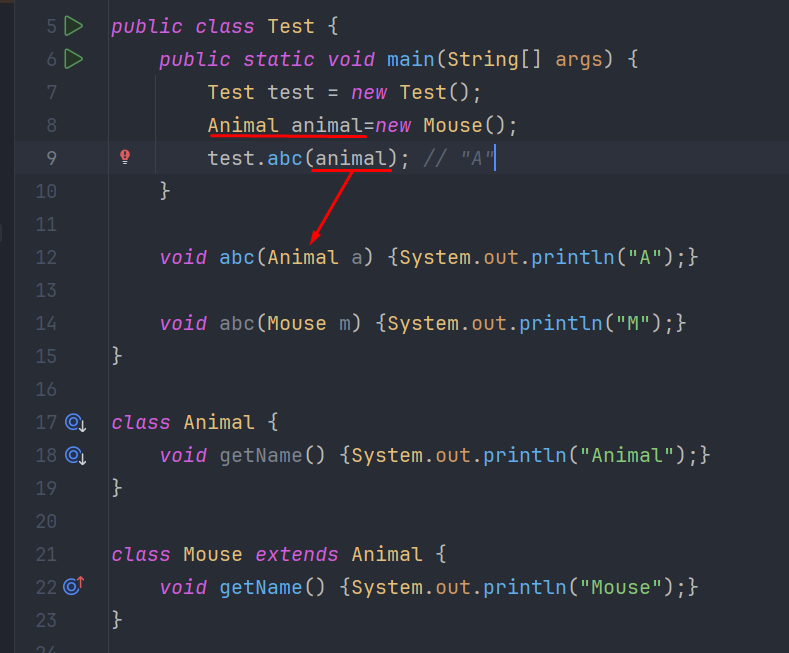
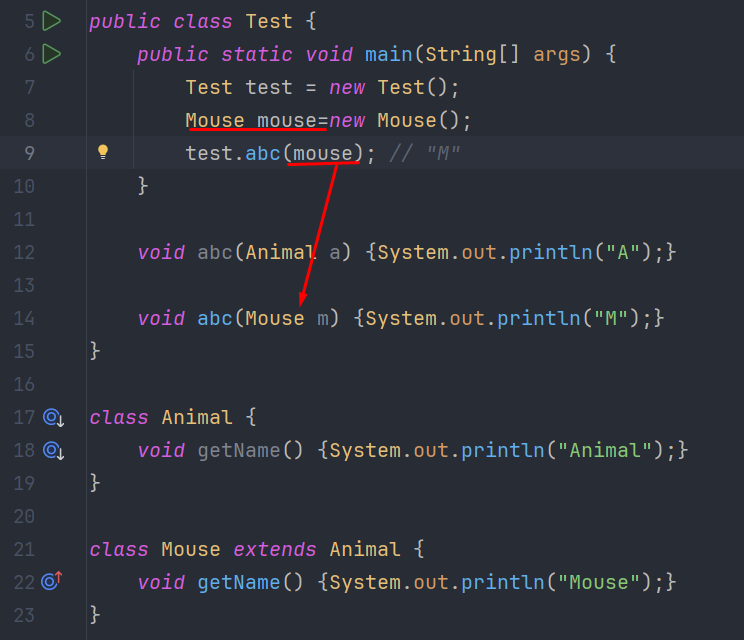


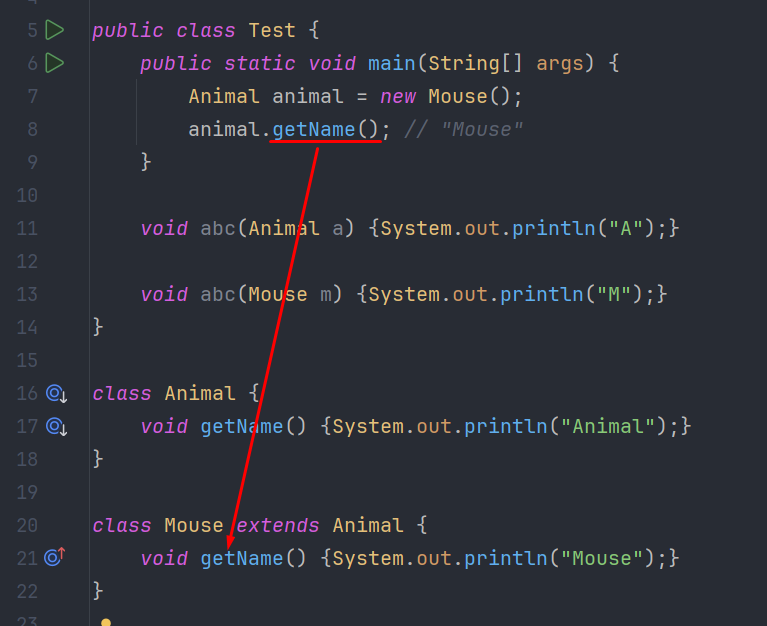
Hamma **o’zgaruvchilar** Compile time binding xususiyatiga ega. Bu degani hamma o’zgaruvchilarni type qanday bo’lsa, o’sha typega tegishli bo’lgan o’zgaruvchi va methodlarni chaqiradi degani. Pastdagi misolda ham **Animal animal=new Mouse();** ifodadagi **animal** o’zgaruvchisi compile time binding xususiyatiga ega bo’lib, abc() methodni chaqirganimizda, parametriga Animal ni qabul qiladigan 12-qatordagi abc() methodni chaqirdi.



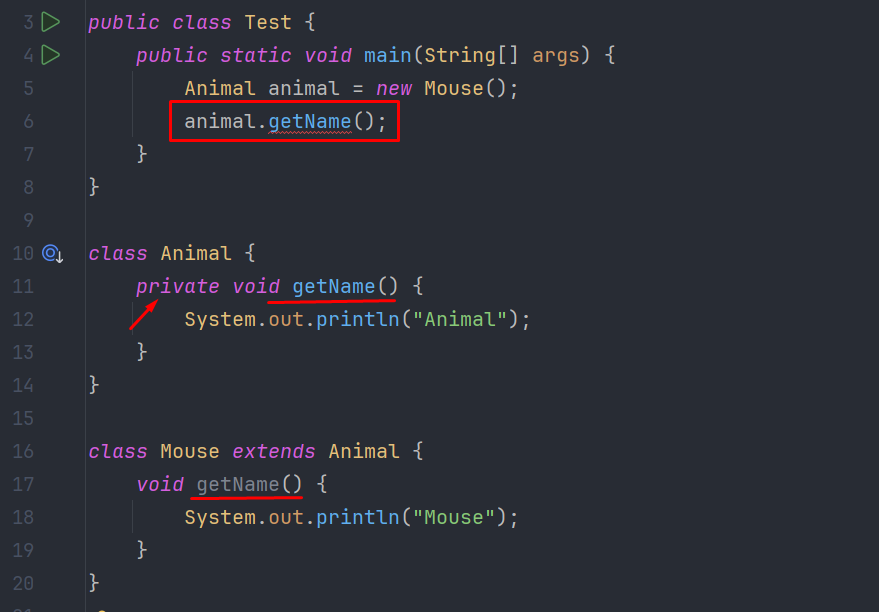
Pastdagi misolda ham Mouse mouse=new Mouse(); ifodadagi mouse o’zgaruvchisi compile time binding xususiyatiga ega bo’lib, abc() methodni chaqirganimizda, parametriga Mouse ni qabul qiladigan 14-qatordagi abc() methodni chaqirdi.



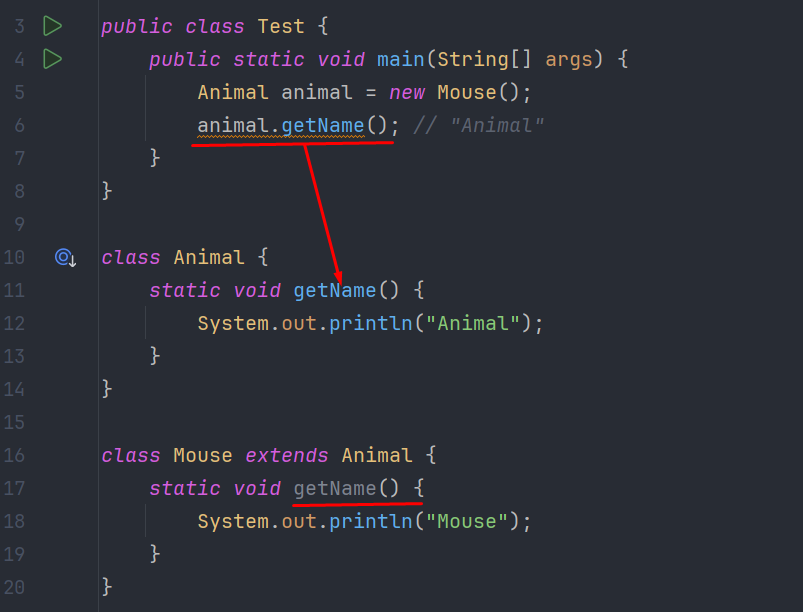
Pastdagi misolda ko’rish mumkinki, hamma methodlar qaysiki final, prive va static bo’lmagan methodlarni hammasi runtime binding hisoblanadi. Runtime bindingda esa qaysi objectga reference qilayotgan bo’lsa, o’sha objectdagi methodlar chaiqrialdi. Masalan, pastda 8-qatorda animal o’zgaruvchisini type Animal bo’lgani bilan, lekin new Mouse(); qilib Mouse objectiga reference qilyapti. Shuning uchun bu yerda animal.getName(); deb chaqirsak, 21-qatordagi Mouse classidagi getName(); methodini chaqiradi:



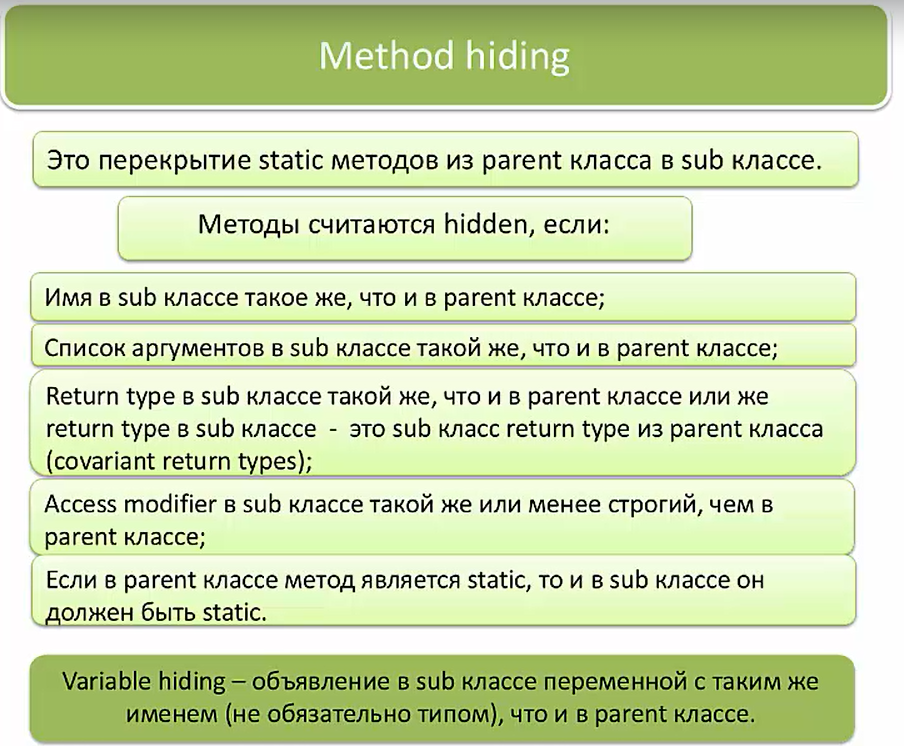
Pastdagi misoldan ko’rish mumkinki, Animal classidagi getName() methodi private bo’lib, undan meros olgan Mouse classidagi getName() methodi esa override bo’lmayapti aslida. Chunki Mouse classidagi getName() methodi aslida yangidan yaratilyapti va Animal classidagi getName() methodi private bo’lgani uchundir. Demak bu yerda overriding holati ketmayapti. Endi 6-qatordagi animal.getName(); ni ko’radigan bo’lsak, xatolik berganini ko’rishimiz mumkin. Sababi bu yerda compile time binding ketyapti, chunki private methodlar compile time binding hisoblanadi. Compile time binding da esa o’zgaruvchini(animal) methodlari chaqiriladi. Bizni holatda method bu Animal classidagi getName() methodidir 11-qatorda. getName() methodi private bo’lgani uchun bu methodni chaqira olmaydi. Shuning uchun xatolik beradi:



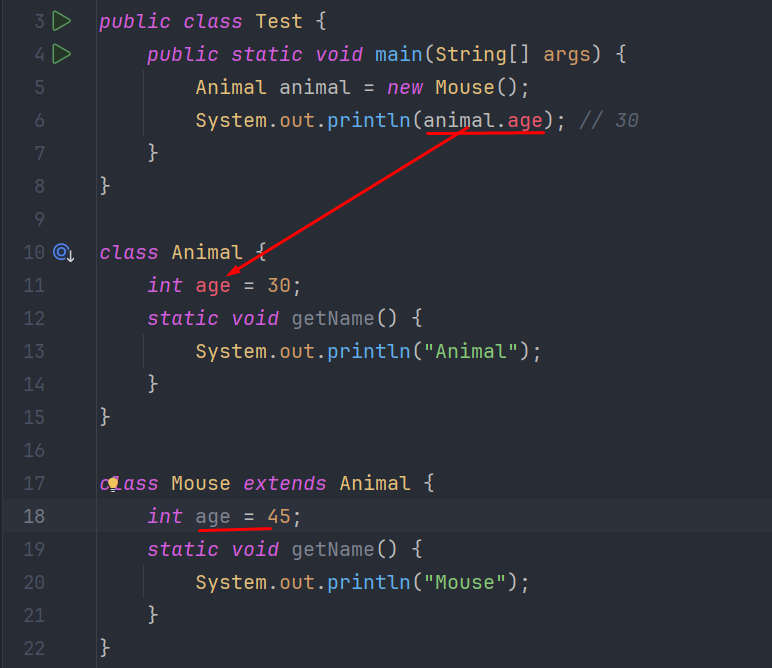
Yuqoridagi misol kabi, static methodlar overriding bo’lmaydi. Static methodlar Compile time binding hisoblanadi. Shuning uchun 6-qatordagi animal o’zgaruvchisi Animal classdan olingan objectga reference qilyapti va Animal classini getName() methodini chaqiryapti. Bu holar **METHOD HIDING** deyiladi:



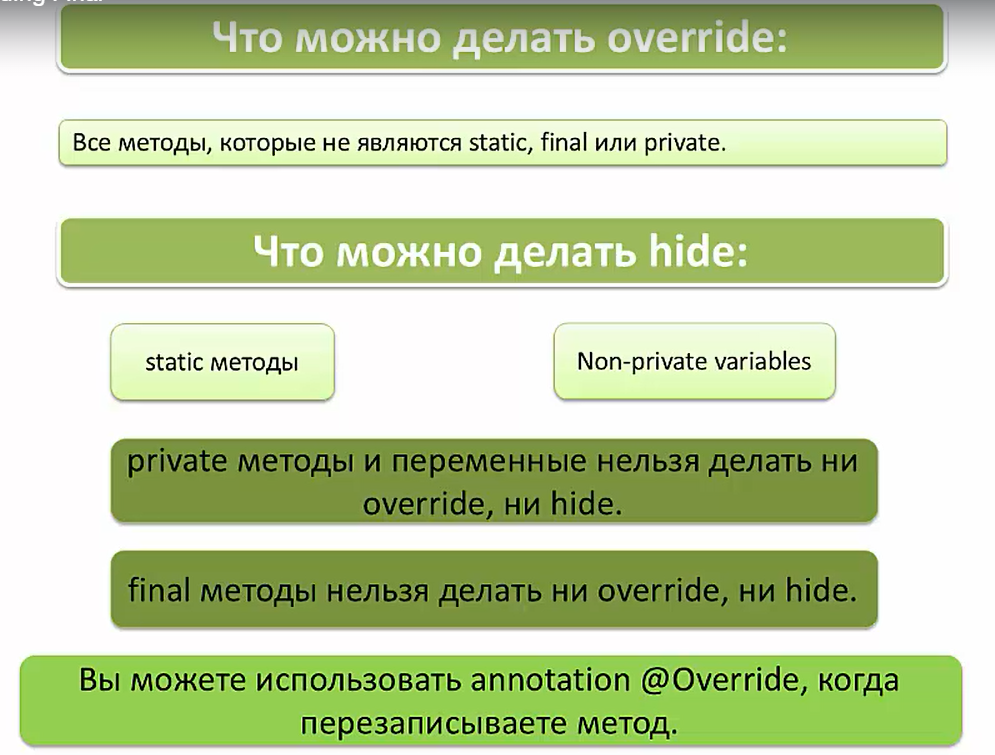
**METHOD HIDING**



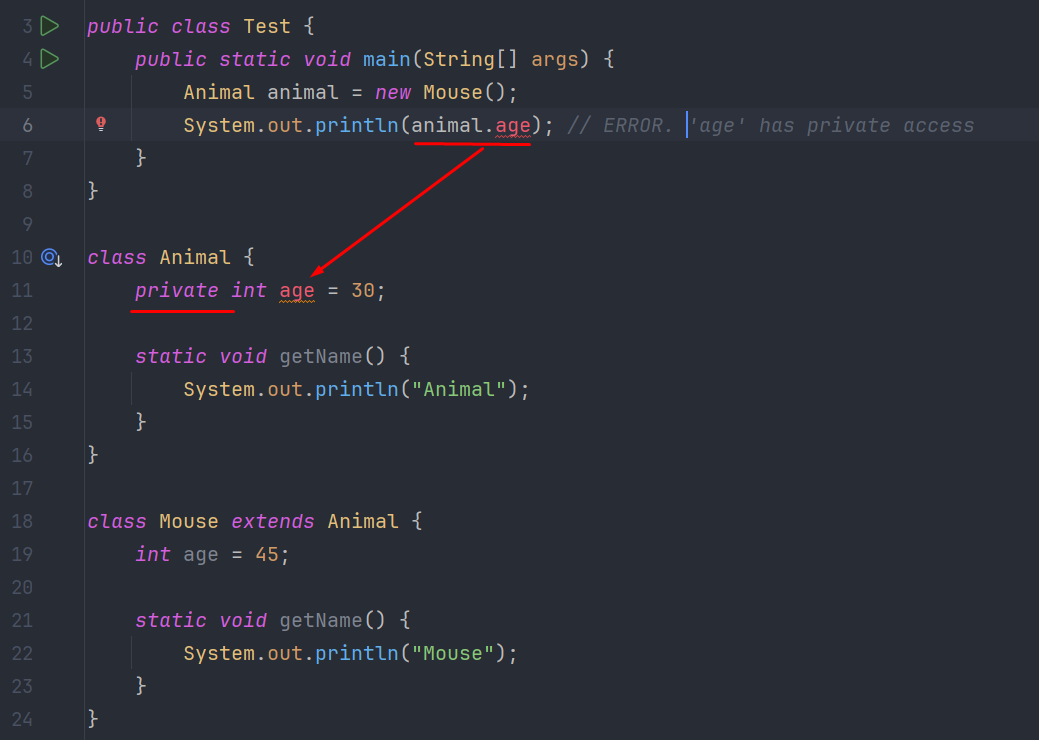
Variable hiding – bu ota classda qanday nomli o’zgaruvchi e’lon qilingan bo’lsa, bola classda ham xuddi shunday nomli o’zgaruvchi e’lon qilinishdir. Bunda o’zgaruvchilarni typelari har xil bo’lishi mumkin, shart emas ikkaladida ham int bo’lishi. Masalan bittasida int ikkinchisida esa String ham bo’lishi mumkin. Variable hiding ham compile time binding bo’lib, bunda object(animal) qaysi classga tegishli bo’lsa, o’sha class(Animal)dagi o’zgaruvchini chaqiradi:



