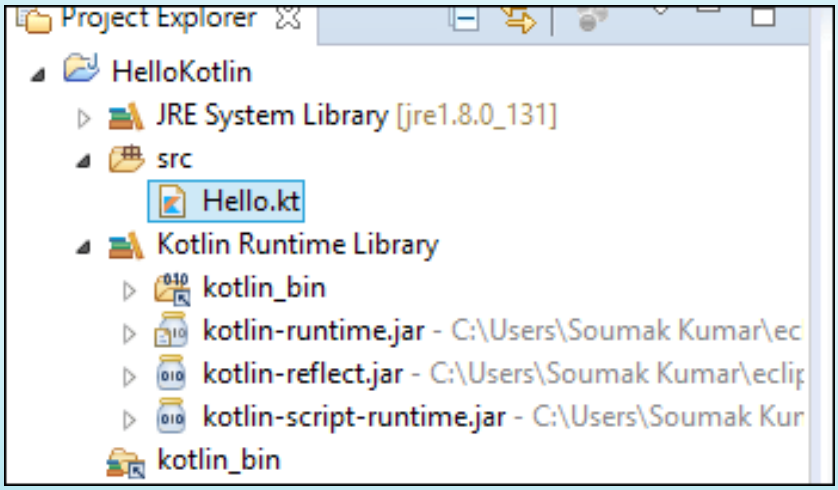
Kotlin را در نوار جستجو جستجو کنید و همین را روی سیستم محلیتان نصب کنید. با توجه به سرعت اینترنتتان ممکن است کمی طول بکشد. احتمالا باید بعد از نصب Eclipse خود را ری استات کنید.

مرحله ی 4: پروژه ی Kotlin

زمانی که Eclipse با موفقیت ری استارت شد و کاتلین نصب شده است، می توانید پروژه ی کاتلین بسازید. به file سپس new سپس others بروید و Kotlin project را از لیست انتخاب کنید.



زمانی که نصب پروژه تمام شد می توانید یک پوشه ی کاتلین زیر فولدر SRC ایجاد کنید.روی فولدر SRC کلیک چپ بزنید و new را انتخاب کنید. یک انتخاب برای فایل کاتلین به شما داده می شود، در غیر این صورت باید از others جستجو کنید. وقتی فایل جدید ساخته شد، راهنمای پروژه ی شما به شکل زیر خواهد بود.



اکنون محیط توسعه ی شما آماده است. کد زیر را در فایل Hello.kt وارد کنید.

```
fun main(args: Array) {
println("Hello, World!")
}
```

آن را به عنوان یک Kotlin application اجرا کنید و نتیجه را در میز فرمان همانطور که در تصویر زیر نشان داده شده است ببینید. ما برای فهم و دسترسی بهتر، از ابزار های پایه ی کد نویسی خودمان استفاده می کنیم.

Class & Object

در این بخش ، اصول اولیه ی برنامه نویسی مقصود گرا با استفاده از کاتلین را یاد میگیریم. در مورد طبقه و مقصودش و این که چطور با آن مقصود کار کنیم می آموزیم. طبق تعریف OOP ، طبقه طرح اولیه ی بخش ران تایم است و مقصود وضعیت آن است، که شامل رفتار و هم وضعیت آن می شود. در کاتلین، اعلام طبقه از سرپیام طبقه و یک بدنه ی طبقه تشکیل شده که است که توسط جفتی مجعد احاطه شده است. شبیه به جاوا.

```
Class myClass { // class Header
  // class Body
}
```

مثل جاوا، کاتلین اجازه ی ساخت مقصود های مختلفی از یک طبقه را می دهد، و شما مختارید تا اعضای طبقه و کارکرد هایشان را اضافه کنید. می توانیم قابلیت رویت متغیر های اعضای طبقه را با استفاده از کلیدواژه های مختلفی که در بخش 10 (کنترل قابلیت رویت) یاد خواهیم گرفت کنترل کنیم. در مثال زیر، طبقه ای همراه با مقصود هایش می سازیم که به وسیله ی آن ها به اعضای داده ی آن طبقه دسترسی پیدا می کنیم.

```
cclass myClass {
  // property (data member)
  private var name: String = "iracode.com"

  // member function
  fun printMe() {
    print("You are at the best Learning website Named-"+name)
  }
```

```
}
fun main(args: Array<String>) {
 val obj = myClass() // create obj object of myClass class
 obj.printMe()
}
```

کد های بالا خروجی زیر را در مرورگر به نمایش می گذارند، که در حال فرا خواندن printMe() از myClass با استفاده از مقصود هایش هستیم.

طبقه ی آشیانه ای

طبق تعریف، وقتی که طبقه ای درون طبقه ی دیگری ساخته شده باشد به آن طبقه ی آشیانه ای می گویند. در کاتلین، طبقه ی آشیانه ای به صورت خودکارثابت است، در نتیجه، بدون ساختن مقصودی برای آن طبقه قابل دسترسی است. در مثال زیر، می بینیم که چطور کاتلین طبقه ی آشیانه ای ما را تفسیر می کند.

```
 fun main(args: Array<String>) {
  val demo = Outer.Nested().foo() // calling nested class method
  print(demo)
}
class Outer {
  class Nested {
    fun foo() = "Welcome to The iracode.com"
  }
}
```

کد های بالا خروجی زیر را در مرورگر نمایش می دهند.

