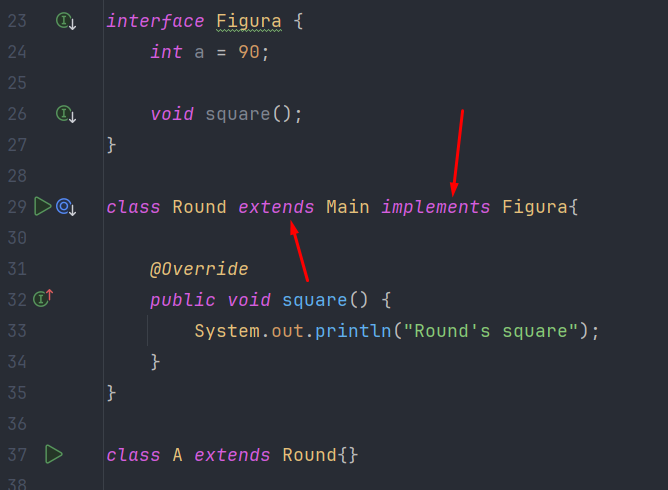
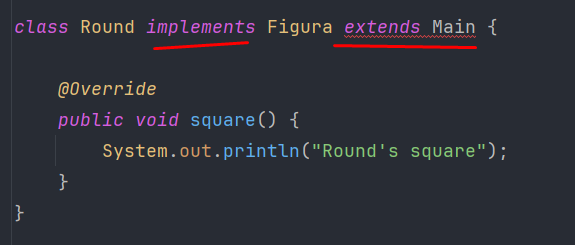
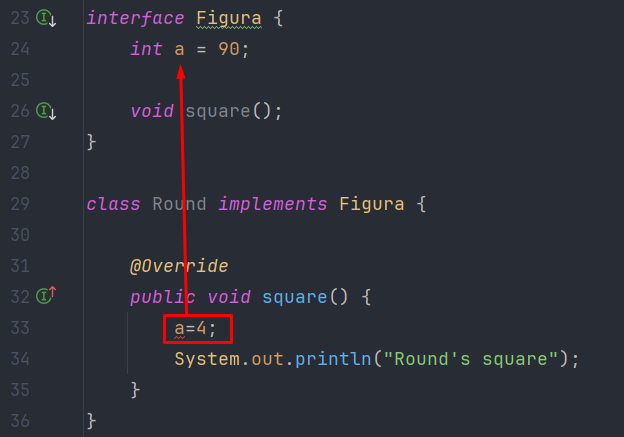
Agar biz bir vaqtni o’zida ham classdan meros oladigan bo’lsak va ham interface dan implement oladigan bo’lsak, u holda avval classdan extends olinadi undan keyin esa interfacedan implement olinadi:

Lekin teskari tartibda bo’lsa, xatolik beradi:

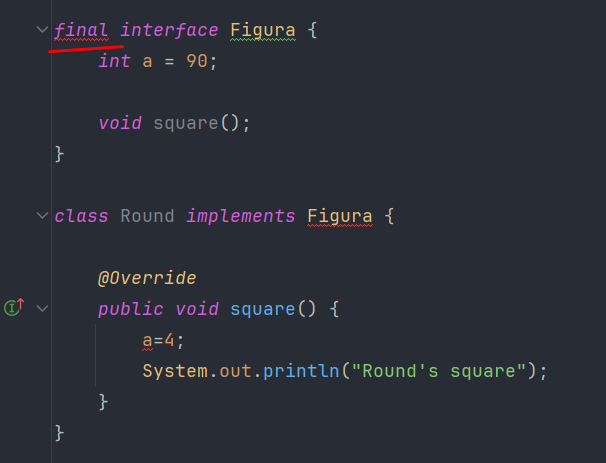


Interface da yaratilgan o’zgaruvchilarga avtomatik ravishda public static final kalit so’zlari qo’shiladi. Bundan chiqadiki bu o’zgaruvchilarga qayta qiymat o’zlashtirib bo’lmas ekan, chunki bu o’zgaruvchilarni yuqorida ko’rdik final keywordi qo’shiladi. Demak xatolik beradi. Pastda ham Figure interfaceda yaratilgan a o’zgaruvchisi avtomatik final bo’ladi va uni qiymatini qaytib o’zgartirib bo’lmaydi. Lekin biz Round classda unga a=4; deb qiymat o’zlashtiryapmiz va natijada xatolik kelib chiqadi:

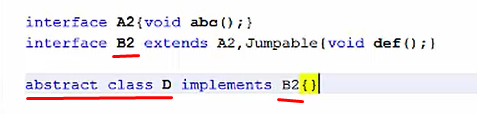


Yana bitta narsa deylik bitta **Book.java** file da faqat bitta **public class A{}** yoki bitta **public interface MyInterface{}** bo’lishi mumkin. Bir vaqtni o’zida ikkalasi ham **public** bo’la olmaydi. Aks holda xatolik beradi. Lekin **Book.java** ni ichida **public** bo’lmagan istalgancha **interface** yoki **class** yaratishimiz mumkin.

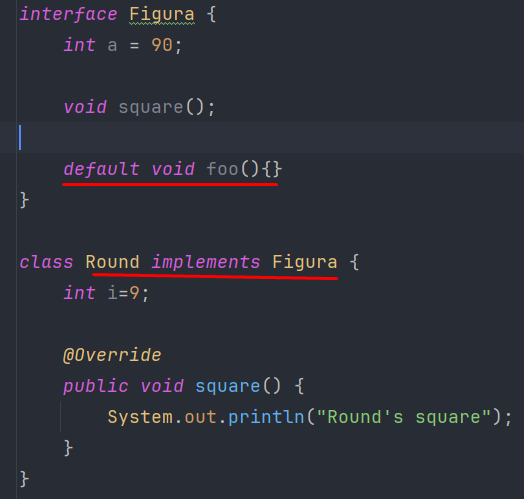
Interface lar **final** bilan e’lon qilinmaydi xuddi **abstract** **classlar** singari. Sababi **final** classdan biz meros ola olmaymiz. Xuddi shunday interface dan ham meros ola olmaymiz. Demak xatolik beradi:



Abstract class lar ham interface lardan implement qiladi:

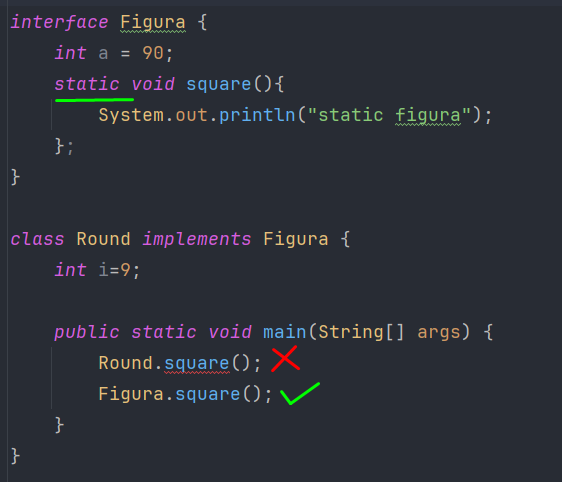


Interface da tanasi bor bo’lgan methodlar ham bo’lishi mumkin. Bunday methodlar default methodlar deb nomlanadi. Nega default methodlar kerak. Tasavvur qilaylik bizda bitta interface bor bo’lsin va uni ichida **100** ga yaqin abstract methodlari bo’lsin. Shu interfacedan deylik **50** ta class implement olgan bo’lsin. Endi biz shu interface ga biz yana bitta method qo’shishimiz kerak bo’lsin, albatta bu method abstract bo’ladi. U holda shu methodni 50 ta class ga kirib override qilib yozib chiqishga to’g’ri keladi, vaholanki bu yangi yaratilgan methodni atigi 5 ta classda ishlatsak ham. Bunday vaziyatda default method qulay bo’lib, bu default methodni boshqa class larda override qilish shart emas. Shuning uchun bu methodni yaratgach, uni override qilib o’tirish shart emas. Shu methodni ishlatmoqchi bo’lgan classlarga kirib, uni override qilib ishlatib ketaveramiz. Demak default methodlarni override qilish shart emas ekan, qilmasak ham xatolik bermaydi. Pastda default bo’lgan **foo()** methodni yozdik va uni override qilmadik:

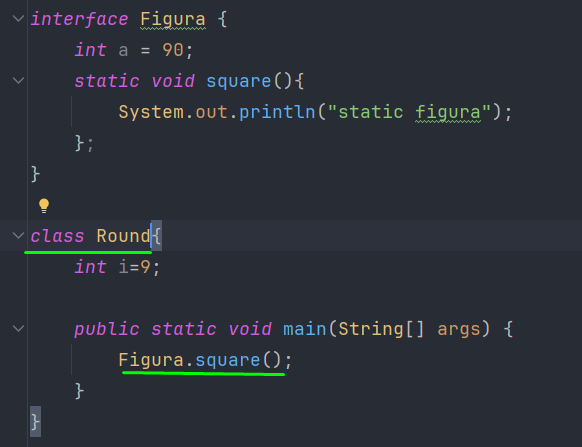


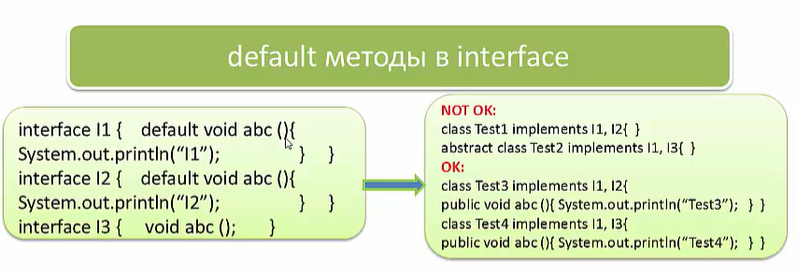
Bitta muhim qoida, default methodlar faqat interface da bo’ladi, boshqa konkret va abstract classlarda bo’lmaydi.

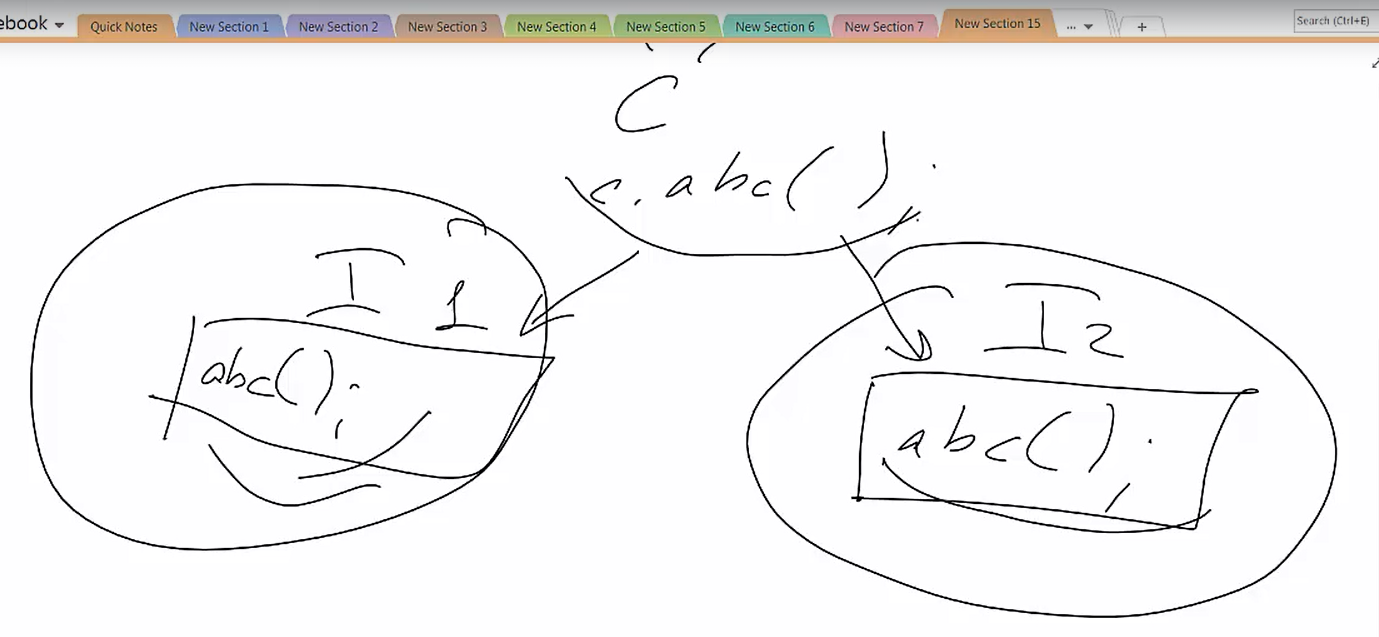
Agar interfaceda default method static bilan e’lon qilinsa, u holda bu static default methodni faqat shu interface nomi bilan chaqira olamiz xolos. Boshqacha chaqirishlar xato hisoblanadi:

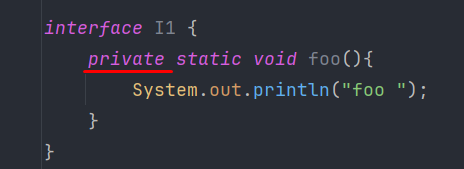


Interface ni ichidagi static default methodni xattoki undan implemetnt olmasdan ham boshqa class da chaqirish mumkin. Bu holatda oddiy classlarda static methodlarni chaqirish qanday ishlasa, bunda ham xuddi shunday ishlaydi:

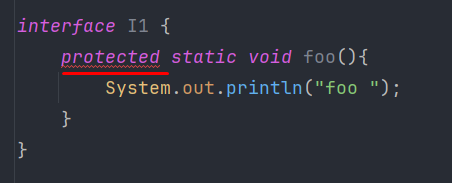




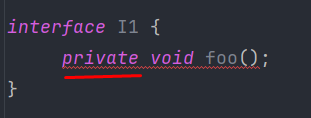


Java 9 dan boshlab, interfacelarda private access modifierni, oddiy va static methodlarda ishlatish mumkin bo’ldi. Interfaceda istasak private access modifieridan foydalanishimiz mumkin:

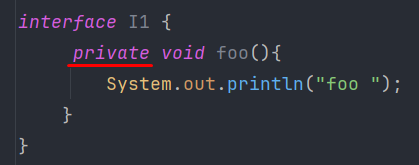
Private ni o’rniga protected yoza olmaymiz, xatolik beradi:



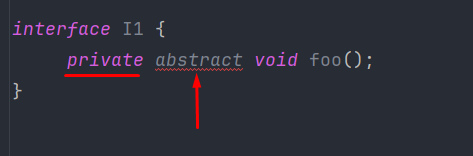
Agar private bilan methodni e’lon qilsak, interface da u hold, uni tanasi albatta bo’lishi shart, aks holda error beradi:



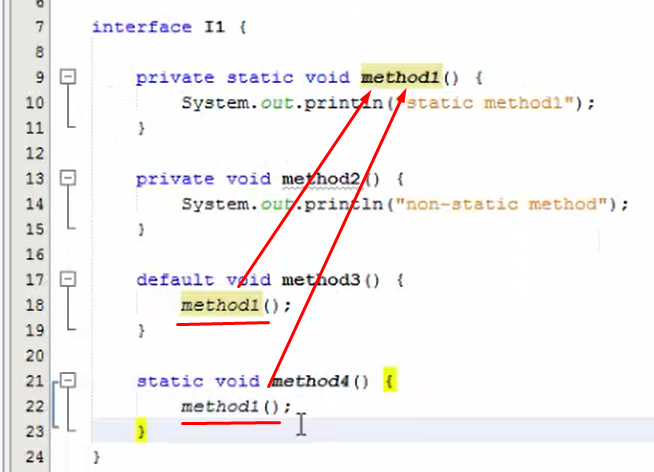
Agar tanasi bo’lsa **private** methodni, u holda xatolik yo’qoladi:



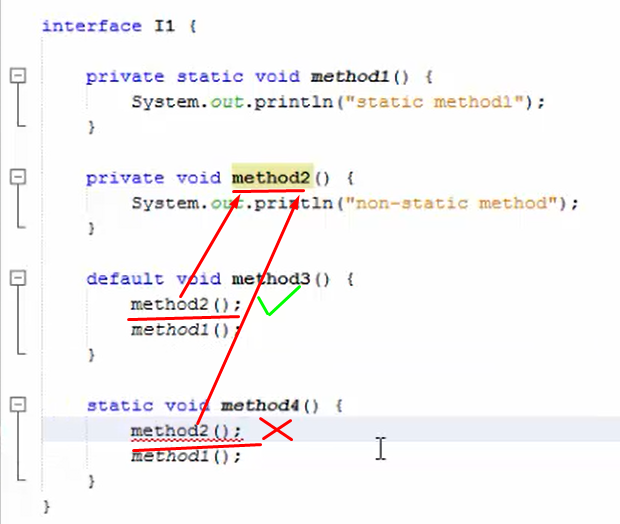
Agar private methodni abstract bilan e’lon qilsak, u holda xatolik beradi. Sababi private methodni albatta tanasi bo’lishi shart, lekin abstract methodda esa aksincha tanasi bo’lmasligi zarur. Shuning uchun xatolik beradi:



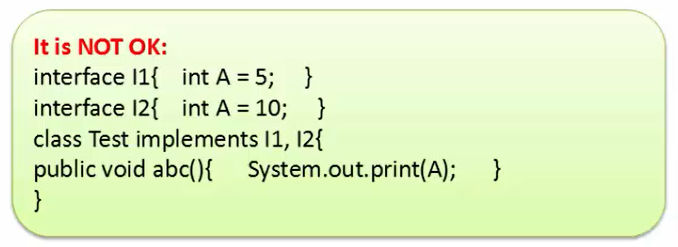
**Private static** methodni istasak static methodni ichida yoki non-static methodni ichida chaqirishimiz mumkin:



Private non-static methodni private-static methodni ichida ishlata olmaymiz. Chunki qoida bo’yicha static bo’lmagan methodlarni static methodlarni ichida chaqira olmaymiz, xatolik beradi:



Pastdagi misolda ko’radigan bo’lsak, I1 va I2 interfacelarda bir xil nomli A o’zgaruvchisi ishlatilyapti. Test classimiz bu 2 ta interfacedan implement olyapti, endi shu o’zgaruvchilarni class da chaqirmoqchi bo’lsak, xatolik kelib chiqadi. Sababi bu ikkala o’zgaruvchi bir xil bo’lgani uchun compilator bilmaydi qaysi birini chaqirishni:



Bunday vaziyatda Interfacelarni nomi bilan chaqirish shart, shundagina to’g’ri ishlaydi:

