کاتلین یک ویژگی خیلی مهم رو ساپورت میکنه به نام توابع الحاقی، که درواقع از C# الهام گرفته شده. ایده پشت این ویژگی اینه که من بتونم یک تابع رو به یک کلاس الحاق کنم بدون این که از اون کلاس ارث بری کنم

کد زیر رو نگاه کنین:

fun String.hello(){  
 *println*("It's me")  
}

در واقع اینجا یک الحاق به کلاس String انجام دادیم. یک تابع جدید به کلاس String اضافه کردیم که تنها یک رشته رو چاپ میکنه و برای استفاده ازش کافیه بنویسیم :

fun String.hello(){  
 *println*("It's me")  
}

fun main(args: Array<String>) {  
 "Sina".*hello*()  
}

و خب اگه اجراش کنین متوجه میشین که تابع الحاقیمون رو اجرا کرده.

خب بیاین یک تابع مفیدتر بنویسیم:

fun String.toTilteCase(): String {  
 return this.*split*(" ").*joinToString*(" ") **{** it.*capitalize*() **}**}

نیاز نیست که حالا درمورد این زنجیره و توابع داخلش نگران باشین، چیزی که اینجا مهمه اینه که با استفاده از this مقدار خود String که این تابع به اون الحاق شده رو داخل تابع الحاقیمون استفاده کردیم.

fun String.toTilteCase(): String {  
 return this.*split*(" ").*joinToString*(" ") **{** it.*capitalize*() **}**}

fun main(args: Array<String>) {  
 *println*("this is a sample string to title case".*toTilteCase*())  
}



خلاصه بخش 9:

1. ما توی این قسمت نگاهی به توابع High-Order انداختیم و دیدیم که توابع High-Order حساب میشن که یا یک تابع رو به عنوان ورودی قبول کنن و یا این که تابع رو به عنوان خروجی پاس بدن
2. دیدم که کاتلین استفاده از Lambda را ممکن کرده
3. دیدم که کاتلین اجازه دسترسی به Closure های تغییر داده شده رو میده، به این معنی که اگر در داخل عبارت لاندامون از متغییری استفاده کردیم که تغییر میکنه، اون تغییر به داخل لاندامون هم سرایت میکنه.
4. و دیدیم که توابع الحاقی توابعی هستند که میتونیم اون هارو بدون ارث بری از کلاس به اون کلاس اضافه کنیم. که این کار رو درواقع برای کلاس های جاوا هم میتونیم انجام بدیم.