طبقه ی درونی

وقتی یک طبقه ی آشیانه ای به عنوان inner علامت گذاری شده است به آن طبقه ی درونی می گویند. یک طبقه ی درونی می تواند از طریق اعضای داده ی طبقه ی خارجی قابل دسترس باشد. در مثال زیر، به اعضای داده ی طبقه ی خارجی دسترسی پیدا می کنیم.

```
fun main(args: Array<String>) {
  val demo = Outer().Nested().foo() // calling nested class method
  print(demo)
}
class Outer {
  private val welcomeMessage: String = "Welcome to the iracode.com"
```

```
inner class Nested {
fun foo() = welcomeMessage
}
}
```

کد های بالا خروجی زیر را در مرورگر به نمایش خواهند گذاشت، که در آن طبقه ی آشیانه ای را با استفاده از سازنده ی پیش فرض کاتلین در زمان اجرا فرا می خوانیم.

طبقه ی داخلی بدون نام

طبقه ی داخلی بدون نام مفهوم بسیار خوبی است که زندگی یک برنامه نویس را به شدت آسان می کند. هر گاه در حال اجرای یک واسط باشیم، مفهوم بلوک داخلی بدون نام به میان می آید. مفهوم ساخت مقصودی واسط با استفاده از مرجع مقصود رانتایم به عنوان طبقه ی بدون نام شناخته می شود. در مثال زیر، یک واسط می سازیم و مقصودی برای آن واسط با استفاده از مکانیزم طبقه ی درونی بدون نام می سازیم.

```
fun main(args: Array<String>) {
  var programmer :Human = object:Human // creating an instance of the
interface {
    override fun think() { // overriding the think method
      print("I am an example of Anonymous Inner Class ")
    }
  }
  programmer.think()
}
interface Human {
  fun think()
}
```

کد های بالا خروجی زیر را در مرورگر نشان می دهند.

نام نمونه

نام های نمونه بخشی از برنامه ی مترجم کاتلین هستند. انعطافی برای تولید نامی جدید از نمونه ای موجود ایجاد می کنند، نمونه ای جدید ایجاد نمی کنند. اگر نام نمونه خیلی بلند باشد، به راحتی می توانید نامی کوتاه تر معرفی کرده و از آن برای استفاده های آینده استفاده کنید. نام های نمونه برای نمونه های پیچیده بسیار مفید اند. در آخرین نسخه، کاتلین پشتیبانی از نام های نمونه را لغو کرده است اما اگر از نسخه ها قدیمی استفاده می کنید می توانید به شکل زیر از آن استفاده کنید.

```
typealias NodeSet = Set<Network.Node>
typealias FileTable<K> = MutableMap<K, MutableList<File>>
```

سازنده ها

در این بخش ، در مورد سازنده ها در کاتلین اطلاعاتی کسب می کنیم. کاتلین دو نوع سازنده دارد. یکی سازنده ی اولیه و دیگری سازنده ی ثانویه. یک طبقه ی کاتلین می تواند یک سازنده ی اولیه و یک یا چند سازنده ی ثانویه داشته باشد. سازنده ی جاوا به متغیر های اعضا مقدار می دهد ولی در کاتلین سازنده ی اولیه به طبقه مقدار می دهد، در حالی که سازنده ی ثانویه سعی در منطق دادن در حین همان مقدار دادن می باشد. سازنده ی اولیه می تواند در سطح سرپیام طبقه مانند مثال زیر نشان داده شود.

```
class Person(val firstName: String, var age: Int){
 //class body
}
```

در مثال بالا، به سازنده ی اولیه داخل پرانتز دستور داده ایم. بین دو بخش، نام اول فقط قابل خواندن است همانطور که با "val" اعلام شده است، در حالی که بخش زمینه قابل ویرایش است. در مثال زیر از سازنده ی اولی استفاده می کنیم.

```
fun main(args: Array<String>) {
  val person1 = Person("Iracode.com", 15)
  println("First Name = ${person1.firstName}")
  println("Age = ${person1.age}")
}
class Person(val firstName: String, var age: Int) {
}
```

کد های بالا به طور خودکار به دو متغیر مقدار دهی می کنند و خروجی زیر را در مرورگر به نمایش می گذارند.

همانطور که قبلا به آن اشاره شده بود، کاتلین اجازه می دهد تا یک یا تعداد بیشتری سازنده ی ثاونیه برا طبقه ی خود بسازیم. این سازنده ی ثانویه با استفاده از کلیدواژه ی "constructor" ساخته شده است.لازم است هر وقت می خواهید بیش از یک سازنده در کاتلین داشته باشید یا هر وقت که می خواهد منطق بیشتری به سازنده ی اولیه ی خود اضافه کنیدو به خاطر این که سازنده ی اولیه ممکن است توسط طبقه ی دیگری فراخوانده شده باشد، نمی توانید این کار را

کنید. به مثال زیر توجه کنید، که در آن یک سازنده ی ثانویه ساخته ایم و از مثال بالا برای همان اجرا استفاده کرده ایم.

```
fun main(args: Array<String>) {
  val HUman = HUman("iracode.com", 25)
  print("${HUman.message}"+"${HUman.firstName}"+
    "Welcome to the example of Secondary  constructor, Your Age
is-${HUman.age}")
}
class HUman(val firstName: String, var age: Int) {
  val message:String  = "Hey!!!"
 constructor(name : String , age :Int ,message :String):this(name,age) {
  }
}
```

توجه: هر تعدادی از سازنده های ثانویه قابل ساخت است، اما، همه ی آن سازنده ها باید به سازنده ی اولیه پاسخ گو باشند چه به صورت مستقیم چه غیر مستقیم.

کد های بالا خروجی زیر را در مرورگر نشان می دهند.