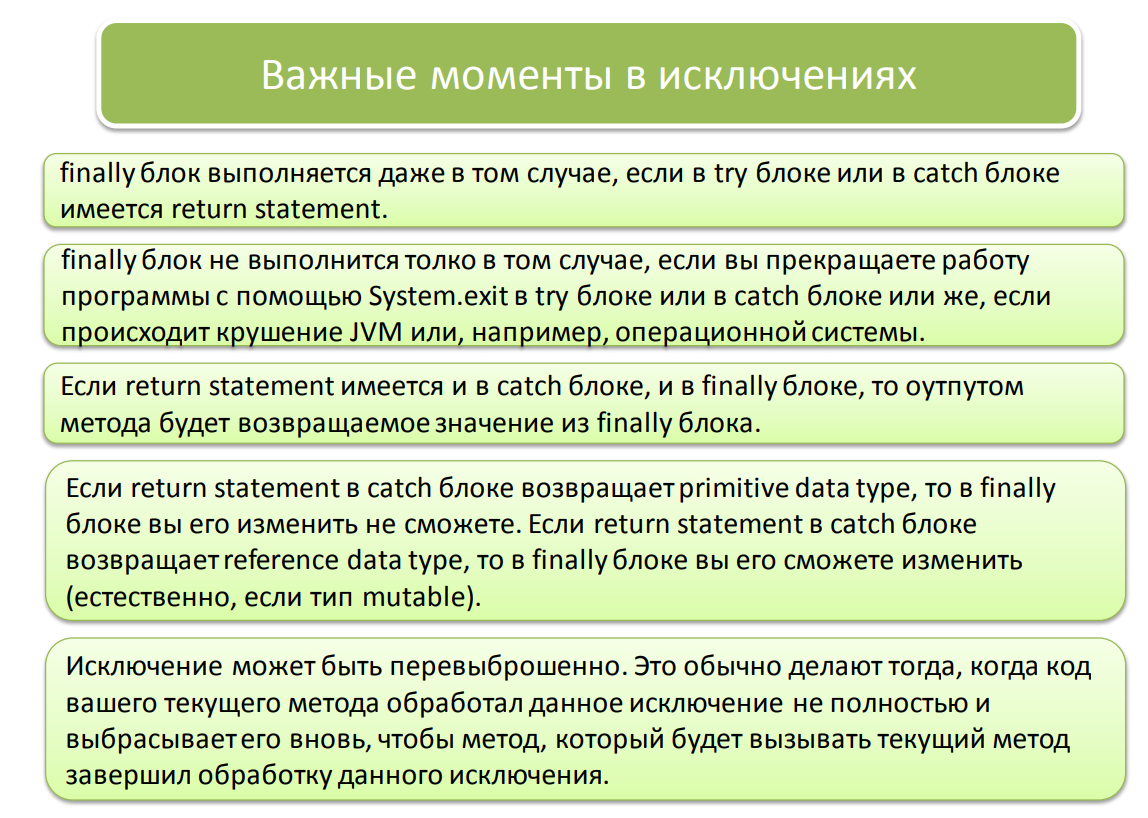
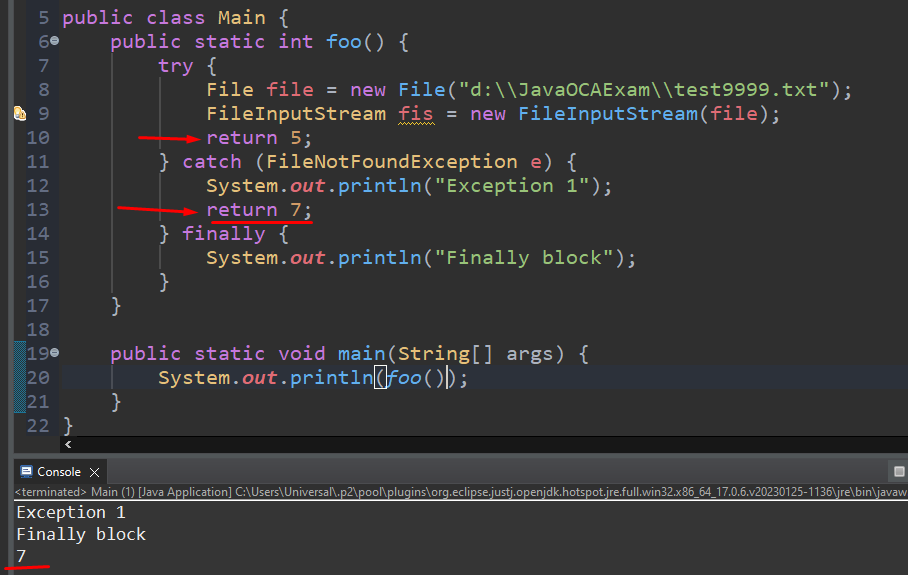
Keling endi finally keyword haqida gaplashamiz. Finally block doim bajariladi, error bo’ladi yoki error bo’lmaydimi farqi yo’q.



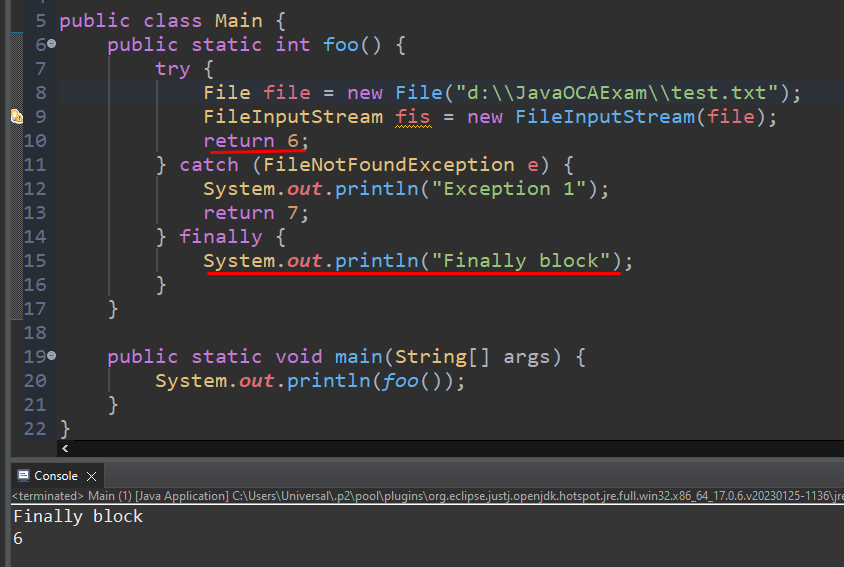
Undan oldin return keywordini ishlatganimizda nimalar bo’lishini ko’raylik. return ni ishlatayotganda e’tiborli bo’lish kerak. Agar methodimiz return type li bo’lsa, albatta return keywordini ham try blockda, ham catch blockda yozish shart, aks holda xatolik beradi. Nega ikkala blockda yozish shart? Sababi agar kodimizda hech qanday xatolik bo’lmasa catch block hech qachon ishlamaydi. Bunday vaziyatda return faqat catch blockda yozilgan bo’lsa, bu methoddan hech narsa return bo’lib qaytmaydi. Tabiiyki return bo’lmagach xatolik kelib chiqadi. Yoki aksi agar kodimizda qandaydir xatolik bo’lsayu, lekin return faqat try blockda yozilgan bo’lsa, kodimiz trydan avtomatik chiqib ketib catchga tushadi. Lekin catchda return yozilmagan. Bunday vaziyatda ham methoddan return bo’lib hech narsa qaytmaydi. Nima bo’lganda ham har doim return qaytishi shart va bu return ham try blockda ham catch blockda yozilishi shart. Pastdagi misolda return ham try da ham catch da yozilgan. Error bo’lmasa trydagi blockdan return qaytadi, agar xatolik bo’lsa, catchdagi blockdan return qaytadi:



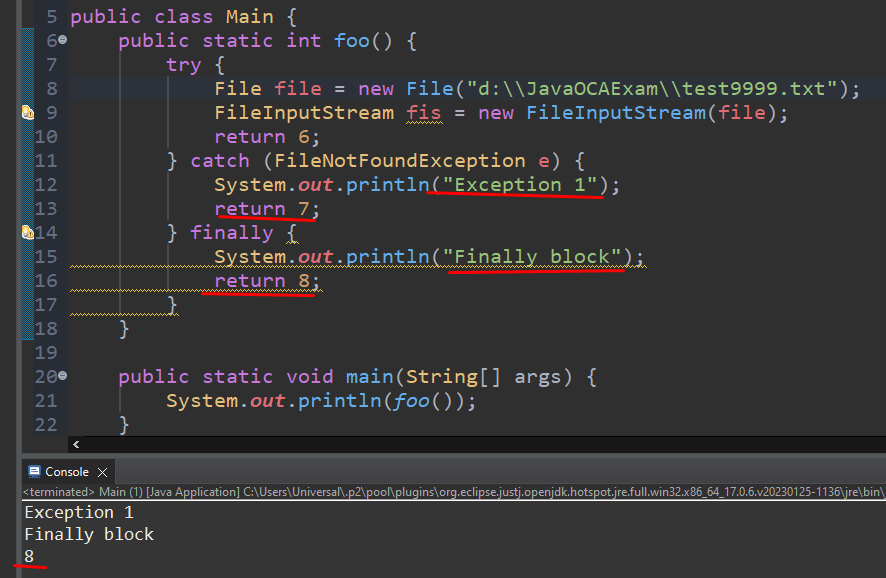
Agar try va catch da return yozilgan bo’lsa, u holda bu retrun doim finally block dan keyin bajariladi. Masalan, pastdagi misolda bizda test9999.txt filemiz yo’q, shuning uchun catch ga tushadi. Avval catchga kirib sout(“Exception 1”); ni chiqaradi, keyin esa return ni bajarmasdan, finally blockdagi sout(“Finally block”) ni bajaradi va yana qaytib kelib catch dagi return ni bajaradi:



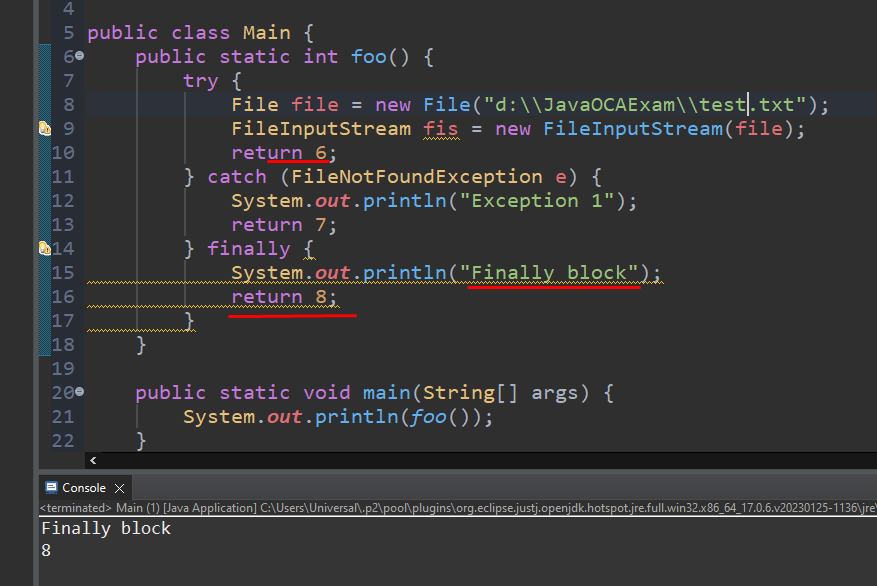
Yoki error yo’q bo’lgan holatni ko’ramiz. Kodimizni bajarib kelib, try blockda return ga kelganda return qilmasdan, birdan finally blockka o’tib ketadi, u yerdan sout(“Finally block”) ni bajarib, keyin try blockdagi return bajarib 6 ni qaytaradi:



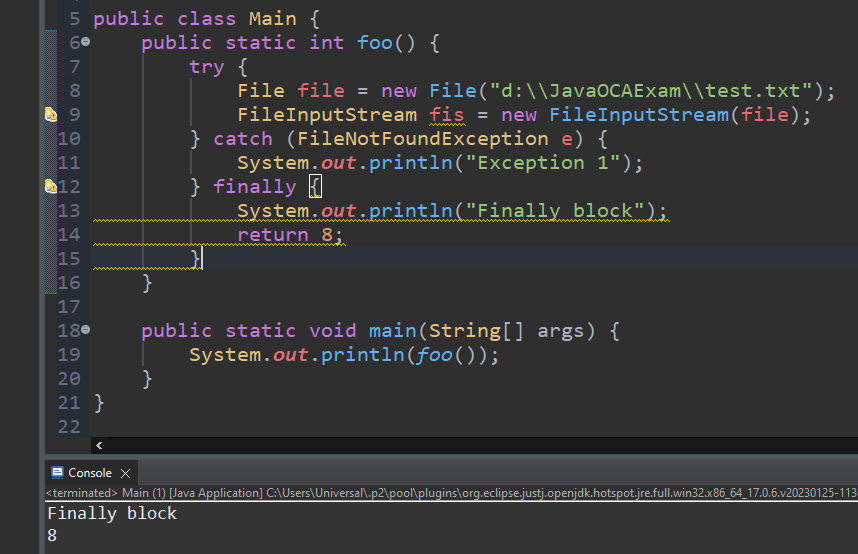
Agar catch blockda ham, finally blockda ham return bo’lsa, u holda faqat finally blockdan return qiymat qaytadi. Catch blockdagi esa e’tiborsiz qoldiriladi. Masalan pastda test9999.txt filemiz mavjud emas, demak kodda xatolik bo’lib, catch blockka tushadi. Lekin u yerdagi return 7; dan hech qanday qiymat qaytmasdan, finally blockdagi return 8; dan qiymat qaytadi:



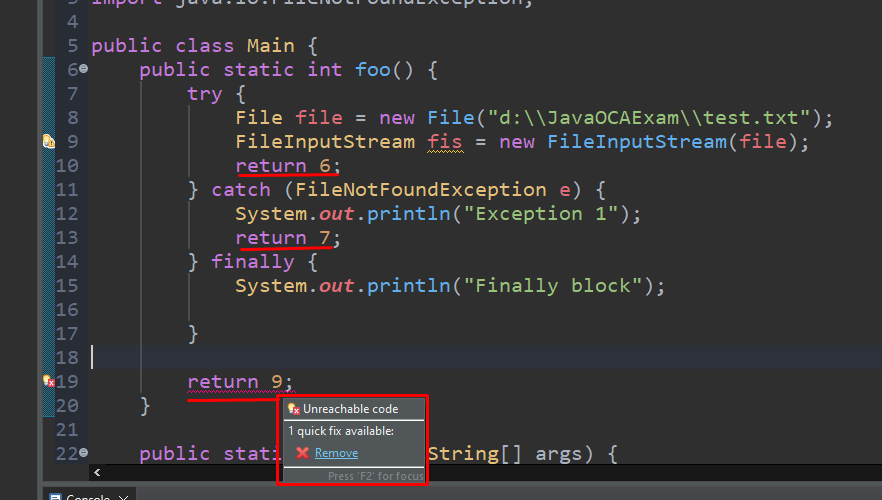
Agar kodimizda hech qanday xatolik bo’lmasa, u holda try blockdagi kod ishlab, return 6; e’tiborsiz qoldiriladi va finally blockdagi return 8; qismidagi return ishlaydi:



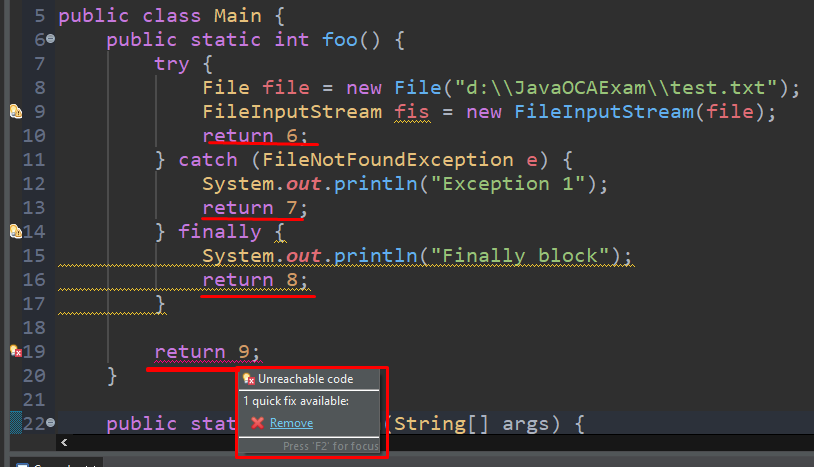
Agar try blockda ham catch blockda ham return bo’lmasdan, lekin faqat finally blockda return bo’lsa, u holda hech qanday xatolik bermaydi. Chunki finally block nima bo’lgan taqdirda ham ishlaydi va albatta return qaytadi:



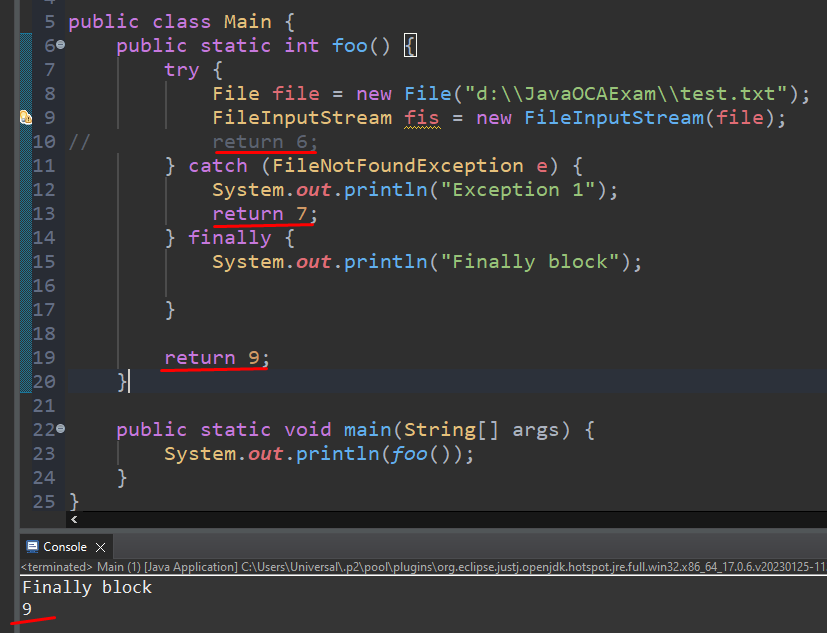
Agar try blockda ham catch blockda ham return bo’lsayu, lekin finally blockda return bo’lmasa, u holda shu blocklardan tashqarida return ni yozsak, bu return unreachable bo’ladi va xatolik beradi, sababi nima bo’lgan taqdirda ham shu 2 ta blockdan biridan, ya’ni yo try dan yo catchdan baribir return qaytadi. Shuning uchun 19-qatordagi return gacha yetib bormaydi va bu return unreachable code bo’ladi va xatolik beradi:



Agar 3 la blockda ham return bo’lsa, u holda 19-qatorda yozilgan return baribir unreachable code bo’lib, bajarilmaydi va xatolik beradi. Chunki bu return gacha yetib kelmaydi:



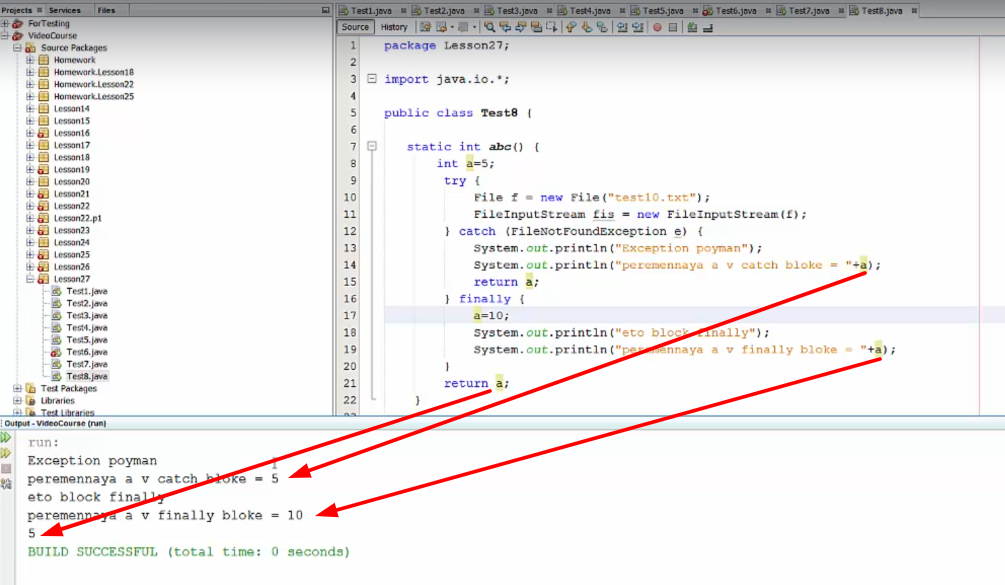
Lekin agar try yoki catchdan bittasida return bo’lib, finally da return bo’lmasa, u holda bu blocklardan tashqarida yozilgan return dan qiymat qaytadi. Sababi try yoki catchdan bittasida return bo’lsa, shu return qaytmasligi mumkin. Lekin methodimiz albatta return qaytarishi kerak. Shuning uchun blocklardan tashqarida yozilgan return ishlaydi:

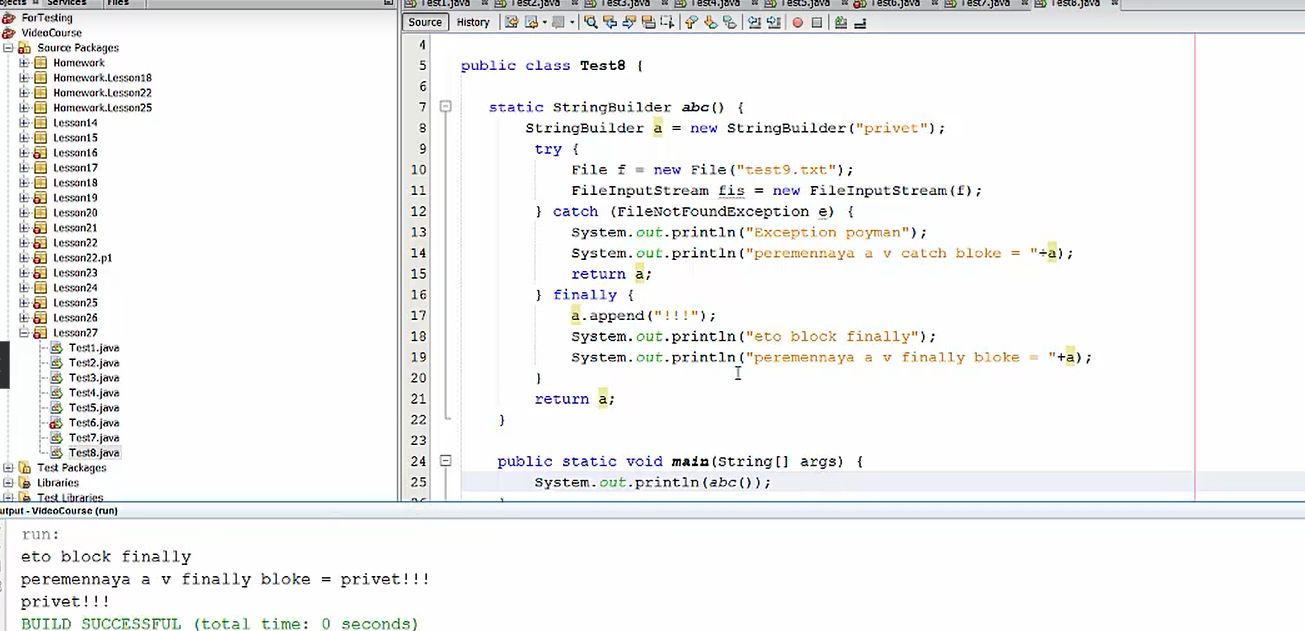


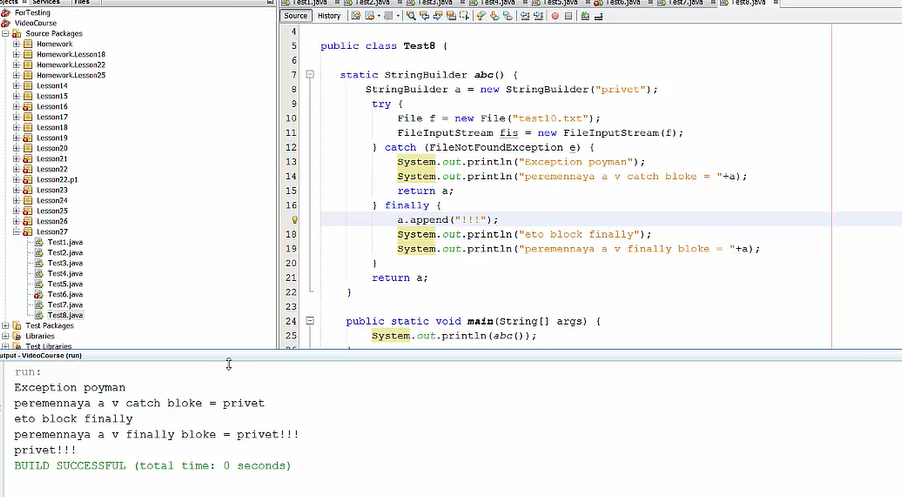
Demak, 3 ta blocklardan biridan albatta return qaytadigan bo’lsa, u holda bu blocklardan tashqarida yozilgan return ishlamaydi, ya’ni ungacha yetib bormaydi unreachable code bo’ladi. Lekin shu 3 ta blocklardan biridan return qaytmaslik holati bo’lsa, bunday holatda shu blocklardan tashqaridagi return ishlaydi xatolik bermasdan.

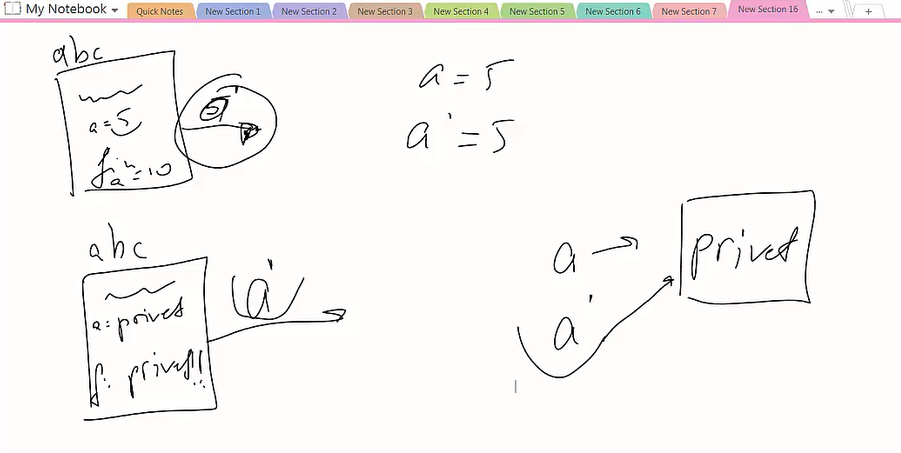


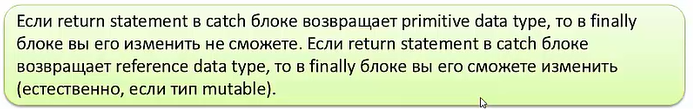
Pastdagidan a=5 qaytadi: 12:40



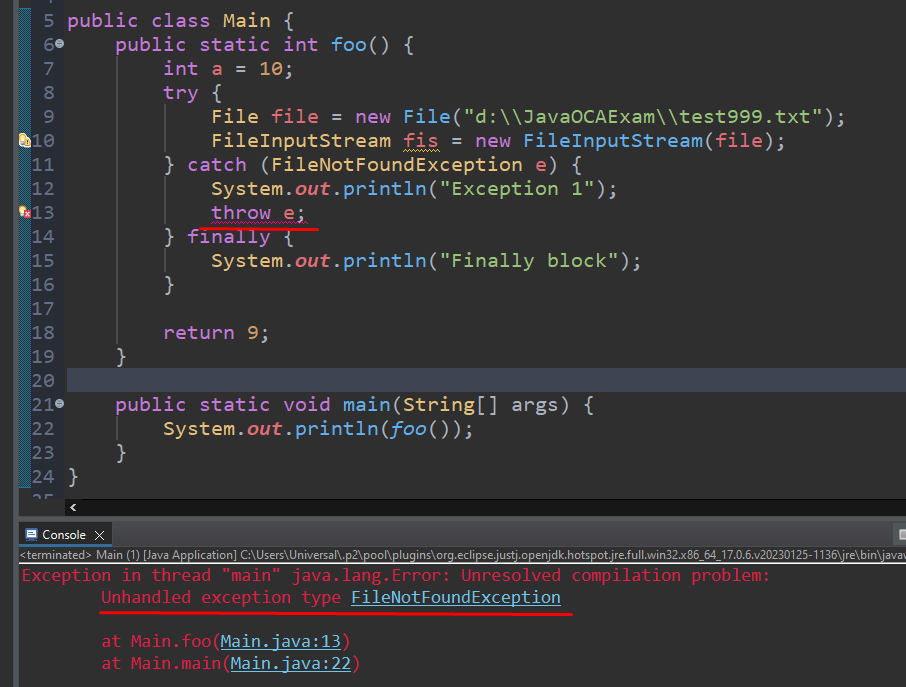




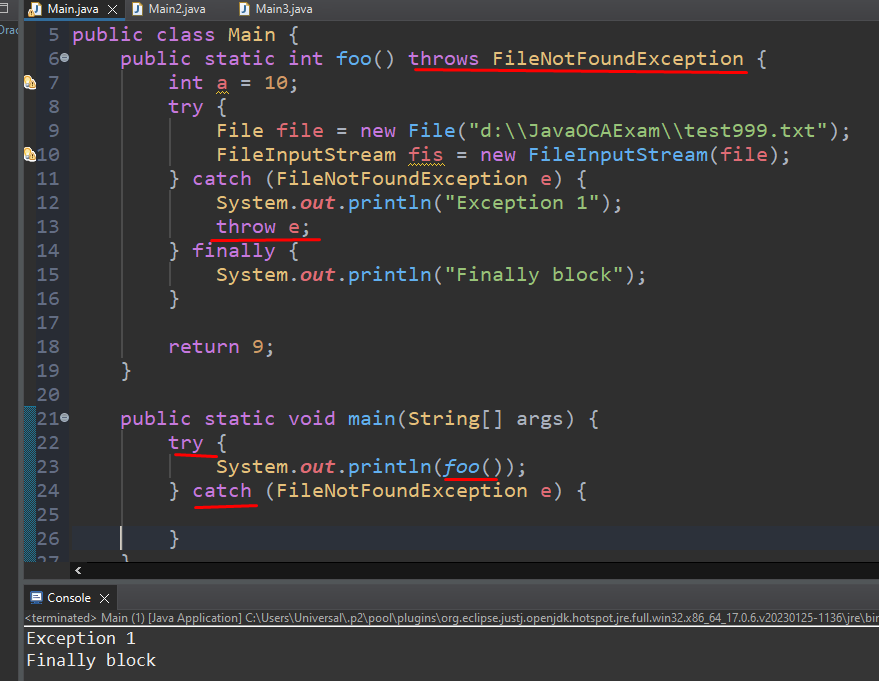




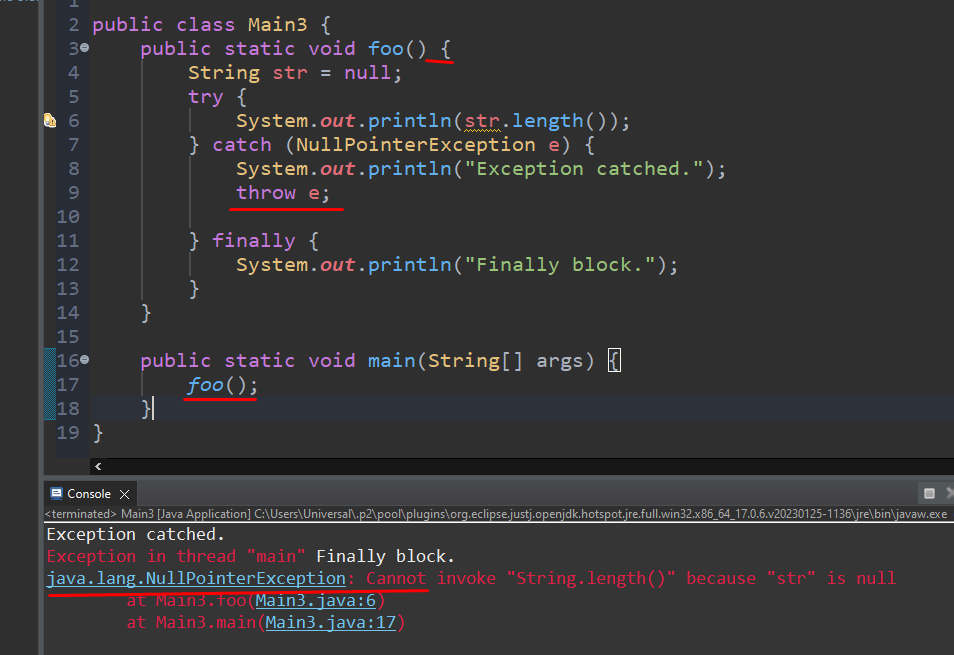
Agar catch blockda exception ni qayta ishlay olishlikni iloji bo’lmasa, u holda qayta exception otish(re-throw) qilishimiz mumkin. Pastda biz 13-qatorda re-throw qilyapmiz FileNotFoundException ni. E’tibor bergan bo’lsangiz xatolik chiqyapti, try-catch ni ichiga yozilgan bo’lsa hamki, error beryapti. Sababi bu try-catch lar 10-qatordan qaytgan exceptionini qayta ishlash uchun yozilgan, re-throwing uchun emas. Shuning uchun xatolik kelib chiqyapti:



Endi bu xatolikni to’g’irlash uchun esa, biz shu foo() m.ni signaturesiga qaysi exceptionni re-throwing qilayotgan bo’lsak, o’sha exception ni yozishimiz kerak. Bizing misolda biz throw e; dagi e bu FileNotFoundException bo’lgani uchun xuddi shu exceptionni foo() m.ni signaturesiga yozyapmiz. Keyin esa foo() m.ni main() m.da chaqiramiz va albatta re-throwing dan kelayotgan exceptionni main() m.da try-catch orqali qayta ishlashimiz kerak yoki main() m.ni signaturesiga FileNotFoundException yozishimiz kerak. Biz pastda try-catch orqali qayta ishlayapmiz:



Agar unchecked(runtime) exceptionlarda re-throwing tashlaganimiz bilan, bu exception uchun methodni sigaturesiga re-throwing bo’ladugan exceptionni yozish shart emas va re-throwing bo’lgan exceptionni qayta ishlash ham shart emas. Bu holat yuqoridagi holatdan farq qiladi. Pastda 9-qatorda biz re-throwing qilyapmiz, lekin shu foo() m.ni signaturesiga exceptionni yozmayapmiz va main() m.da foo() m.ni chaqirganimiz bilan ham uni qayta ishlash shart emas, baribir compile timeda xatolik chiqmaydi, balki runtime da xatolik chiqadi. Shuning uchun pastdagi misolda compile timeda xatolik bermaydi, lekin runtime da console da xatolik ko’ramiz xolos:

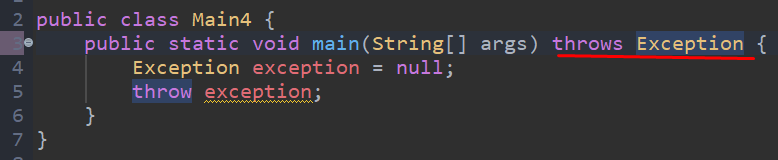


Demak, unchecked(runtime) exceptionlarda re-throwing tashlaganimiz bilan ularni qayta ishlash yoki method signaturesiga exceptionlarni yozish shart emas.

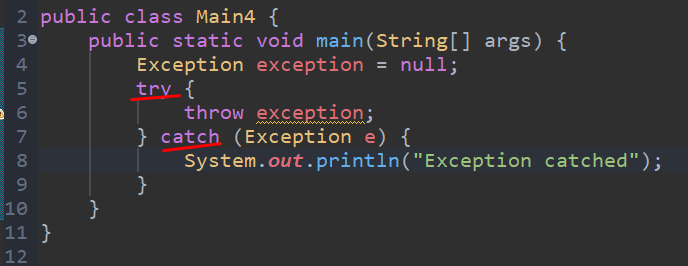
Agar biz pastdagi misol kabi exception o’zgaruvchini olsak va uni throw qilsak, u holda error beradi. Sababi bu exception ni qayta ishlash kerak yoki method signaturega exception yozish kerak:



Buni yo’qotish uchun esa method signaturega exception yozish kerak:



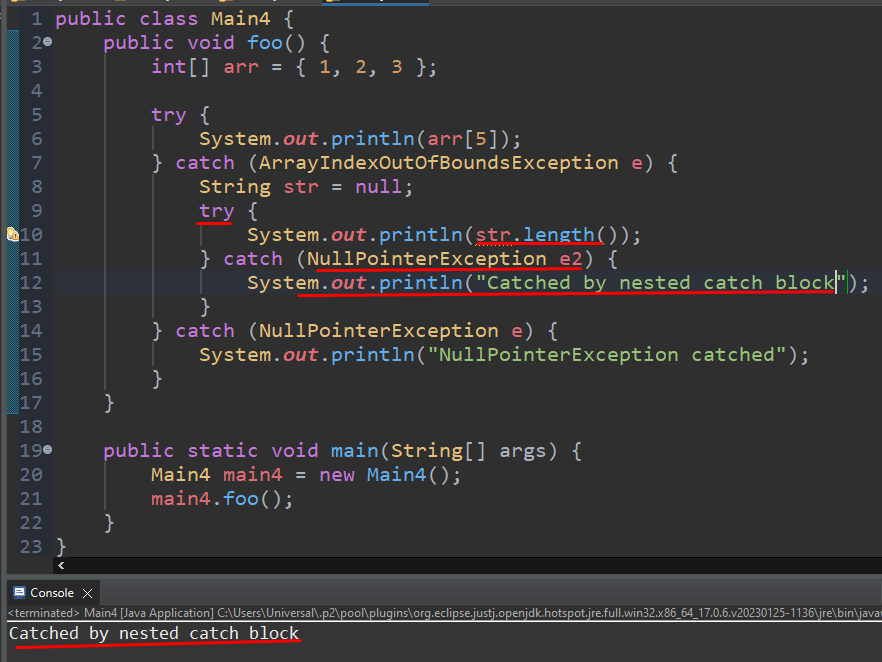
Yoki try-catch bilan ham qayta ishlash mumkin.



Bitta try uchun 2 ta catch yozilishi mumkin, buni bilamiz. Masalan pastdagi misolda try blockda 6-qatorda exception otiladi va bu exception 7-qatordagi catchda tutib olinadi. Lekin bu catch blockni ichida 9-qatorda yana exception otiladi. Bu exception 10-qatordagi catch tomondan tutib olinmaydi. Sababi bu catch block 7-qatordagi catch kabi faqat 5-qatordagi try blockdan chiqqan exceptionni tutib oladi. Shuning uchun bu 2 ta catch block faqat 5-qatordagi catch blockdan chiqqan exceptionni tutish uchun mo’ljallangan. Shuning uchun pastda console da NullPointerException ni 10-qatordagi catch orqali tuta olmaganimiz uchun NullPointerException ni chiqardi:



Endi yuqoridagi misoldagi NullPointerException ni nested try-catch orqali tutib olishimiz kerak. Buning uchun 7-qatordagi catch blockda NullPointerException ni ushlaydigan yana boshqa bir try-catch block ochamiz va shu orqali str.length() dan chiqishi mumkin bo’lgan NullPointerException ni 11-qatordagi catch blockda tutamiz:



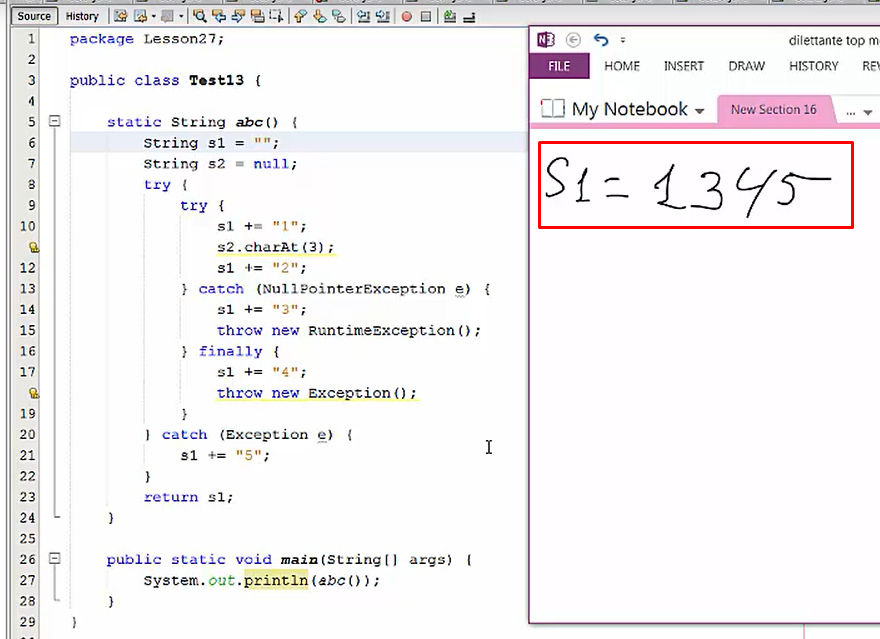
[42:00, 44:30]

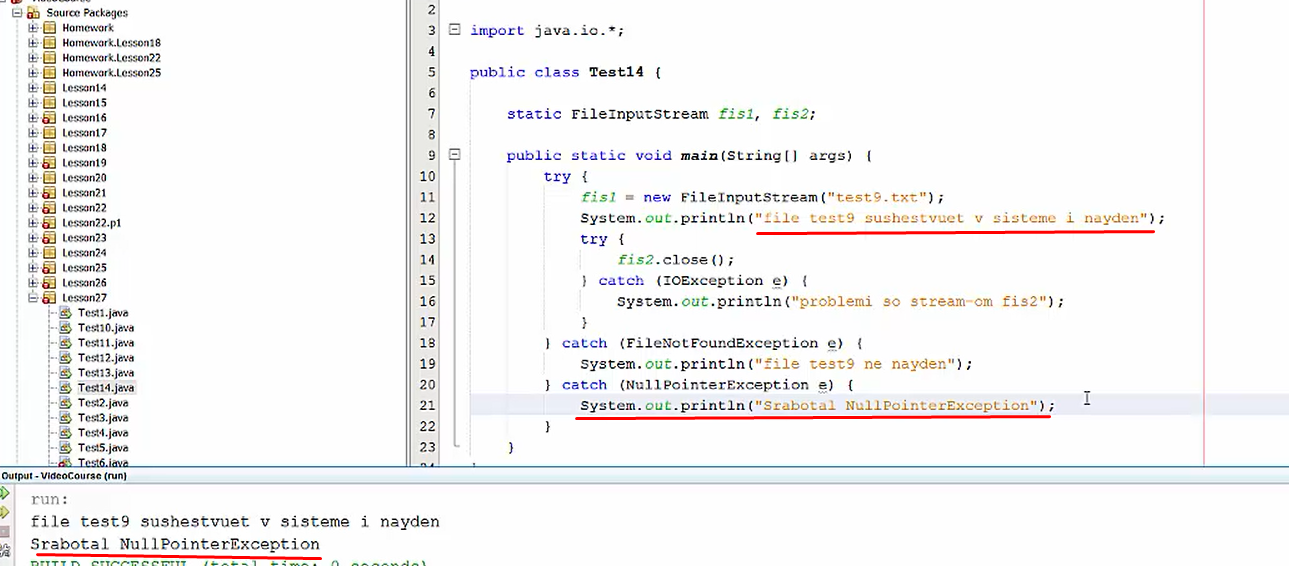
Exception VS Error

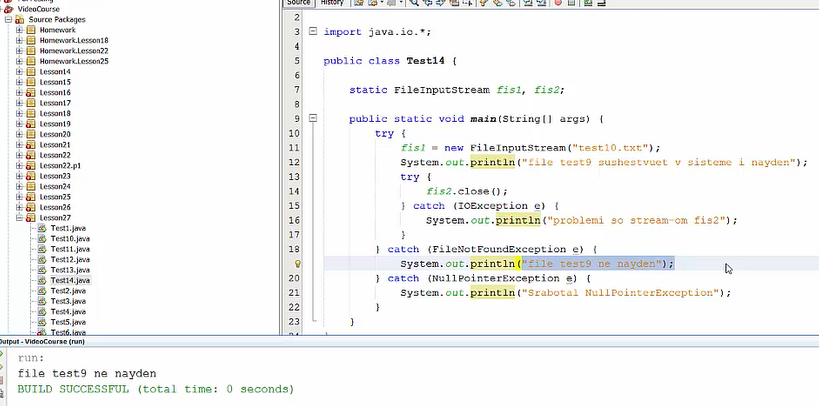
Exception(ex.) va Error(err.)ni nima farqi bor? Keling shu haqida gaplashaylik. Error bu dasturda nimadir xato ketganini bildirib, bu err.ni dasturchi tomonidan handle qilib bo’lmaydi, ya’ni try-catch ga olib bo’lmaydi. Xattoki err.ni try-catch ga olish juda katta hisoblanadi, bunday qilmaslik kerak. Keling err.ga misol ko’raylik. Deylik bizda dastur bor bo’lib, uni ishga tushirganimizda memorydan ko’p joyni olib qo’ydi, shu payti JVM bizga OutOfMemory degan err. beradi. Bu degani err.ni try-catch bilan handle qilganimiz bilan ham, sodir bo’lgan err.ni to’g’irlab bo’lmaydi. Chunki dasturni ishlashi uchun memory yetmayapti. Uni try-catch ga olib memoryni kattalashtira olmaymiz. Chunki memory cheklangan bo’ladi. Err.lar faqat runtime da sodir bo’ladi, xuddiki unchecked errorlarga o’xshab. StackOverflow error ham err. hisoblanadi. Bu err. recursive f-yani qayta va qayta chaqirishdan hosil bo’ladi.

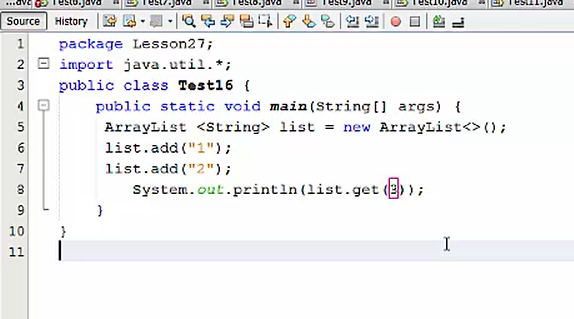
Exception esa dasturchi tomonidan noto’g’ri input kiritilganidan dasturimizda xatolik sodir bo’lishi mumkin. Ayniqsa RuntimeException(unchecked) lar bu dasturchini qo’pol xatoligi tufayli sodir bo’ladi. Masalan, deylik bizda 3 ta elementdan iborat massiv bor. Shu massivning 5-elementini olmoqchi bo’lsak, ArrayIndexOutOfBoundsException beradi. Bu yerda dasturchi bilishi kerak ediki, arrayda 3 ta element borligini, 5-elementni olmoqchi bo’lsak, xatolik berishi aniq. CompileTimeException(Checked exception)lar esa bu compile vaqtida dasturchini exceptionga tushishi mumkin bo’lgan kodlarni, try-catch ga olishga majburlaydi.

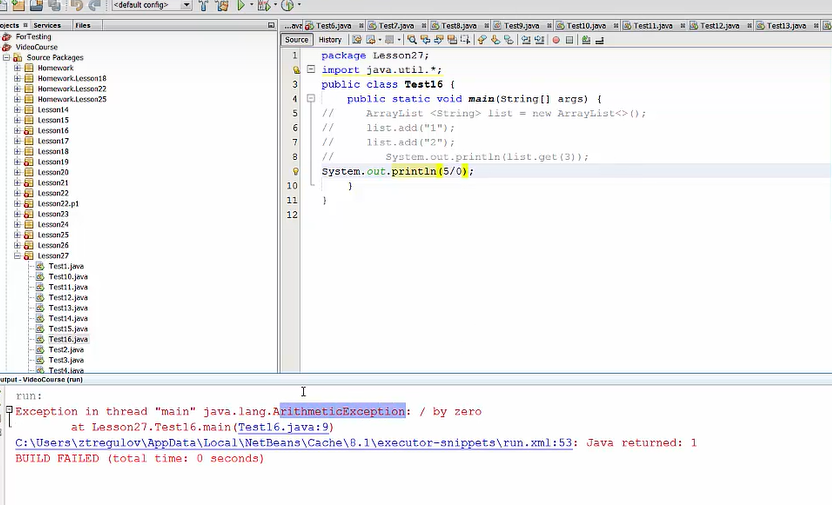
Yana bir boshqa misol ko’ramiz. Bizda s1 va s2 o’zgaruvchilar bor. S1+=”1” bo’ladi, keyin 11-qatorda NullPointerException tashlaydi va 12-qatorga o’tmasdan, 13-qatorda shu exceptinni catch ushlaydi s1+=”3” bo’ladi. 15-qatorda esa RuntimeException() tashlanadi. Bu exception esa tutib olinmaydi, chunki runtime(unchecked) exceptionlar tutib olinmaydi. Keyin 16-qatorga o’tib, finally blockka o’tadi, s+=”4” bo’ladi va u yerda 18-qatorda throw new Exception() otiladi. Bu exception 8-qatordagi try ga tegishli bo’lgani uchun, bu try ni catch i esa 20-qatorda joylashgan. Keyin s1+=”5” bo’ladi. Natija s1=”1345” bo’ladi:

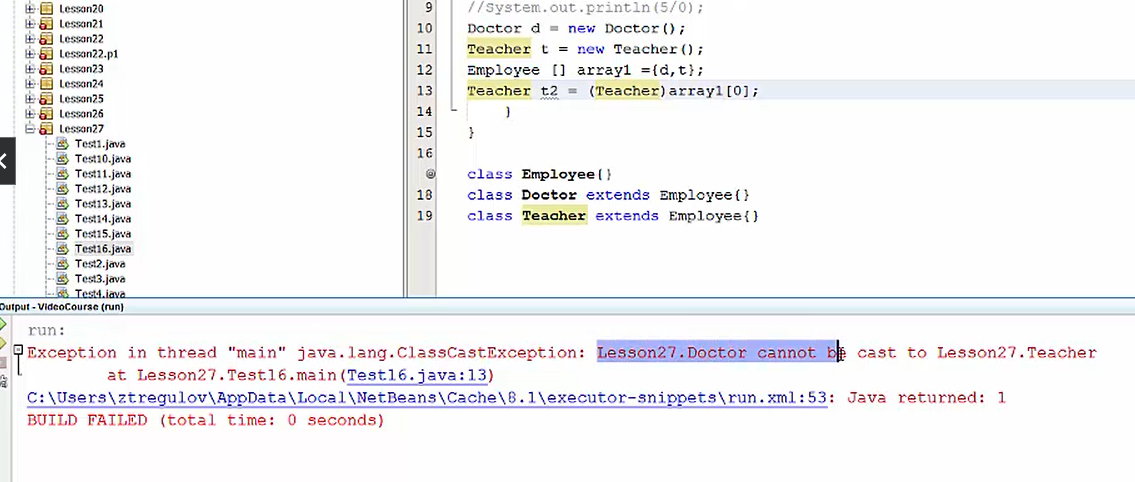


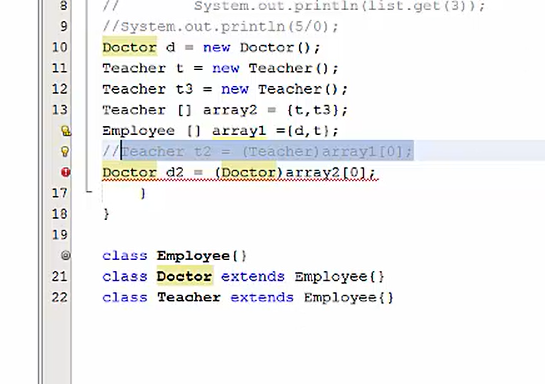


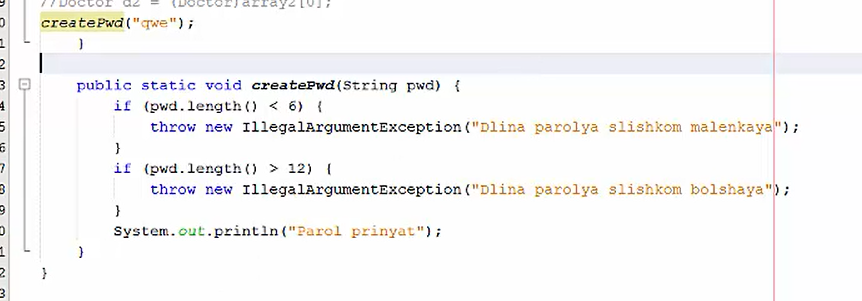




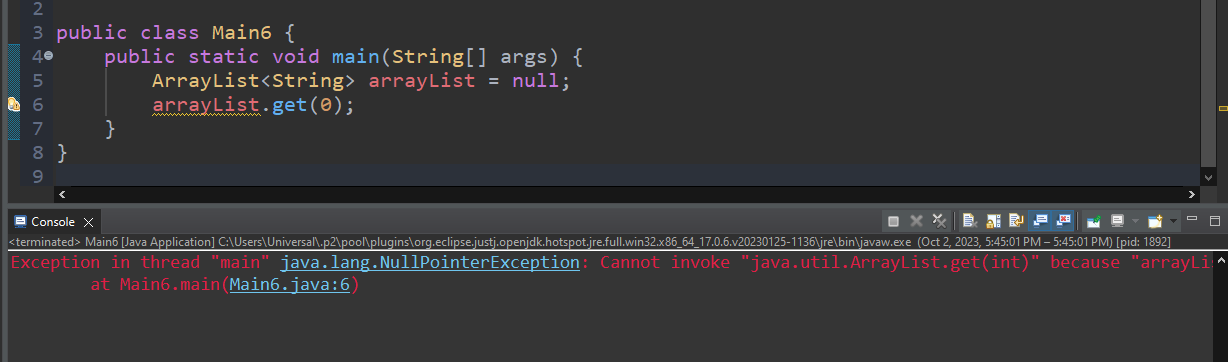




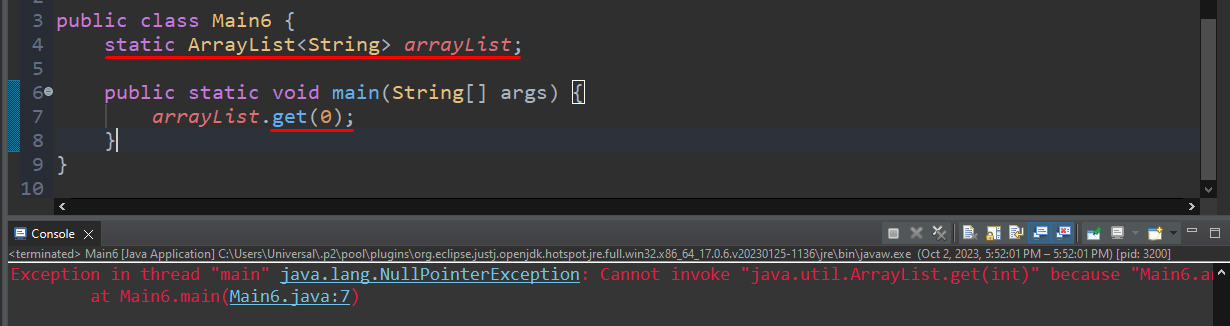




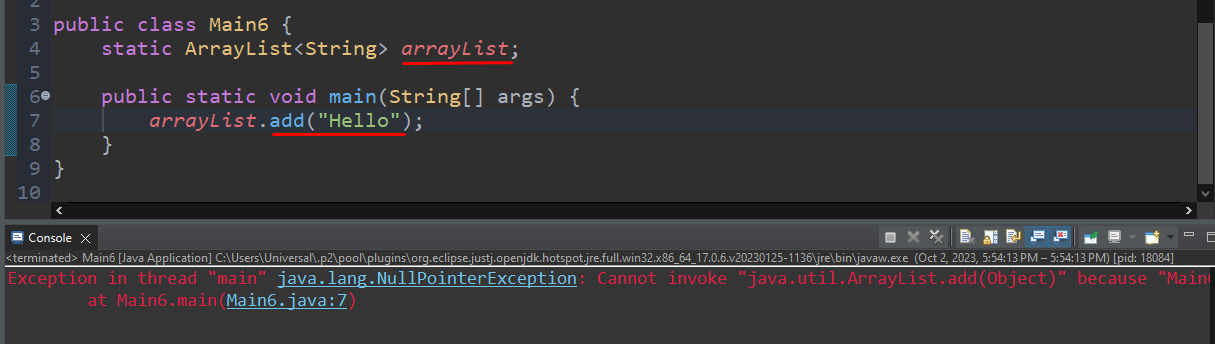
NullPointerException ga misol. arraylist ga biz null ni o’zlashtirdik, endi undan 0 indexdagi elementni olmoqchi bo’lsak, NullPointerException exceptionni qaytaradi, chunki null ni o’zlsahtirgan:



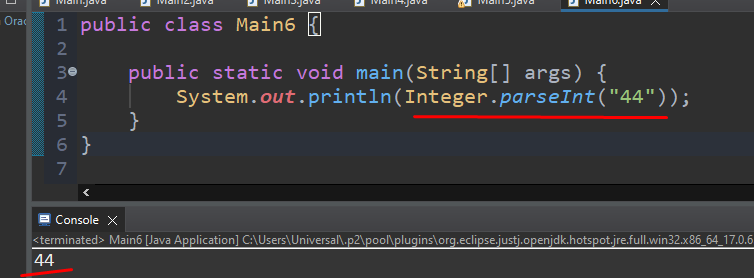
NullPointerException ga boshqa misol ko’ramiz. Pastda 4-qatorda arrayList o’zgaruvchisi default holatda boshlang’ich qiymat sifatida null ni oladi, endi undan 0 indexdagi elementni olmoqchi bo’lsak, NullPointerException exceptionni qaytaradi, chunki null ni o’zlsahtirgan:



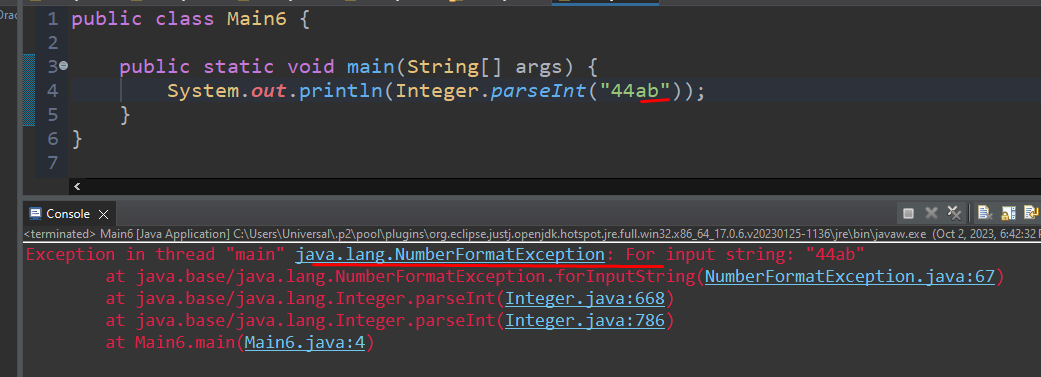
Agar qiymat qo’shmoqchi bo’lsak ham, NullPointerException beradi. Chunki arrayList o’zgaruvchisi o’zida boshlang’ich qiymat sifatida null ni ushlab turibdi:



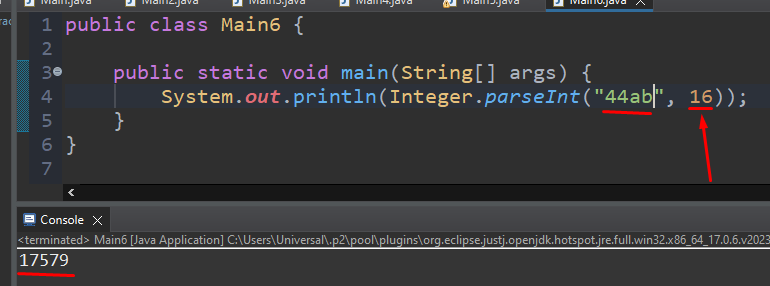
NumberFormatException haqida ko’ramiz misol. Agar string imiz sondan tashkil topgan bo’lsa, uni integer ga cast(o’girishimiz) qilishimiz mumkin:



Agar string imiz ham sondan ham stringdan tashkil topgan bo’lsa, u holda uni cast qilishda NumberFormatException tashlanadi:



Lekin parseInt() method 2 argument sifatida 2lik, 8lik, 10lik va 16 lik sanoq sistemasini qabul qilishi mumkin. Bunda “44ab” sonini 16 lik sanoq sistemasiga o’girishimiz mumkin va bu xatolik keltirib chiqarmaydi. Nega bu yerda aynan 16 likni deyapmiz, chunki “a” va “b” harflari 16 lik sanoq sistemasida mavjuddir. Pastda 16 likka o’girgan holat ko’rsatilgan:



Lekin 16 lik sanoq sistemasiga o’girib bo’lmaydigan son olsak, u holda NumberFormatException exception otiladi, masalan “p” va ”k” simvollari 16 lik sanoq sistemasida yo’q, shuning uchun xatolik beradi:

