کاتلین ارث بری رو مثل بقیه زبون های شی گرا ساپورت میکنه و توی کاتلین، کلاس پایه ای همه ی کلاس ها Any هستش. یعنی درواقع اگه یک کلاس باز کنم به این فرم

class Customer

در واقع هیچ فرقی نداره با این که کلاسی باز کنم به این فرم

class Customer: Any**()**

خب حالا بیاین یه کلاس پدر برای Customer بسازیم.

class Person  
  
class Customer: Person**()**

فقط یادتون باشه وقتی کلاس رو inherit میکنین کانستراکتورش رو هم مشخص کنین.

اگه کد بالا رو بزنین متوجه میشین که ارور توی برنامتون هست.مشکل اینجاست که توی کاتلین همه ی تایپ ها final هستند بنابراین نمیشه ازشون ارث بری بشه.یه جورایی میشه گفت برعکس جاوا که همیشه میتونیم ارثبری کنیم مگر این که final باشه، توی کاتلین هم میشه گفت همیشه میتونیم از یک کلاس ارث بری کنیم اگر اون کلاس open باشه.

open class Person  
  
class Customer: Person**()**

برای توابع هم اگه میخوایم ازشون ارث بری بشه باید open قبلشون بیاد

open class Person{  
 open fun validate**()**{  
  
 }  
}  
  
class Customer: Person**()**{  
 override fun validate**()** {  
 super.validate**()** }  
}

خب حالا اگه به همراه main بخوایم تابع validate رو صدا کنیم داریم

open class Person{  
 open fun validate**()**{  
  
 }  
}  
  
class Customer: Person**()**{  
 override fun validate**()** {  
 super.validate**()** }  
}  
  
fun main**(**args: Array<String>**)** {  
 val customer = Customer**()** customer.validate**()**}

خب حالا فرض کنین توی Customer هستیم و میخوایم کانستراکتور پدرش رو صدا بزنیم، یادتونه گفتیم یه کانستراکتور دوم داریم؟ اینجا تقریبا میخوایم از همون استفاده کنیم، با این تفاوت که به جای this باید از super استفاده کنیم

open class Person{  
 open fun validate**()**{  
  
 }  
}

class Customer: Person{  
 override fun validate**()** {  
 super.validate**()** }  
  
 constructor**()**:super**()**{  
   
 }  
}

فقط اینجا توجه کنین که علاوه بر این که به جای this از super استفاده کردیم پس باید کانستراکتور Person رو زمان ارث بری حذف کنیم، یعنی مثلا اینجا اومدیم و پرانتز های جلوی Person رو برداشتیم.