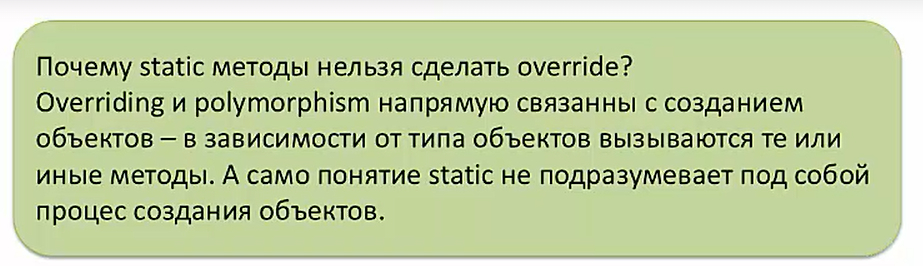
Pastdagi yozuvlardan ko’rish mumkinki, faqatgina non-private ya’ni private bo’lmagan o’zgaruvchilar hide qilish mukin. Chunki private o’zgaruvchilar meros bo’lib o’tmaydi.

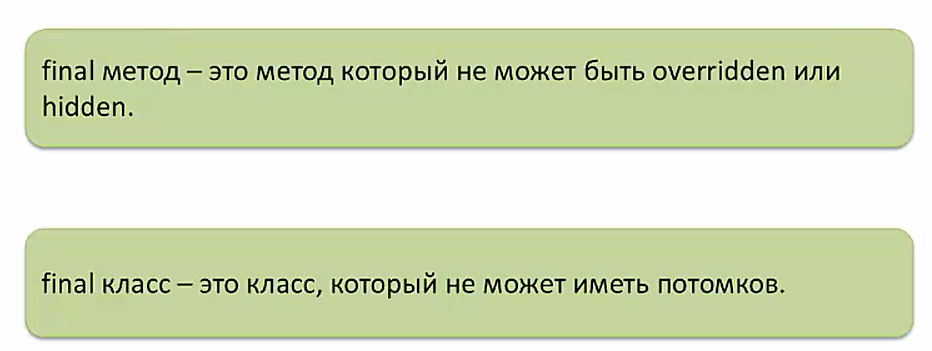


Pastda yuqoridagi narsani isboti berilgan, animal o’zgaruvchisi compile time binding bo’lgani uchun, animal objecti faqat Animal classiga tegishli bo’lib, faqatgina Animal classidagi o’zgaruvchilarnigina chaqira oladi. Demak 11-qatordagi age ni chaqira oladi, lekin bu o’zgaruvchi private bo’lgani uchun uni animal.age qilib chaqirishimiz xatolikka olib keladi:

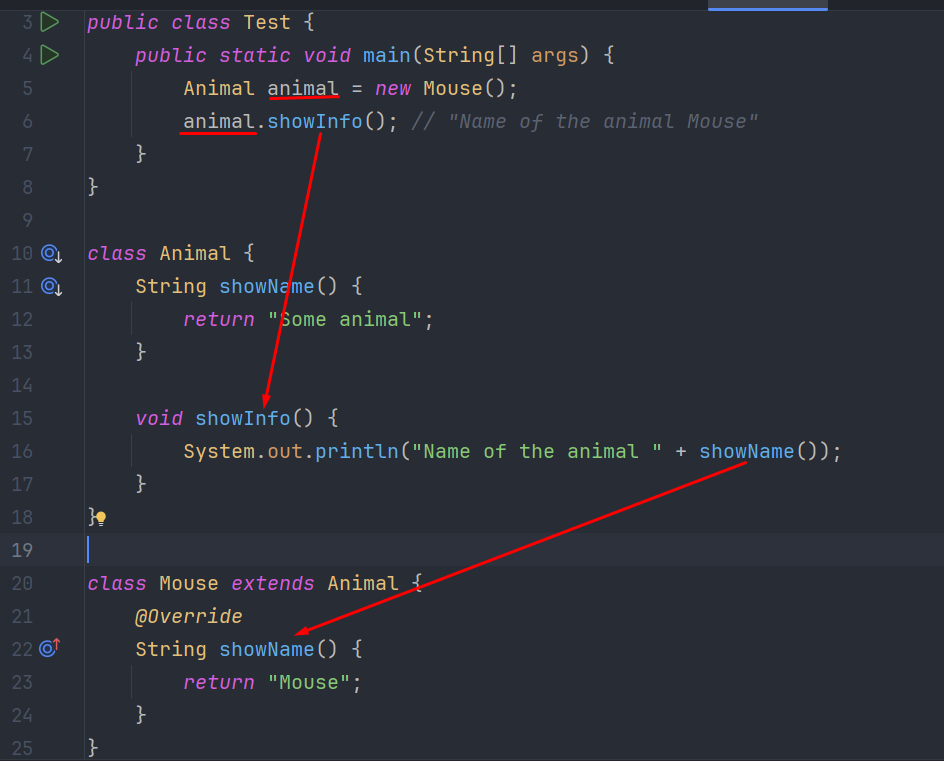


Pastda nega static methodlar override qilinamasligi haqida yozilgan:

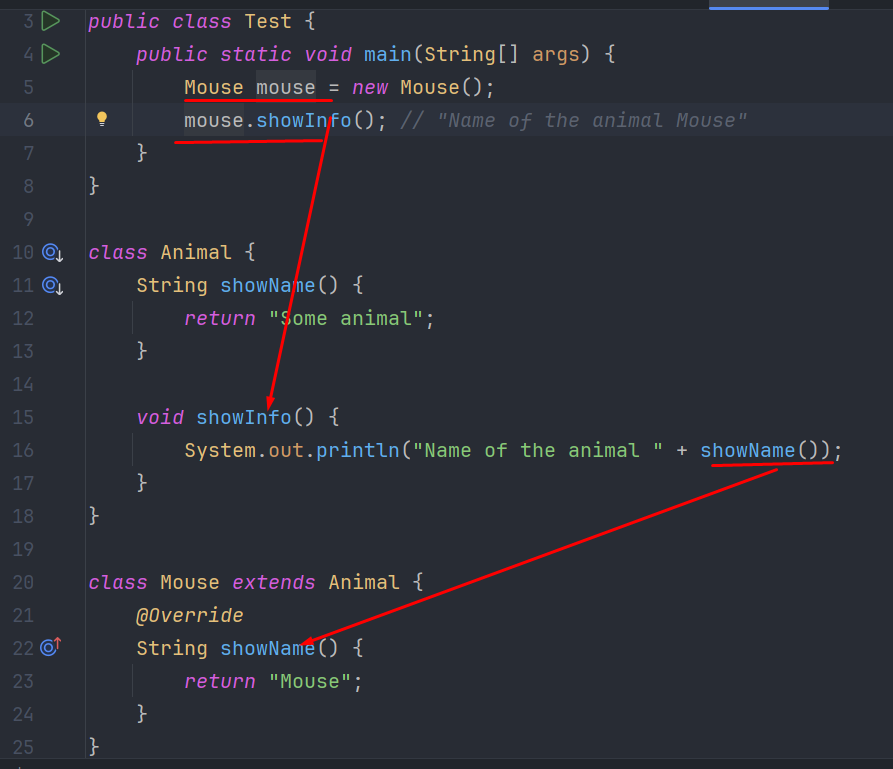




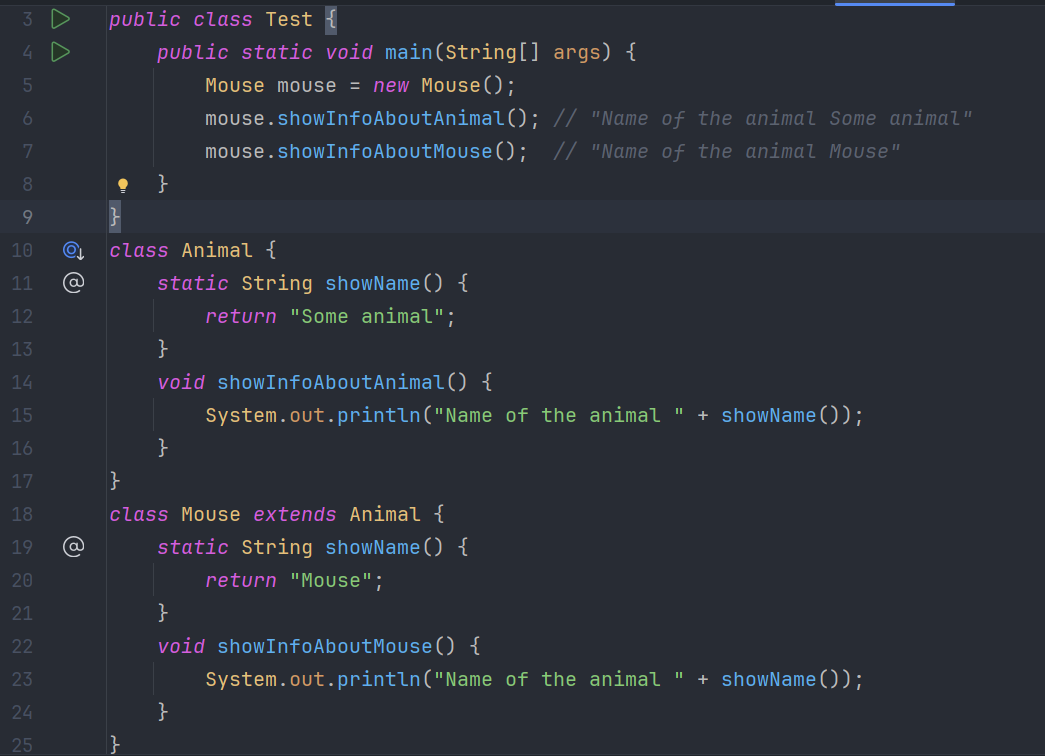
Pastda yana bir misol ko’ramiz. 6-qatordagi animal.showInfo() deb chaqirilganda, 16-qatordagi showName() methodi qaysi classdagi showName() ni chaqiradi, chunki Animal va Mouse classlarida ham showName() methodlari bor. Bu yerda run time binding ketyapti. Chunki showName() method na static, na private va na final. Demak bu yerda runtime binding ketyapti. Runtime bindingda esa esa Animal animal=new Mouse(); ifodadagi animal o’zgaruvchidagi showName() methodni emas, balki runtimeda aniqlangan new Mouse(); objectidagi showName() methodni chaqiradi. Shuning Mouse classdagi showName() method chaqirilyapti:



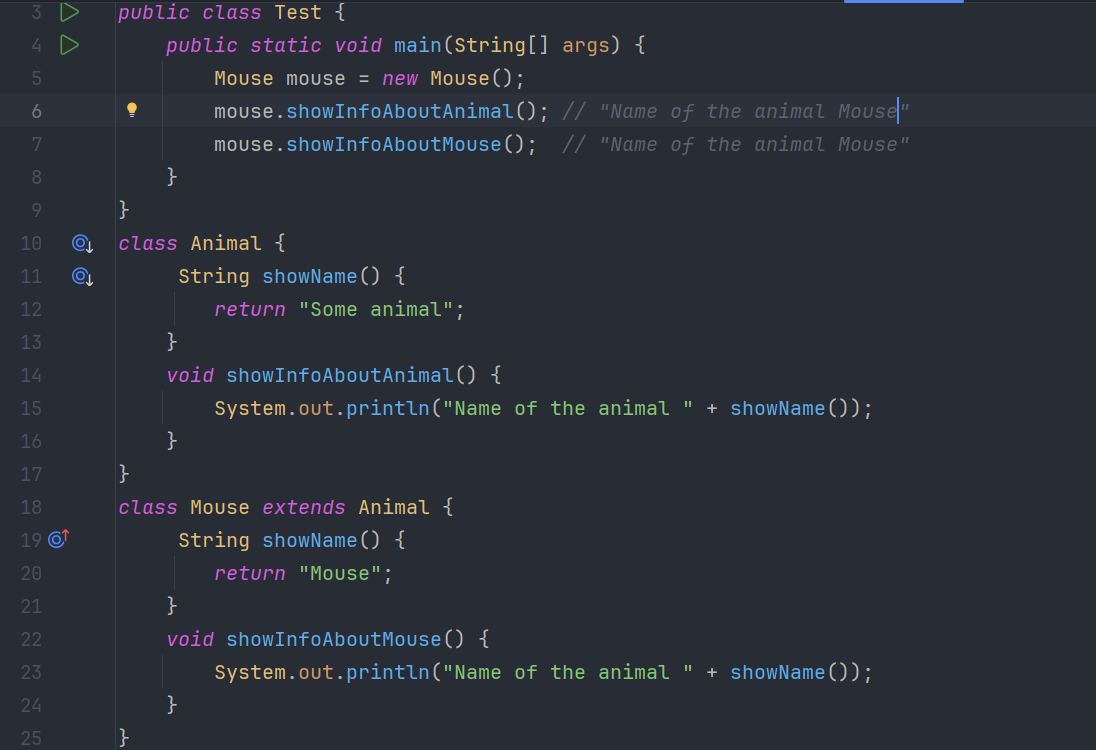
Pastdagi misol yuqoridagi misolni o’zi faqat 5-qatorda ozgina o’zgartirish kiritilgan. Misoldan ko’rish mumkinki, mouse.showInfo(); deyilganda 15-qatordagi showInfo() chaqiriladi. Lekin uni ichidagi showName() method bizda 2 ta classda bor, bunday holatda showName() methodlarini qarab ko’ramiz, ya’ni ular na static, na private van a final keyword bilan yozilgan. Demak ular compile time bindingga mos kelmaydi, balki runtime bindingga mos keladi. Runtime bindingda esa object yaratilayotgan objectdan ya’ni new Mouse(); dan olinayotgan objectdagi showName() methodni chaqiradi, ya’ni 22-qatordagi override bo’lgan showName() ni chaqiradi:



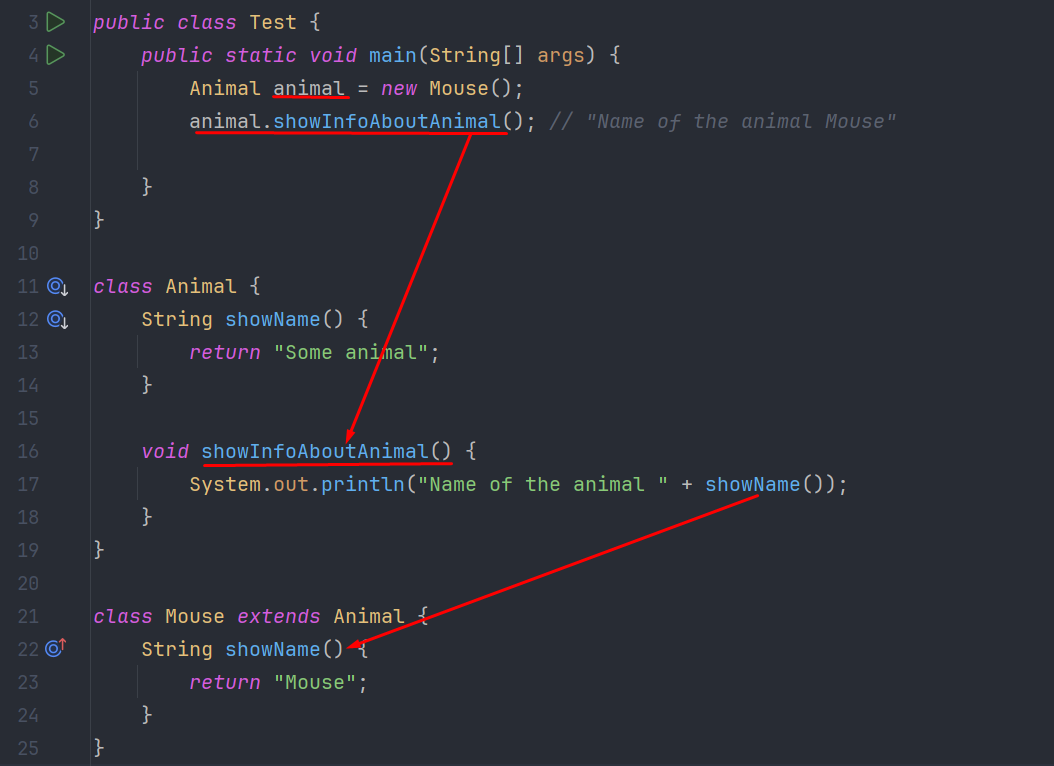
Pastdagi misoldan ko’rish mumkinki, Animal va Mouse classlarida showName() methodlari static keywordi bilan e’lon qilingan, shuning uchun bu yerda compile time binding ketyapti. Compile time bindingda 5-qatordagi Mouse mouse = new Mouse(); ifodadan mouse o’zgaruvchisiga tegishli bo’lgan methodlardni chiqaradi. mouse.showInfoAboutAnimal(); ifoda 14-qatordagi methodni chaqiradi, o’z navbatida 11-qatordagi showName() method ham shu Animal classdagisi chaqiriladi. Sababi showName() static bilan e’lon qilingan,demak compile time binding bo’ladi. Compile time bindingda Mouse classdagi showName() ni emas, balki Animal classdagi showName() ni chaqiradi,chunki showName() method override bolmaydi static bo’lgani uchun. Endi mouse.showInfoAboutMouse(); desak u holda 22-qatordagi method chaqiriladi. O’z navbatida 19-qatordagi showName() chaqiriladi:

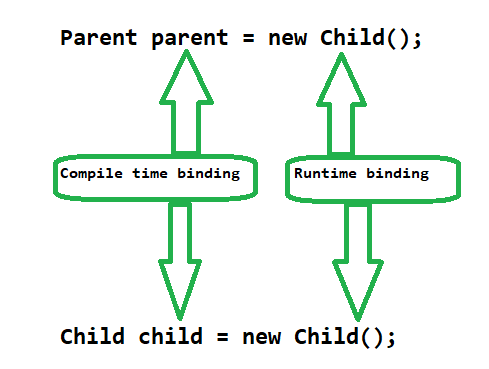


Bu misolda yuqoridagi misolni o’zi, faqat showName() lardan static kalit so’zini o’chirib chiqqanmiz. Output ham o’zgaradi. 6-7-qatorlardagi methodlarni chaqirganimiz bilan ham 2 la holatda Mouse classdagi showName() method chaqiriladi. Sababi showName() methodlari na static, na final, na private bo’lgani uchun runtime binding jarayoni ketadi. Runtime bindingda esa 5-qatordagi new Mouse() objectiga tegishli bo’lgan showName() methodi chaqiriladi. Bu metod bizda 19-qatordagi showName() methodidir. Shuning uchun bizga “Mouse” yozuvi qaytarildi:

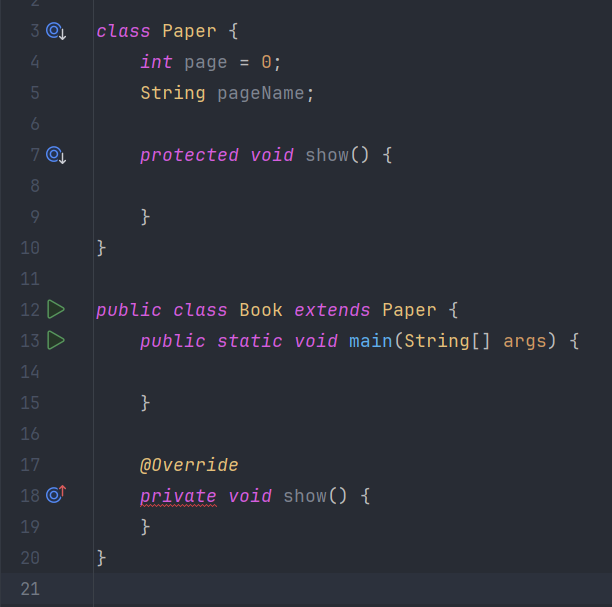


Pastdagi misolda showName() methodlari na static, na final va na private bo’lgani uchun runtime binding bo’ladi. Runtime binding da 5-qatordagi new Mouse() dan olingan objectni showName() ni chaqiradi. Bu showName() methodi Mouse classidagisidir:









Agar ota class dagi methodni parametrida boshqa type, bola classni methodini parametrida boshqa type bo’lsa, ular overriding ga kirmaydi. Pastdagi **eat()** methodlari ham overriding qoidasiga mos kelmayapti. Overriding qoidasiga ko’ra methodlarni parameterlari va ularni parameterlarini typelari ham absolyutno bir xil bo’lishi shart. Bizda esa biri **double**, biri **int** typedadir:

