خب حالا بیاین در مورد ساختار یک برنامه Kotlin صحبت کنیم. یک فایل کاتلین به نام main.kt میسازیم و داخلش خط های زیر را مینویسیم :

fun main**(**args: Array<String>**)** {  
 println**(**"Hello World!"**)**}

خب در اینجا به یک نکته باید توجه کنیم. این که کاتلین نیازی به نقطه ویرگول";" نداره و اضافه کردن نقطه ویرگول کاملا دلخواه است. تنها در یک جا هست که در شرایط ویژه باید از نقطه ویرگول استفاده کنیم که بعدا در مورد اون صحبت خواهیم کرد ولی در موارد غیر اون استفاده از نقطه ویرگول کاملا دلخواه است.

اگر متوجه شده باشید کلیدواژه fun نشان دهنده اینه که ما چطور در زبان کاتلین یک تابع رو میسازیم و اگر دقت کنید من یک تابع رو داخل یک فایل نوشتم و نیاز به ساختن هیچ کلاسی نداشتم و این هم به این دلیل است که ما در کاتلین چیزی به نام Top-Level Functions داریم. دقیقا به مانند جاوااسکریپت، میتونیم یک فایل رو باز کنیم و یک تابع رو داخلش بنویسیم، بدون نیاز به نوشتن کلاسش.

چرا من اسم این تابع رو main گذاشتم؟ باید بدونیم که main یک نقطه شروع برای کامپایلر به حساب میاد. درواقع مشخص کردم که این یک تابع ورودی برنامه است.

خب تا اینجا من این فایل رو ذخیره میکنم. و سپس دستور زیر رو توی کنسول میزنم:

>>kotlinc Main.kt

و این درواقع یک سری فایل رو برای من خروجی میده. یک فولدر به نام META-INF که اینجا مورد نظرمون نیست، و یک MainKt.class که در واقع همان بایت کد های جاوایی هست که کاتلین برامون ساخته.

خب برگردیم سراغ فایل Main، اگر دقت کنین میبینین که هیچگونه اطلاعاتی در مورد package هم نیست! ما در کاتلین مفهوم پکیج رو داریم ولی اگر پکیج مشخص نشه، کاتلین اون رو به صورت پکیج دیفالت در نظر میگیره ولی من میتونیم بیام و به اون صورت کدم رو تغییر بدم:

package com.kotlinfarsi.introduction  
  
fun main**(**args: Array<String>**)** {  
 println**(**"Hello World!"**)**}

حالا اگر دوباره فایلمون رو کامپایل کنیم،با همون دستور قبلی:

>>kotlinc Main.kt

و حالا اگه خروجی رو نگاه کنیم متوجه میشیم که فولدری به نام com ساخته شده است و اگر به داخلش بریم متوجه میشیم که داخلش یک فولدر دیگه ای هست به نام kotlinfarsi و اگر به داخل ان رفته فولدر دیگری به نام introduction خواهیم دید و اگر به داخل ان نیز رفته متوجه خواهیم شد که فایلی به نام MainKt.class در اونجا ساخته شده است. درواقع کاتلین اجباری به مسئله Packaging نمیکنه ولی در واقع برای مرتب بودن کدمون پیشنهاد میشه که همچین کاری صورت بگیره و در واقع همونطور که قبلا گفتیم کاتلین jvm رو هدف خودش قرار داده و این مسئله Packaging کاریه که خیلی از توسعه دهنده های جاوا انجام میدم واین درواقع کار خوبیه.

خب حالا بیاین برناممون رو اجرا کنیم، به سه فولدر قبل برمیگردیم، همونجایی که فولدر com در اون قرار داشت و همینطور فایل Main.kt و فایل کامپایل اولمون MainKt.class .

خب اگر دستور زیر رو وارد کنیم :

>> java com.kotlinfarsi.introduction.MainKt

با ارور مواجه میشیم!! دلیل این ارور هم اینه که کاتلین به همراه یک runtime ارائه شده!! درواقع شما برای اجرای برنامه های کاتلین باید این رانتایم رو همراه برنامتون کنین. میتونیم این رانتایم رو به کامپایلر include کنیم، درواقع همراه کامپایلرمون جاساز کنیم و دلیل این که برنامه ناموفق بود اینه که نتونست رانتایم رو پیدا کنه. من نمیدونم شما رانتایمتون رو کجا ذخیره کردین. اگر فایل زیپ شده رو دانلود کردین داخل فولدر lib و یا اگر با brew نصبش کردین با دستور which پیداش کنین، به هر حال اگه به محل رانتایمتون برین متوجه خواهید شد که فایلی به نام kotlin-runtime.jar داریم که همون رانتایم مورد نظرمونه.

خب به هر حال ما میخوایم فایلمون رو اجرا کنیم و یک روش اینه که شما از همون اول مشخص کنین که فایل رانتایمتون کجا قرار داره، مثلا کد زیر رو بزنین:

>> java –cp <ادرس فایل رانتایم> com.kotlinfarsi.introduction.MainKt

Hello World

بعله، فایلمون اجرا شد، خروجی رو نمایش دادیم!

ولی خب اگه چنتا فایل کاتلین داشتیم چی ؟ باید فایل class همه ی اون ها رو به کاربر بفرستیم و بگیم اجراش کن؟ خب درواقع همونطور که جاوا برای این منظور فایل .jar درست میکرد، کاتلین هم همین کار رو میکنه. تنها کاری که باید بکنیم اینه که دستور زیر رو وارد کنیم:

>>kotlinc Main.kt –d hello.jar

و اگر نگاه کنیم فایلی به نام hello.jar در کنار فایلامون خواهیم داشت. خب اگر من بخوام اجراش کنم بازم با همون ارور مواجه میشم چراکه نمیتونه فایل رانتایم رو پیدا کنه. کامپایلر کاتلین درواقع به ما این امکان رو میده که بتونیم رانتایم رو include کنیم :

>>kotlinc Main.kt –include-runtime –d hello.jar

و حالا تنها کاری که من باید انجام بدم اینه که دستور زیر رو وارد کنم:

>>java –jar hello.jar

Hello World!