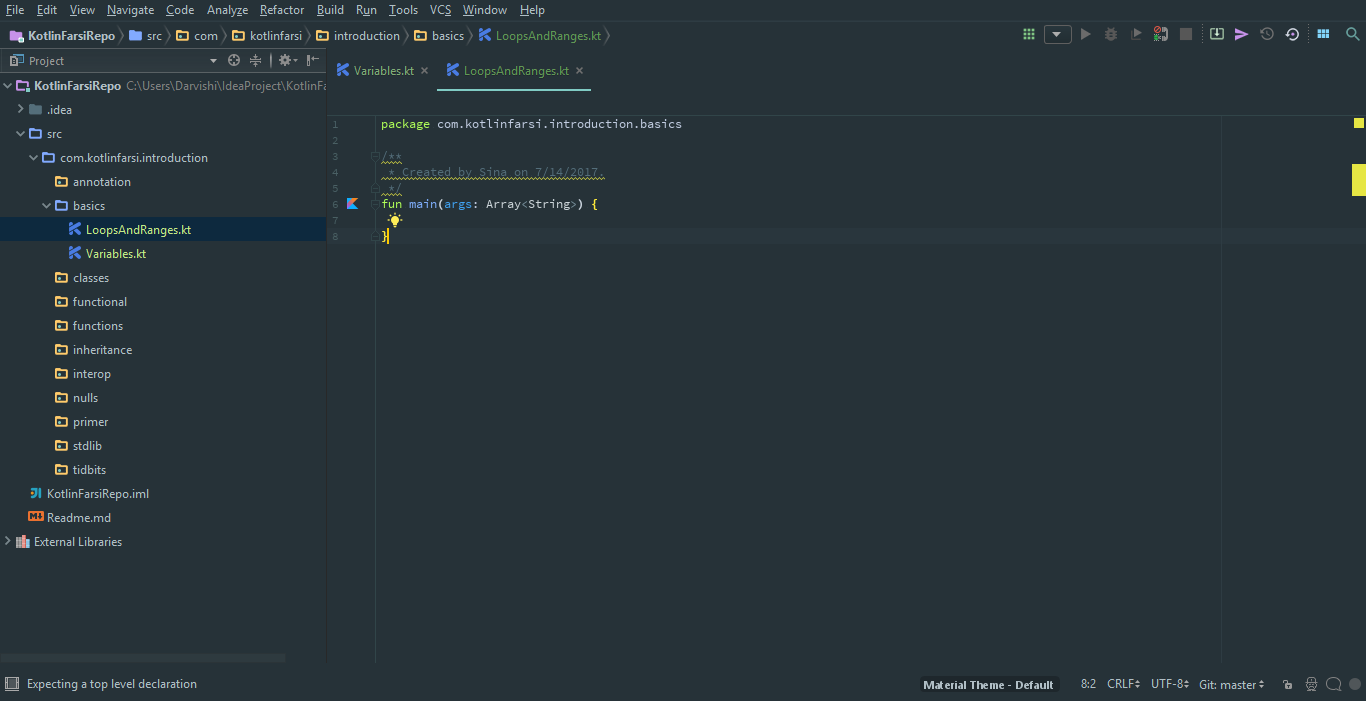
حالا بیاین یکم نگاه حلقه ها کنیم،

یک فایل دیگه به نام LoopsAndRanges درست میکنیم و main رو داخلش مینویسیم



اگه دقت کنیم داخل این فایل هم main داریم و داخل فایل قبلی هم main داریم. ایا همچین چیزی ممکن است؟ درواقع کاتلین همیچین اجازه رو میده که به ازای هر فایل یک main داشته باشیم.

خب حالا از این قضیه بگذریم، بیاین یک حلقه for درست کنیم

for**(**a:Int in 1..100**)**{  
 println**(**a**)**}

ابتدا نام یک متغیر رو تعریف میکنیم که میخوایم بر روی اون راه بریم. سپس تایپش رو معرفی میکنیم سپس با استفاده از کلید واژه in میگیم که میخوایم در چه محدوده ای(هرچی جلوش میاد) راه بریم و بعدش محدوده رو تعریف میکنیم و در انتها هم هرچی جلوی اکولاد ها میاد رو هر دفعه انجام میده. در واقع ما اینجا هم میتونیم تایپ رو مشخص نکنیم و خود کامپایلر میفهمه منظورمون کدوم تایپه.

نکته ای که در اینجا باید توجه کنیم بهش اینه که اون ". ." که اومده درواقع همان تابع rangeTo() است. در واقع ما میتونیم اینجور بنویسیم:

val numbers = 1..100

که درواقع داریم یک لیست درست میکنیم از اعداد 1 تا 100 . و اگه بخوام یک حلقه for درست کنیم میتونیم از همین numbers استفاده کنیم

for**(**a in numbers**)**{  
 print**(**a**)**}

حالا اگه بخوایم یک حلقه برعکس بنویسیم میتونیم به این روش انجام بدیم

for**(**a in 100 downTo 1**)**{  
 println**(**a**)**}  
  
for**(**a in 100..1**)**{  
 println**(**a**)**}

همینطور که میبینید میتونیم به دو روش انجام بدیم، یا مستقیم از downTo استفاده کنیم و یا از ".. " .

حالا فرض کنید میخوایم از 100 تا 1 پایین بیایم ولی 5 تا 5 تا :

for**(**b in 100..1 step 5**)**{  
 println**(**b**)**}

ولی خب این تنها یک حلقه برروی شماره ها بود، کاتلین این اجازه رو میده که برروی دسته ها هم راه برید. مثلا کد زیر رو در نظر بگیرین

val capitals = listOf**(**"London","Paris","Rome","Madrid"**)**for**(**capital in capitals**)**{  
 println**(**capital**)**}

حالا کار نداشته باشین اون listOf چیه و چگونه کار میکنه فعلا فقط بدونین که یک لیست درست میکنه. به همین راحتی این لیست رو به for میدیم و اون روی این لیست راه میره و تموم اون مرکز شهر هارو چاپ میکنه.

خب حلقه ها تنها for نیستند، مثلا while و do while نیز به صورت زیر نوشته میشوند

var i = 100  
while **(**i>0**)**{  
 i--  
}  
var x = 10  
do {  
 x--  
}while **(**x>0**)**

حالا خارج از بحث حلقه ها، کاتلین این ویژگی را به کاربر میدهد که بتواند حلقه را ادامه دهد یا از حلقه بیرون بیایید. به عنوان مثال به تکه کد زیر دقت کنید:

loop@ for **(**i in 1..100**)**{  
 for**(**j in 1..100**)**{  
 if**(**i%j == 0**)** break@loop  
 }  
}

در اینجا ما با هدف قرار دادن حلقه اول از حلقه دوم خارج میشویم به این صورت که وقتی break را مینویسم با یک @ یک لیبل را مشخص میکنیم که میخواهیم به ان وارد شویم. و همینطور با continue

loop@ for **(**i in 1..100**)**{  
 for**(**j in 1..100**)**{  
 if**(**i%j == 0**)** continue  
 }  
}