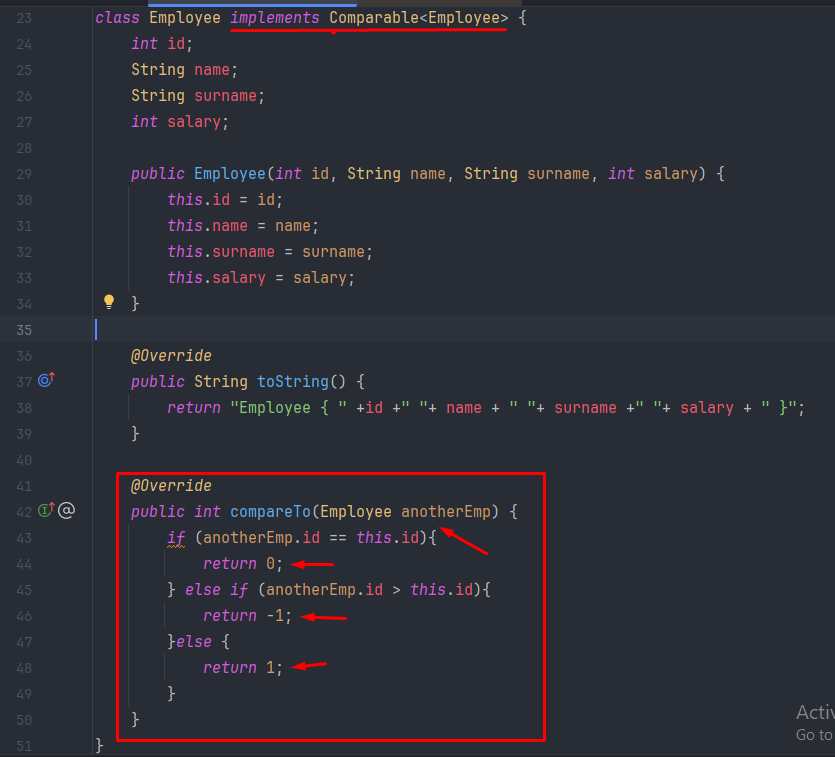
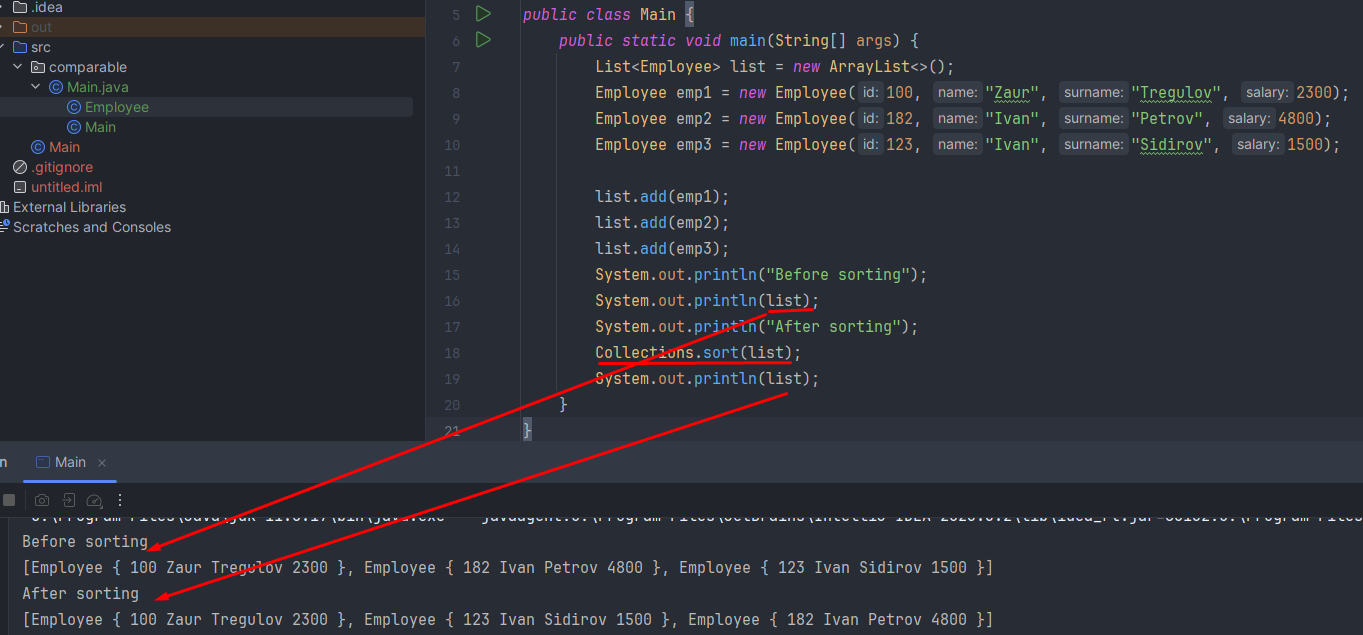
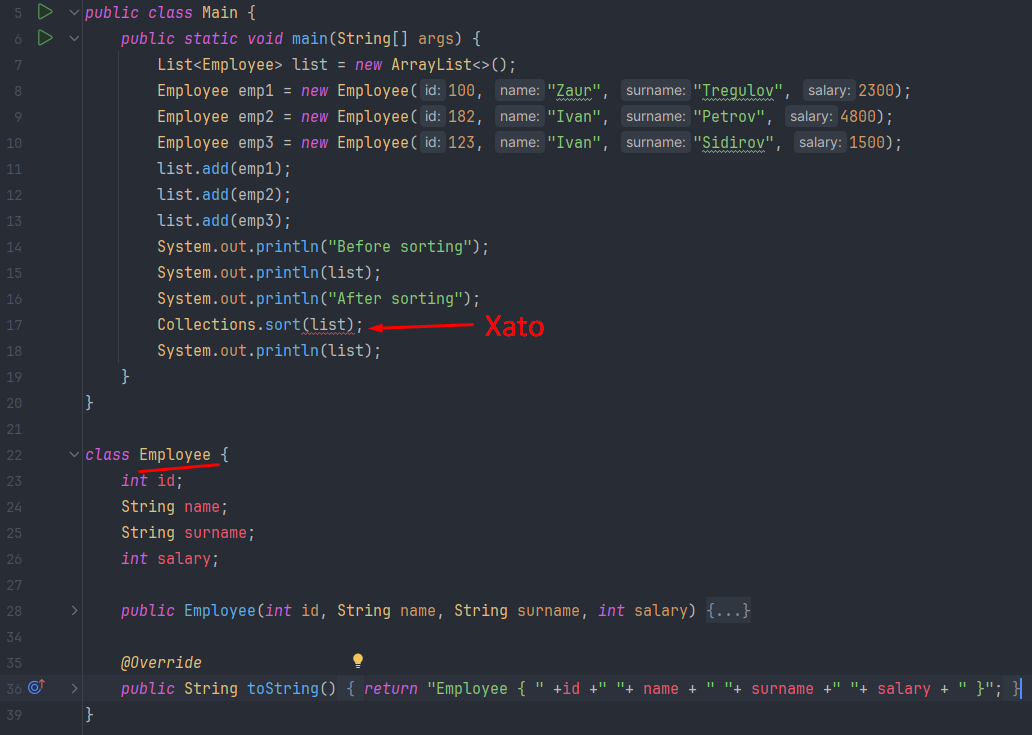
Javada objectlarni to’g’ridan to’g’ri taqqoslab bo’lmaydi. Nafaqat Javada balki boshqa tillarda ham, masalan JavaScriptda ham. Objectni biror propertysiga nisbatan taqqoslash kerak. Masalan id siga nisbatan yoki name, yoki surname, age propertyiga nisbatan taqqoslash kerak. Javada Comparable va Comparator interfacelari bor bo’lib, bu interfacelar shu ishlarni amalga oshiradi. Avval **Comparable** i.ni ko’raylik: Bizda Employee degan class bor bb, bu classda id,name,surname,salary fieldlari bor. Agar biz Comparable i.dan implement olsak, uning compareTo() methodini override qilishimiz kerak bo’ladi. Bu method int type qaytaradi. Agar 0 qaytarsa ular teng bo’ladi. Bu method JS dagi sort methodiga o’xshab ketadi. Unda ham 0,-1 va 1 qaytarar edi.



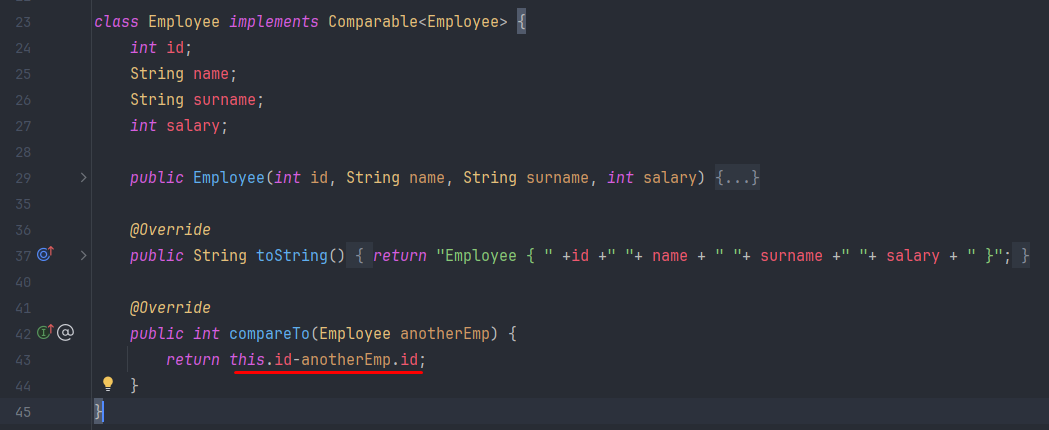
Endi yuqoridagi holatni ishlatib ko’ramiz. Employee dan object yaratib, Listga qo’shib uni sort qilamiz. Biz id bo’yicha sort qildik shuning uchun id lar saralanib qoldi:



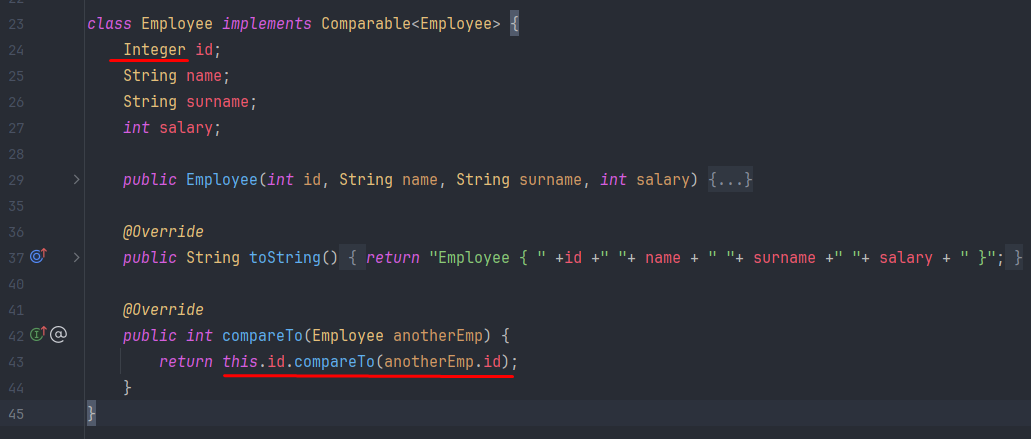
Agar biz Comparable interfacedan implement olmasak, u holda biz Listimizni sort qila olmaymiz. Xatolik beradi. 17-qatorda berilgan:



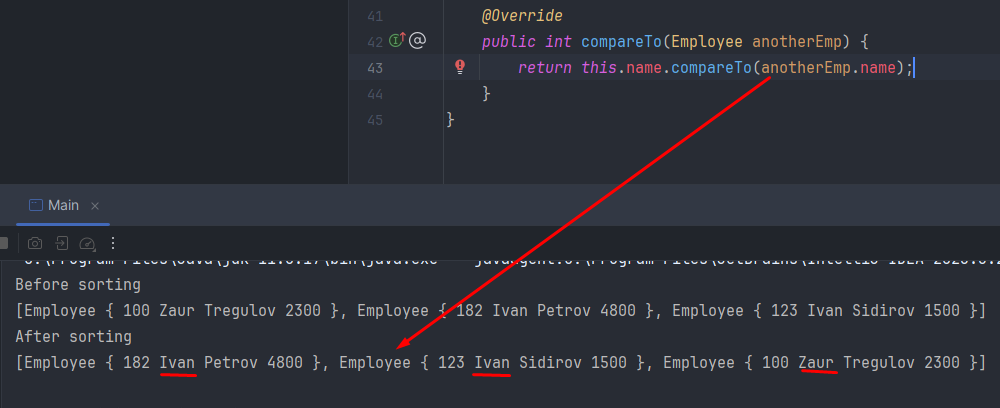
Agar istasak shunchaki id larni ayirmasini ham olib, qaytarishimiz mumkin:



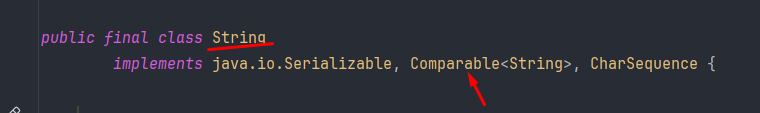
Agar istasak biz int ni o’rniga uning wrapper classini(Integer) yozishimiz mumkin. Sababi Integer classi ham Comparable i.dan impl. olgan:



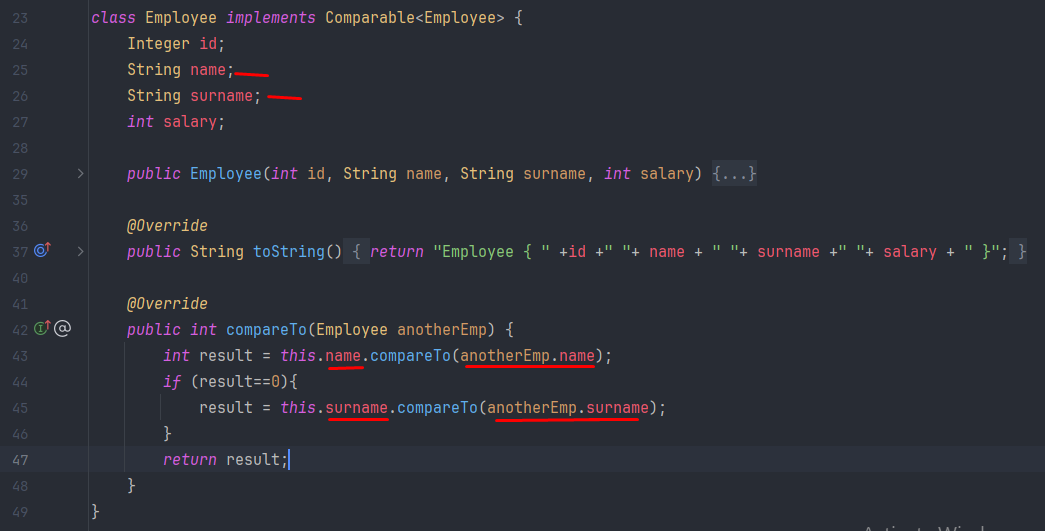
Agar name bo’yicha sort qilish kerak bo’lsa, u holda pastdagi kabi yozish mumkin.



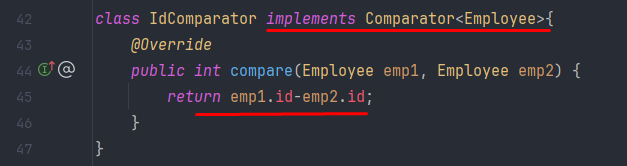
Chunki String classi Comparable interfacedan implement olgan.



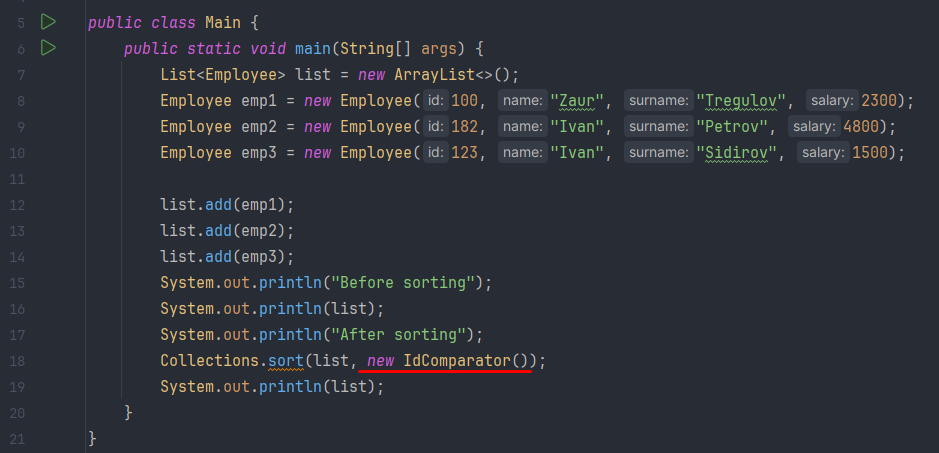
Agar istasak ham name bilan ham surname bilan solishtirish mumkin. Bunda masalan agar name lar teng kelib qolsa, ularni surname bilan ham solishtirish mumkin:



**Comparator** interfaceni ko’raylik. Bu ham Comparable ga o’xshaydi. Ozgina farqi bunda Employee class implement olmasdan, balki boshqa class orqali implement olinadi va uning **compare()** methodi orqali **compareTo()** methodida yozgan logikamizni yana yozib qo’yamiz. Bu method 2 ta object qabul qiladi**. Employee emp1** va **Employee emp2** objectlarni qabul qiladi:



Endi **IdComparator** classimizni eski kodimizda ishlatgandek, 18-qatorga undan object olib yaratib qo’yamiz:



Istasak name fieldi uchun ham NameComparator class yozib, uni sort qilishimiz mumkin:

