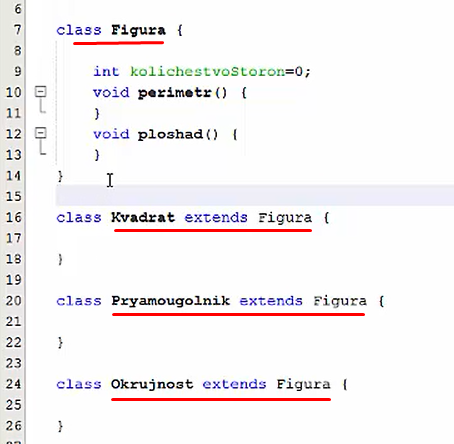
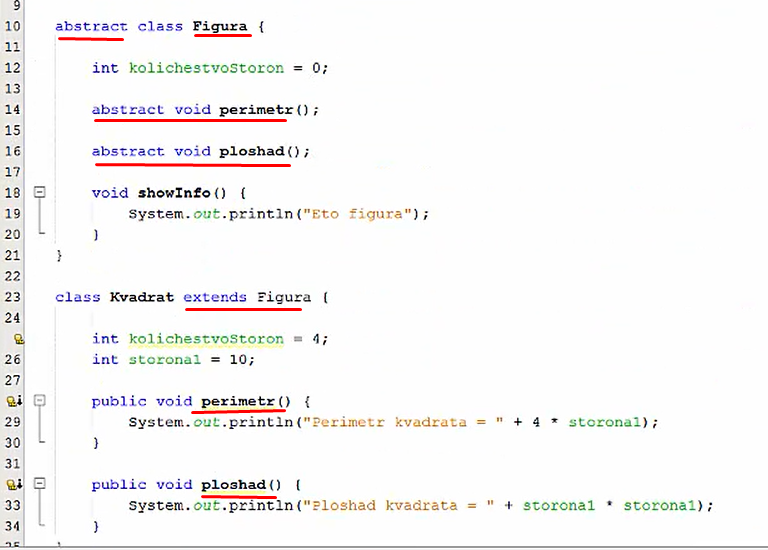
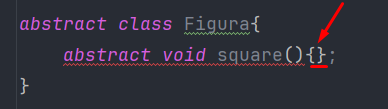
Pastdagi misolda **abstract** class nega kerakligini tushunamiz. E’tibor bergan bo’lsangiz, bizda **Figura** nomli class bor bo’lib, bu classdan **Kvadrat**, **Pryamougolnik**, **Okrujnost** kabi classlar meros olyapti. **Figura** classi subclasslar uchun umumiydir va uni ichida **perimetr()** va **ploshad()** methodlari bor bo’lib, bu methodlar bola classlarda har xil implement qilinadi, ya’ni **Kvadratni** perimetri bilan **Pryamougolnikni** perimetri mos tushmaydi, xuddi shunday **Figura** class dagi **ploshad()** method bola classlarda ham har xil implement qilinadi, ya’ni **Kvadratni** yuzasi bilan **Pryamougolnikniki** har xil hisoblanadi, xuddi shunday **Okrujnostniki** ham. Demak ota **Figura** class da bitta **perimeter()** nomli class yozib qo’yib, uni boshqa classlarda to’g’ridan to’g’ri ishlatish unchalik ham qulay emas ekan:



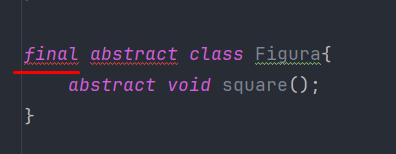
Endi yuqoridagi Figura nomli classni abstract qilamiz. Aslida figura ham abstractdir. Sababi bizga biror bir figura chizib bergin deyishsa, biz istalgan bir figura chizishimiz mumkin, chunki bizga aniq mana bu figurani chizib ber demadi. Istasak uchburchak yoki to’rtburchak, kvadrat yoki aylana yoki beshburchak chizishimiz mumkin. Sababi figura bu mavhum(asbtract)dir. Aniq bir figura haqida gapirilmayapti. Pastda biz Figura nomli abstract class yaratdik va unda perimetr() va ploshad() nomli abstract methodlar ham yozdik. Nega bu methodlar ham abstract bo’ldi. Sababi bu methodlar aylana uchun boshqacha hisoblanadi, kvadrat uchun boshqacha va to’rtburchak uchun boshqacha hisoblanadi. Shuning uchun aniq emas qaysi figura uchun hisoblash. Shunchaki umumiy qilib abstract deb e’lon qilib ketamiz. Aniq bir konkret figuraga qarab, uni hisoblashimiz mumkin:



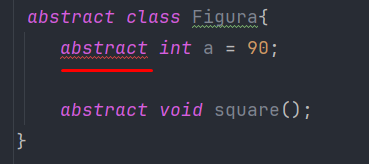
Abstract classda abstract methodni bodysi bo’lishi mumkin emas, xattoki shunchaki ichi bo’shi bo’lishi ham mumkin emas. Aks holda xatolik beradi, pastdagi kabi:



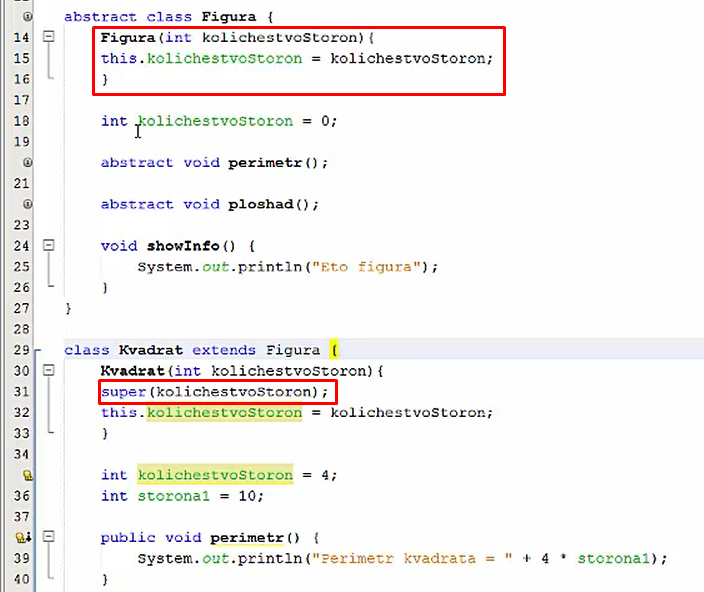
Abstract class **final** kalit so’zi bilan e’lon qilinmaydi. Sababi **final** classdan biz meros ola olmaymiz. Meros olaolmaganimizdan keyin, abstract classni ham ishlata olmaymiz. Abstract classni ishlatishimiz uchun esa, albatta undan meros olishimiz kerak. Pastdagi ifoda xato hisoblanadi:



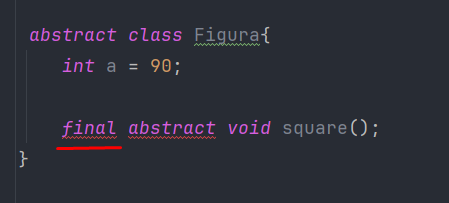
O’zgaruvchilar abstract bo’la olmaydi, faqat methodlargina abstract bo’la oladi. Pastdagi ifoda xato:



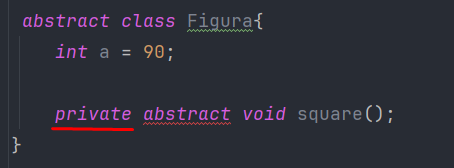
Abstract classda constructor bo’lishi mumkin. To’g’ri g’alati tuyulishi mumkin. Sababi abstract classdan object yarata olmaymiz, lekin bo’lishi mumkin. Hech qanday xatolik bermaydi:



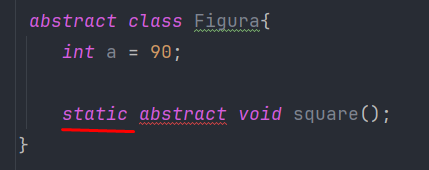
Abstract methodlarni **final** keyword bilan e’lon qila olmaymiz. Sababi **abstract** methodlarni bola classda albatta override qilib qayta yozishimiz kerak. Biz bilamizki **final** methodlarni override qilib bo’lmaydi. Demak mumkin emas ekan xatolik beradi:



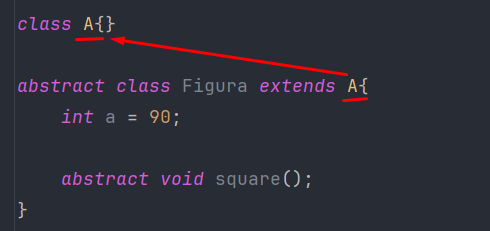
Xuddi shunday abstract methodlar **private** keyword bilan e’lon qilinmaydi, chunki **private** methodlar bola classda ko’rinmaydi va meros ham bo’lib o’tmaydi. Shuning uchun xatolik beradi:



Xuddi shunday **static** keyword bilan ham abstract methodlarni e’lon qila olmas ekanmiz. Sababi **static** methodlarni **override** qilib bo’lmaydi, demak xatolik beradi:



Konkret(oddiy) class dan abstract class meros olishi mumkin:



Agar abstract classdan konkret class meros olsa, u holda konkret class abstract classni hamma abstract methodlarini override qilishi shart. Lekin bu yerda yana bitta holat bor, agar shu konkret classdan boshqa bir konkret class meros olgan bo’ls, u holda bu konkret class abstract classni methodlarini override qilmasa ham bo’ladi. Chunki Ota(konkret) class abstract classdagi methodlarni override qilib yuboradi. Shuning uchun bola classda uni override qilish shart emas. Pastda abstract Figura classdan meros olgan konkret class Round bo’lib, undan meros olgan yana boshqa concret class A berilgan:

