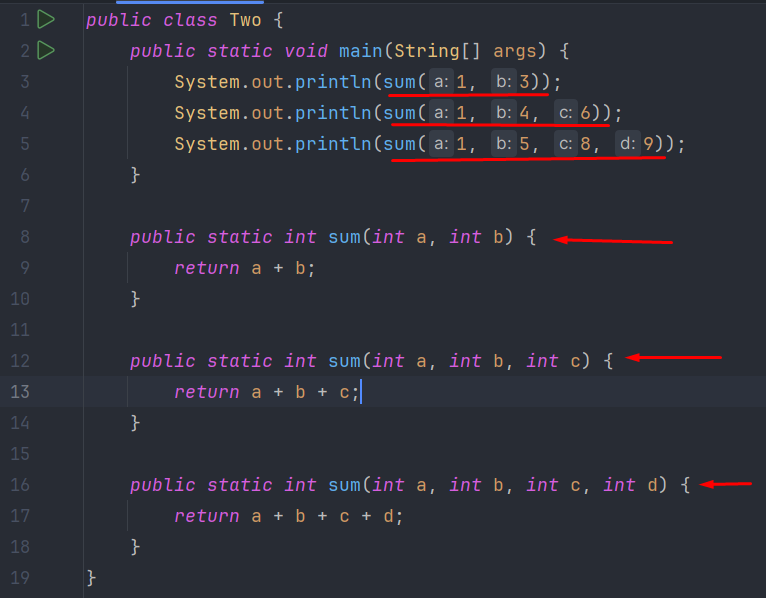
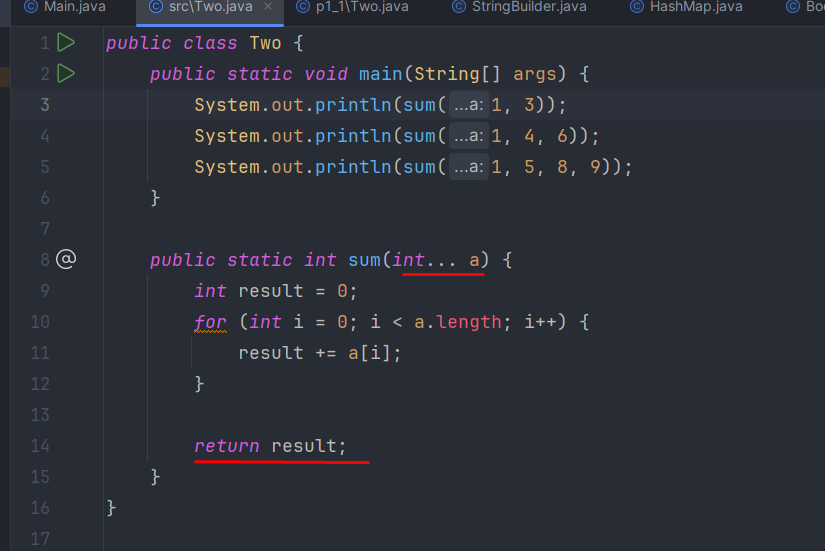
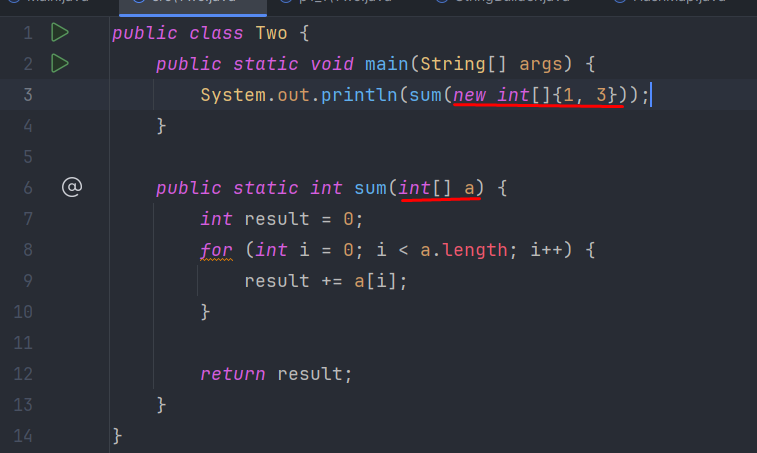
Keling **varargs** ni ko’rishdan oldin bir misol ko’raylik. Pastda bizda 3 ta **sum()** m. bor bo’lib, har birini argumentiga har xil argumentlar berib yuborilgan. E’tibor bersangiz bu argumentlar ko’paygan sari, methodimiz ko’rinishi qo’pollashib boraveradi:



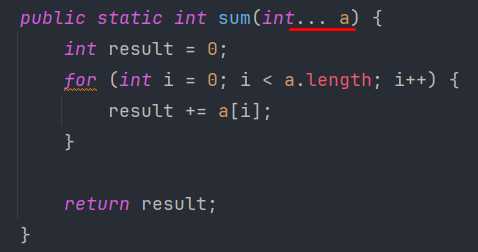
Shuning uchun bunday holatlardan qutulish uchun, **varargs** degan sintaksis kiritilgan. E’tibor bersangiz biz bor yog’i bitta method yozdik va uni argumentiga **int …a** deb **varargs** ni e’lon qildik. Endi qancha qiymat bersak ham bu methodga farqi yo’q qabul qilaveradi. Juda qulay:



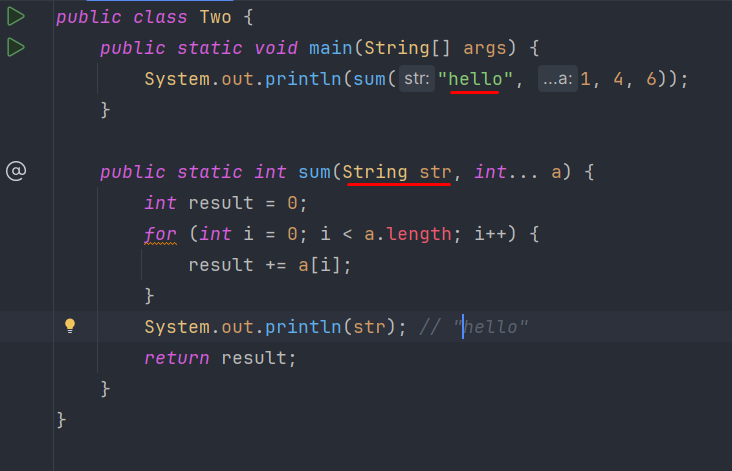
Aslida massiv berib yuborsak ham bo’lar edi, pastdagi kabi, lekin **varargs** ancha qulayroq:



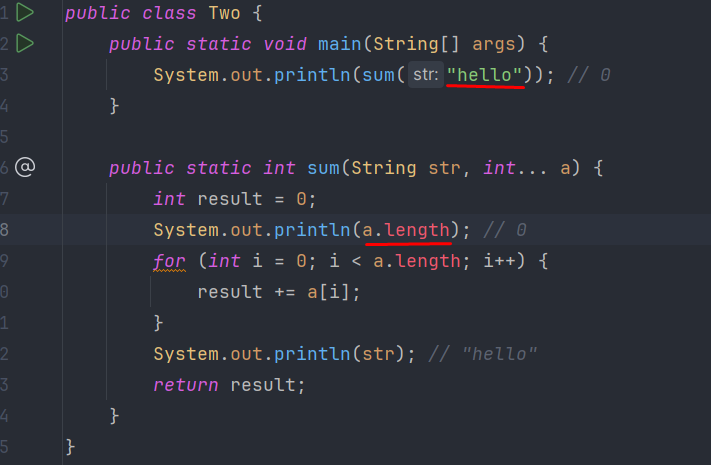
**Varargs** dagi **a** o’zgaruvchimiz massiv hisoblanadi:



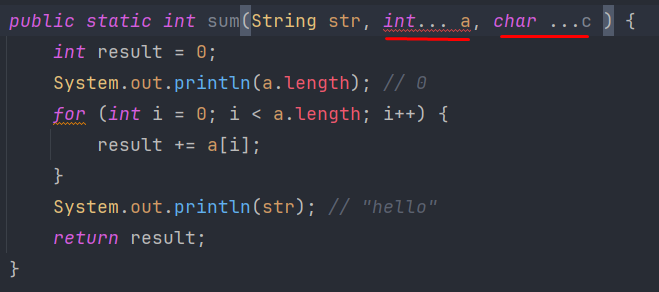
Varargs dan tashqari istasak boshqa qiymat berishimiz mumkin, masalan string qiymat. Bunda faqat bir narsani unutmaslik kerakki, varargs har doim eng oxirida yozilishi shart, aks holda xatolik beradi:



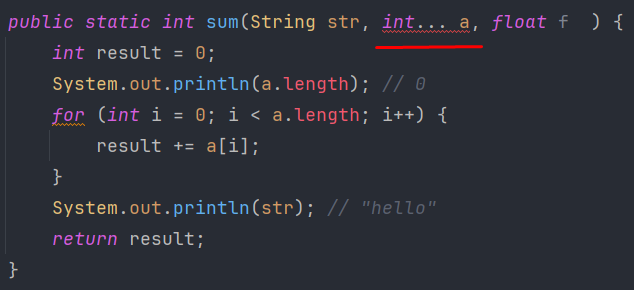
Agar istasak **varargs** ga qiymat bermasligimiz ham mumkin, ya’ni qiymat bermay qoldirishimiz ham mumkin, xatolik bermaydi.



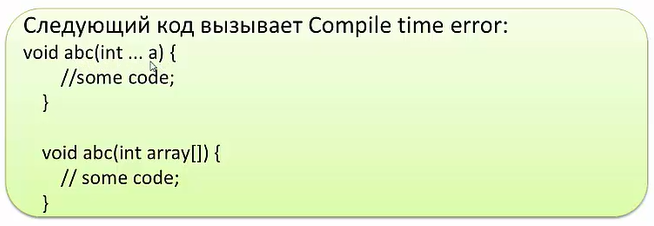
**Varargs** larning soni doim bitta bo’lishi kerak, aks holda xatolik beradi:



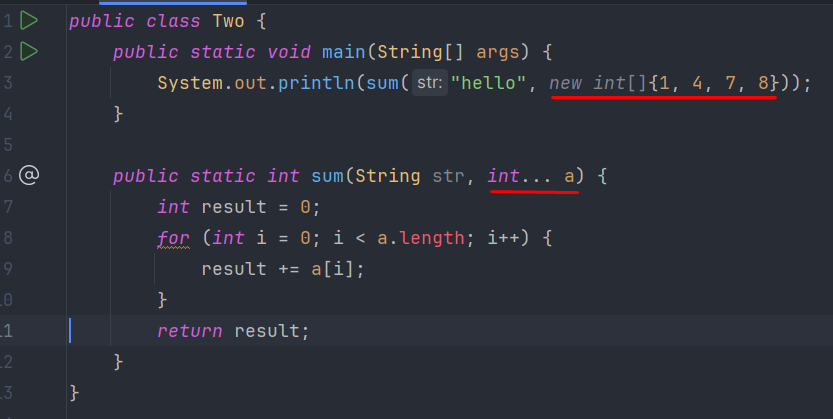
Varargs doim eng oxirida yozilishi shart aks holda xatolik beradi:



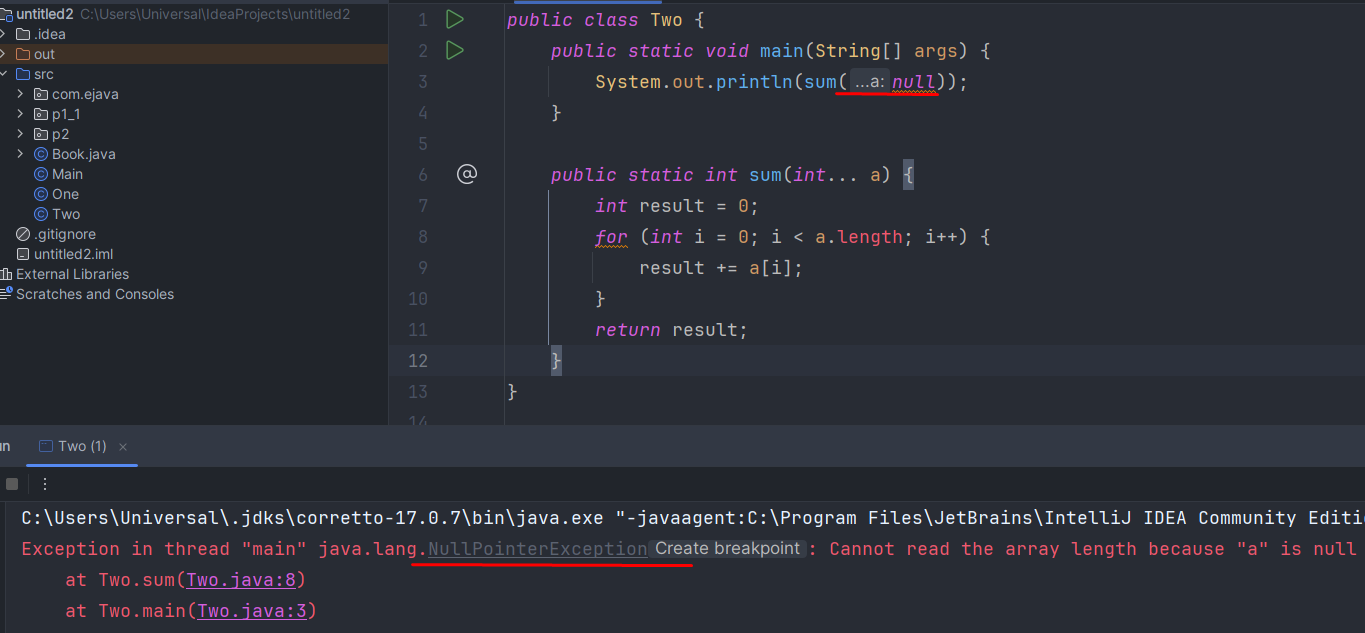
Pastdagi 2 ta abc() methodlar yozilgan bo’lib, bu ifodalar xatolik beradi. Sababi varargs **(int …a)**ham oxirida **int a[]** ga aylanadi. Natijada 2 la methodda ham argumenti **int** type li massiv(**int a[]**) bo’lib qoladi. **Overloading** qoidasiga bu ziddir:

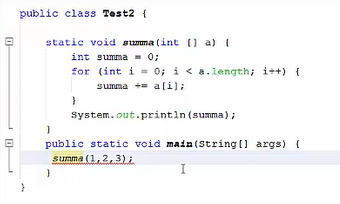


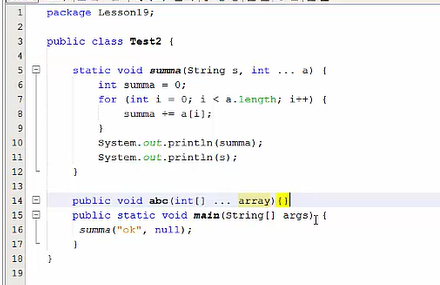
Varargs oxiri baribir massiv bo’lgani uchun, istasak biz massiv(**new int[]{1,4,7,8}**)ni **varargs** ga berishimiz mumkin:

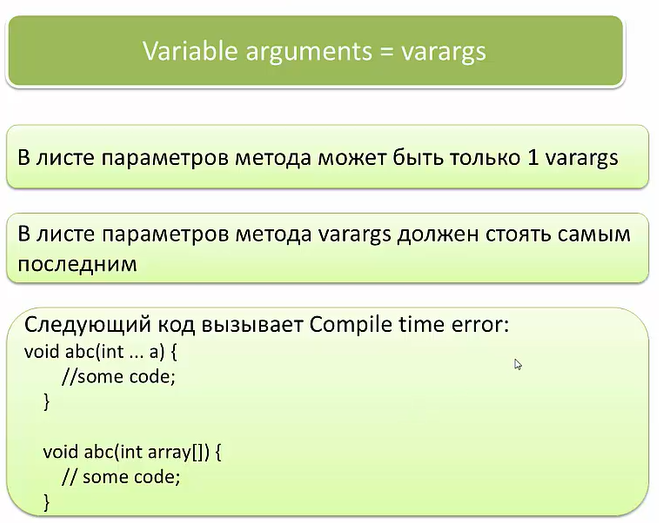


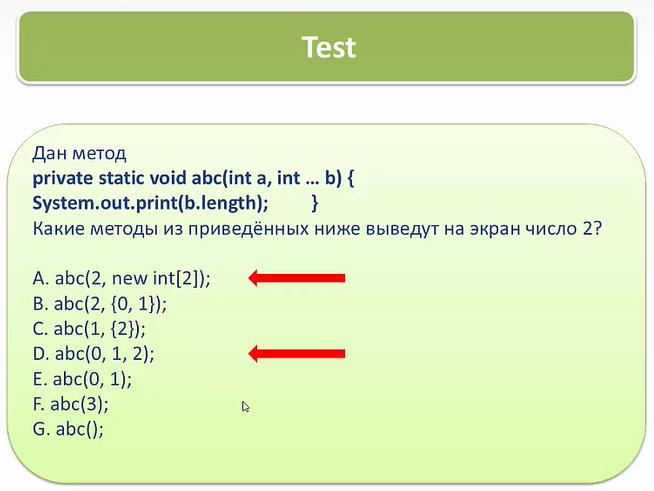
Varargs massiv bo’lgani uchun va massiv object hisoblangani uchun, unga biz **null** ni berib yuborishimiz mumkin. Bunday holatda compile timeda xatolik bermaydi, aksincha runtime da xatolik beradi:

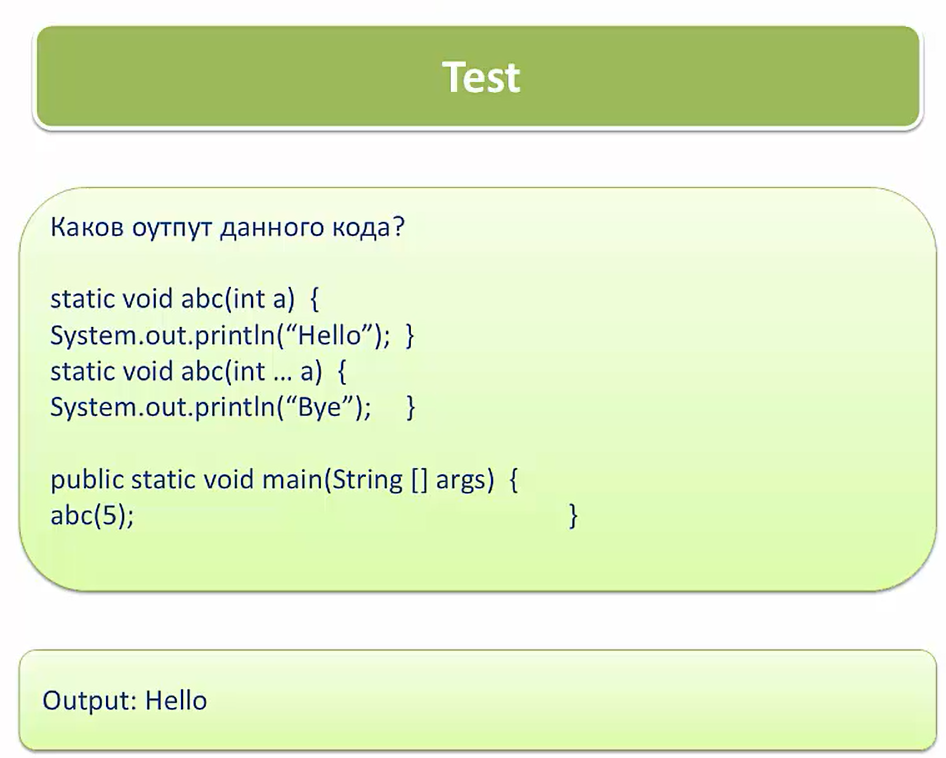




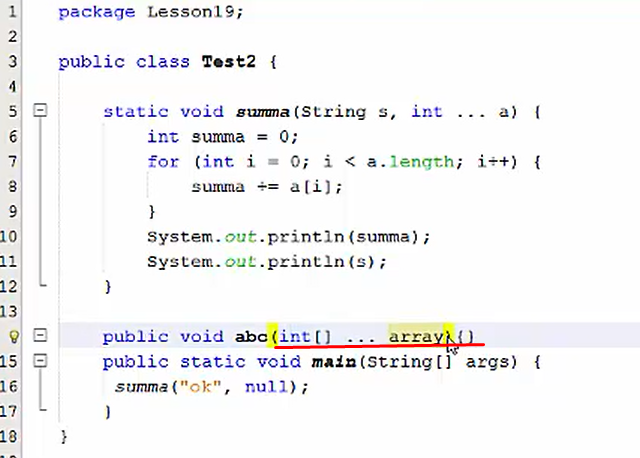






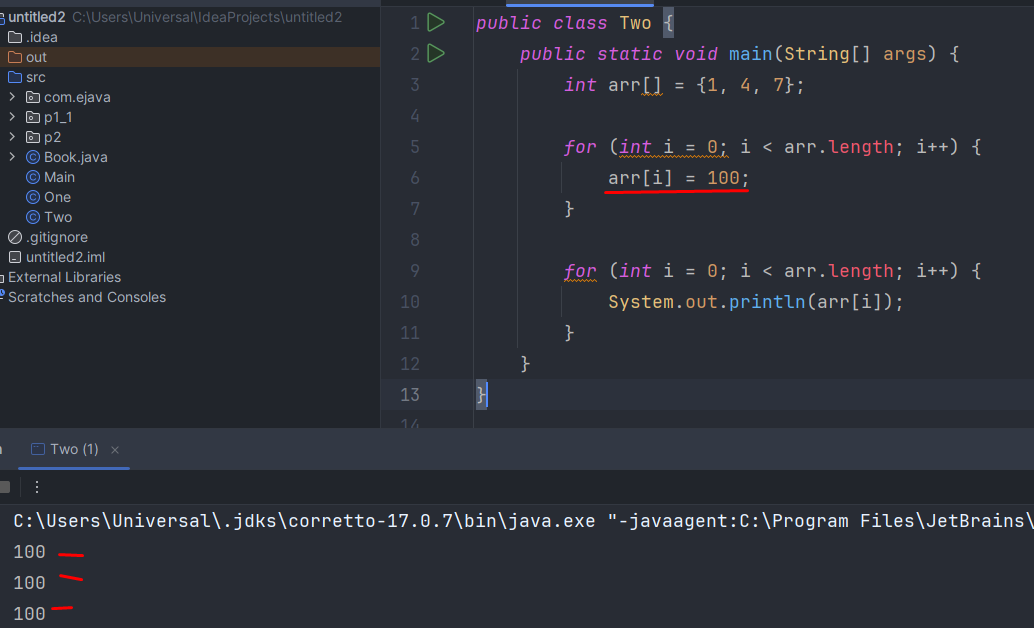


Sababi **abc(int a)** method **abc(int …a)** methodga nisbatan ko’proq mos keladi.

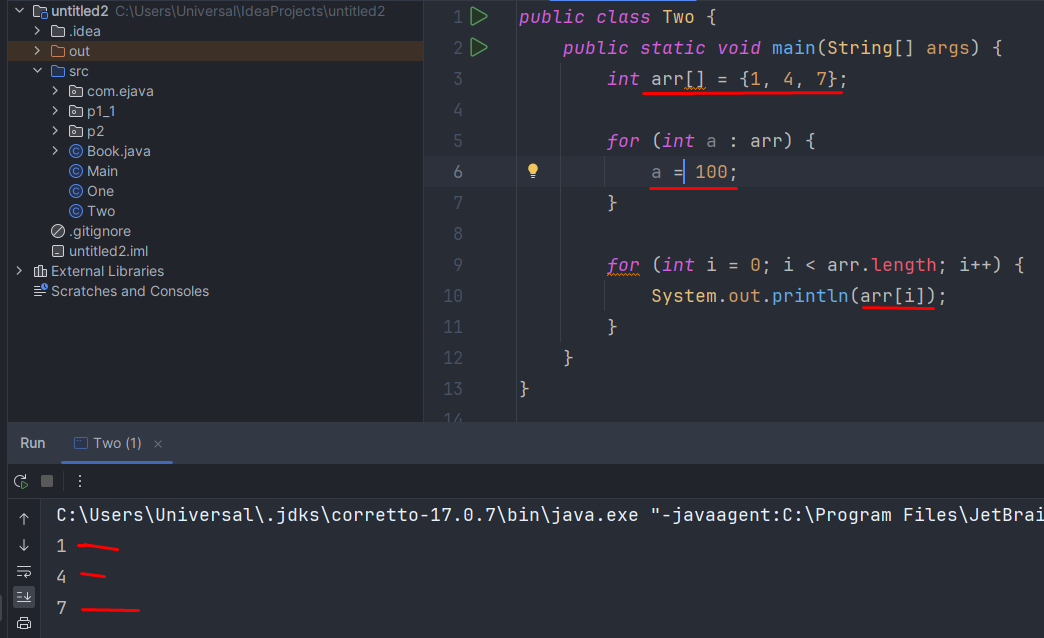


**ForEach vs For loop**

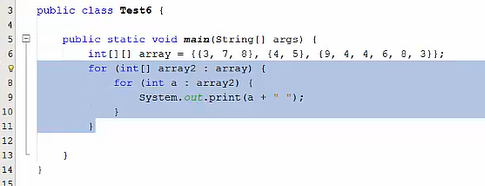
For loop yordamida biz asl arrayni qiymatini o’zgartirishimiz mumkin. Pastda **5-**qatordagi **for-loop** bilan asl arrayni o’zgartirib, keyin uni **9-**qatordagi for-loopda chiqarib ko’ryapmiz:

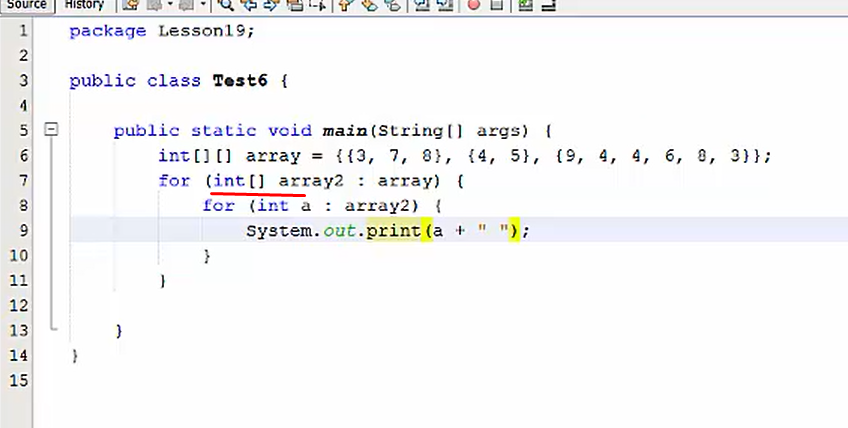


Lekin yuqoridagi holatni **forEach** da yozsak o’xshamaydi. Sababi **forEach** da **5-**qatorda **int a:arr** deganmiz. Bu shuni anglatadiki **a** o’zgaruvchi forEachda har siklda yangidan yaratilyapti, bu degani **6-**qatordagi **a=100;** ifoda faqat har siklda yaratiladigan **a** o’zgaruvchisini qiymatini o’zgartirgin degani. Demak, bunday yozsak, faqat **a** o’zgaruvchisinigina qiymatini o’zgartirar ekanmiz. Lekin asl **arr** massiv o’zgarmasdan qolaverar ekan:



Agar massivimiz **2** o’lchamli bo’lsa, u holda **forEach** quyidagicha yoziladi:





Keling oddiy for loop bilan forEach ni farqini ko’ramiz.

Agar biz da S

