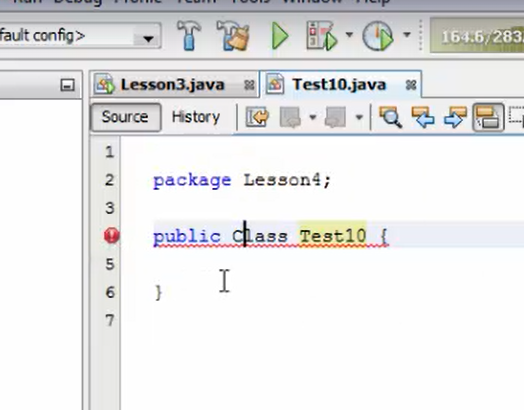
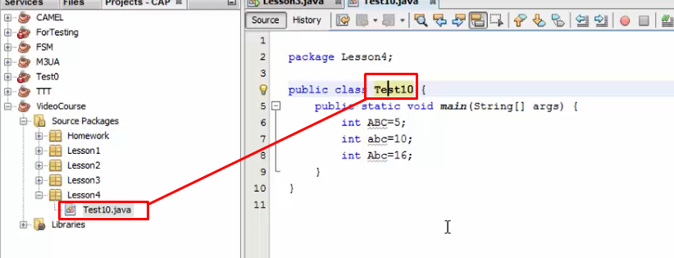
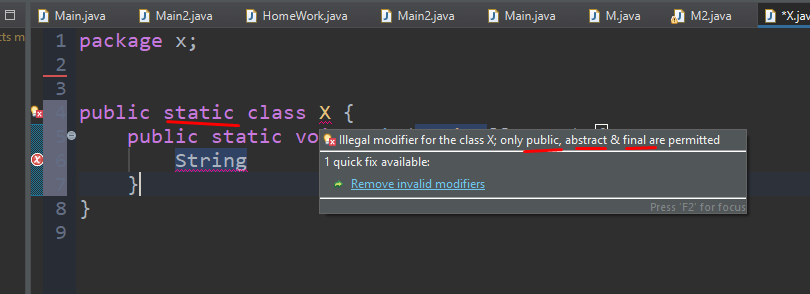
-Java da **class** so’zini **katta harflar** yoki **katta-kichik harflar** bilan aralashtirib yozish mumkin emas, xatolik beradi faqatgina **class** deb yozilishi kerak:



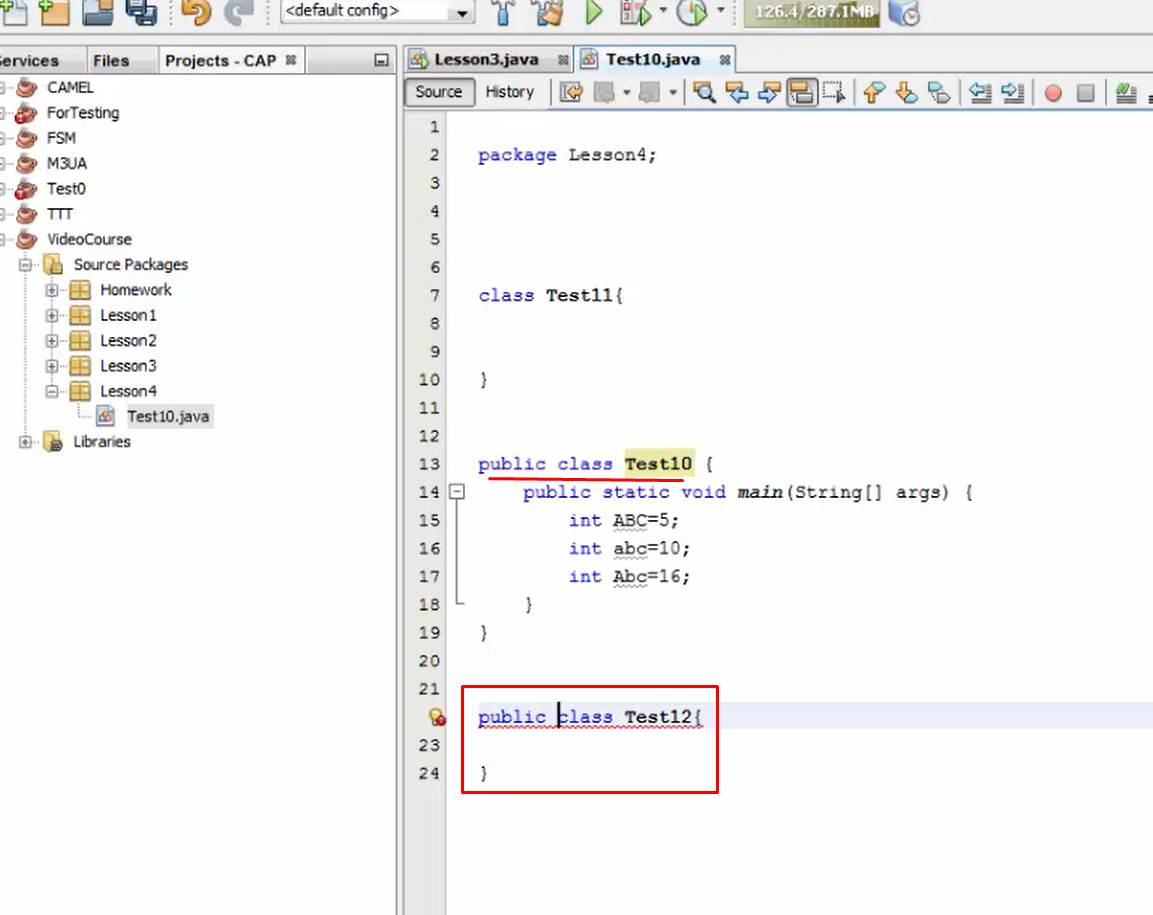
**-Class** nomi va **fayl** nomi bilan bir xil bo’lishi kerak aks holda error beradi, ya’ni **Test10.java** faylni nomi **Test10** class bilan bir xil bo’lishi kerak:



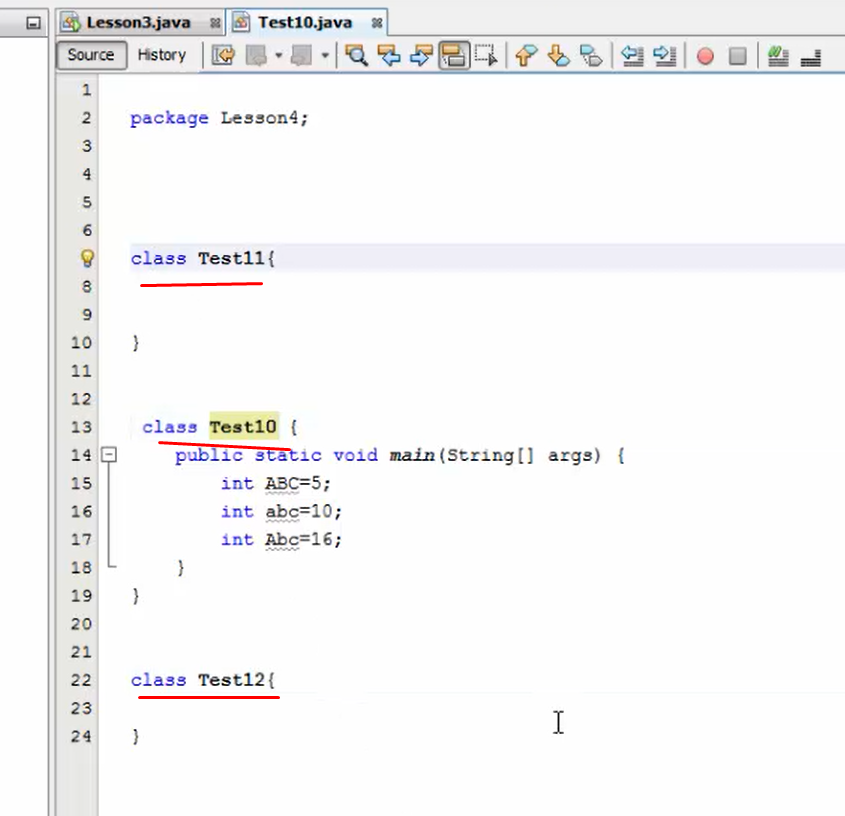
**Class** lar faqatgina **public**, **abstract** va **final** keywordlari bilangina yozilishi kerak, aks holda xatolik beradi. Masalan, pastda **X** class **static** kalit so’zi bilan e’lon qilinyapti. Bu esa xato chunki classlar faqatgina **public**, **abstract** va **final** keywordlari bilangina yozilishi kerak, aks holda xatolik beradi. Pastdagi misolda xatolik bergan holat aks etgan:



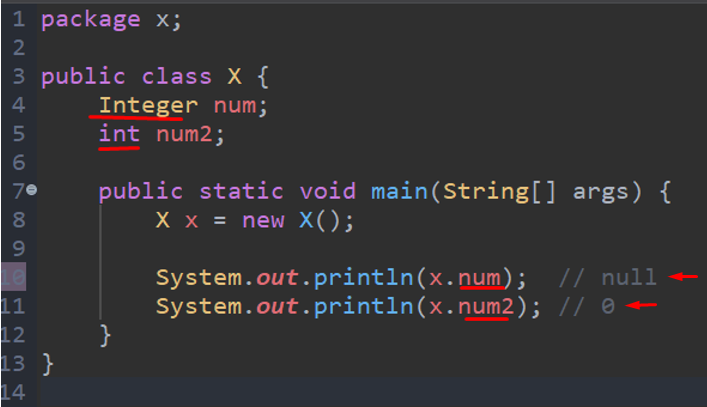
- Bitta java file da faqat bittagina **public** class bo’ladi, aks holda **error** beradi:



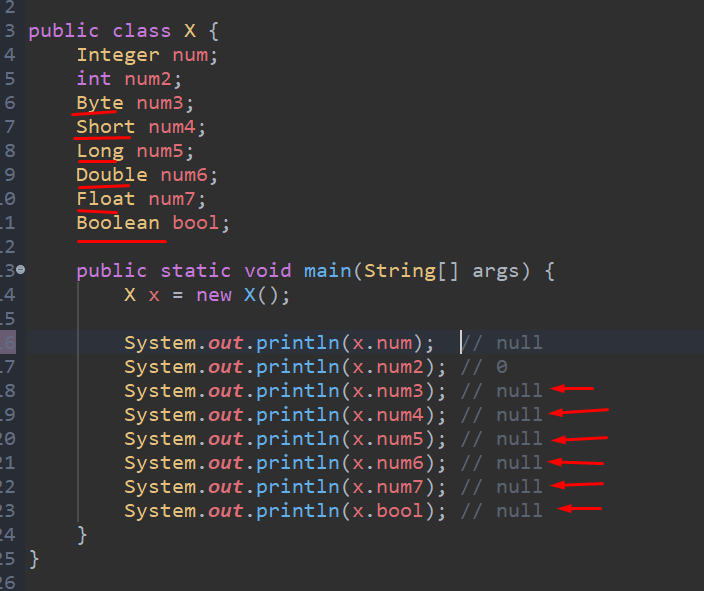
- Lekin bitta **java faylda** bir nechta **public** bo’lmagan class lar bo’lishi mumkin. Bunda hech qanday xatolik bermaydi, to’g’ri ishlaydi:



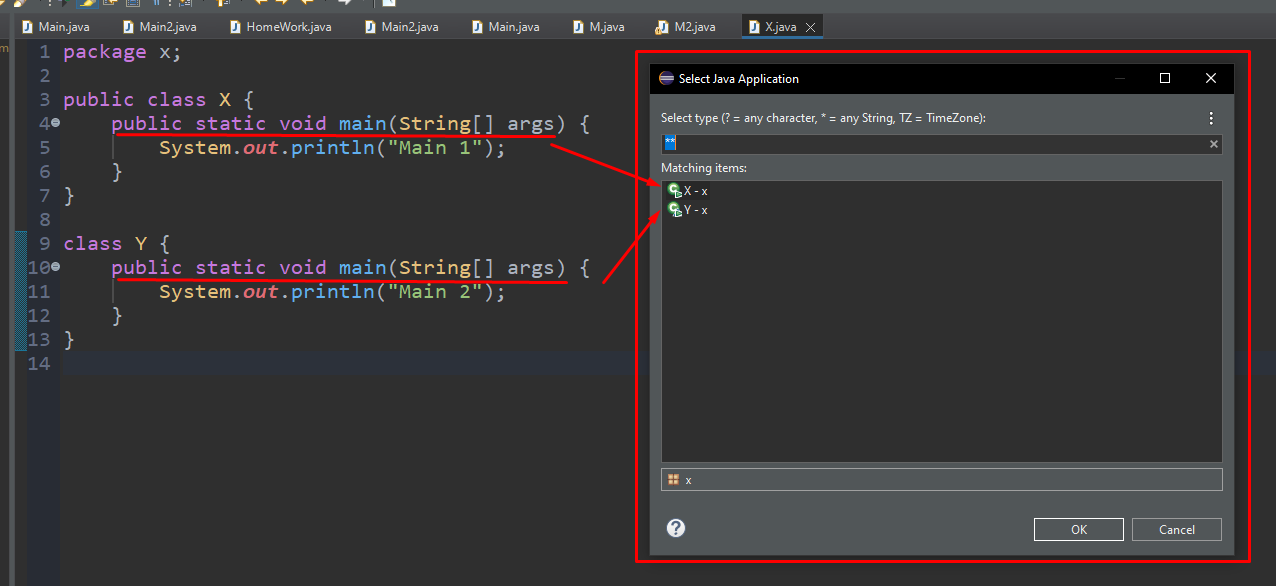
- Biz bilamizki, agar **primitive** instance o’zgaruvchimiz **int** bo’lsa va unga e’lon qilinganda hech qanday boshlang’ich qiymat berilmasa, u holda bu o’zgaruvchi default **0** sonini oladi(**5**-qator va **11**-qator). Xo’sh agar shu **int** typeni o’rniga uning **Wrapper** classi bo’lmish **Integer** classni yozsak, boshlang’ich qiymat sifatida nimani oladi? Javobi oddiy, **null** ni oladi chunki **Integer** class bu **reference** typedir. Shuning uchun **null** ni oladi(**4**-qator va **10**-qator):



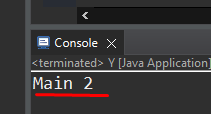
Xuddi shunday boshqa **primitive**(**byte**, **short**, **long**, **double**, **float**, **boolean**) typelarni **Wrapper**(**Byte**, **Short**, **Long**, **Doouble**, **Float**, **Boolean**) classlari ham boshlang’ich qiymat sifatida **null** ni oladi. Chunki **Wrapper** classlar **reference** typedir:



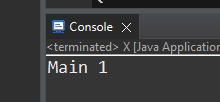
Agar java faylimizda 2 ta **main()** method bo’lsa, bizdan so’raydi qaysi birini run qilishini:



Agar **Y** classni tanlasak u holda **Y** class dagi **main()** methodni run qiladi:

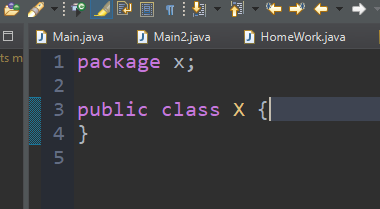


Agar **X** classni tanlasak u holda **X** class dagi **main()** methodni run qiladi:



E’tibor bergan bo’lsangiz, **Y** class run bo’lganda, faqat shu **Y** classni **main()** methodi ishga tushdi, **X** classniki esa yo’q. Chunki bu classlar mustaqildir. Xuddi shuni teskarisi **X** ishga tushganda, **Y** niki chaqirilmaydi.

Agar classimizda **main()** method bo’lmasa, u holda biz bu dastrurni ishga tushira olmaymiz. Chunki **main() m.** bor class **executable** hisoblanadi:



**Reference type in method parameter**

Javada methodlarni parametriga reference type berilsa, u holda parameterdagi qiymat asl objectdan copy oladi. Bu degani asl object va copy object ikkalasi ham bitta objectga murojaat qiladi degani. Pastda misol bilan ko’ramiz. Bizda **car** nomli object yaratildi va bu object yashil rang bilan ko’rsatilgan o’zi yasagan **car** object ga murojat qilyapti. Endi biz shu **car** objectni **foobar()** methodga parameter sifatida berib yuboryapmiz. **foobar()** methodda **Car c** parameter dagi **c** object ham boyagi **car** objectga reference qilyapti. Demak parameterdagi reference type ham asl objectdan copy oladi va asl object bilan bitta joyga pointer(reference)qilar ekan:

