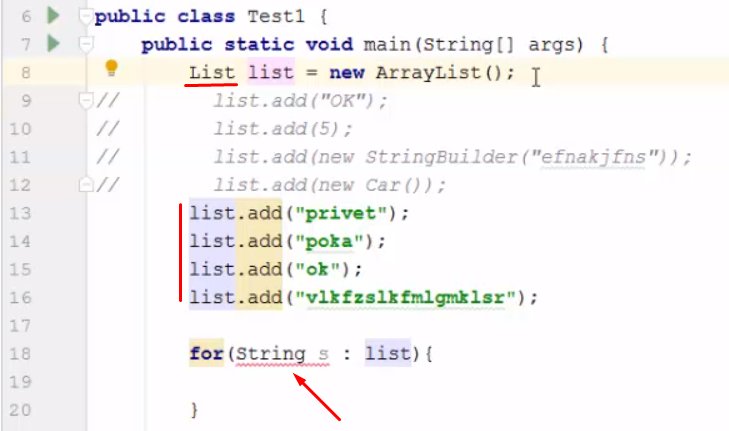
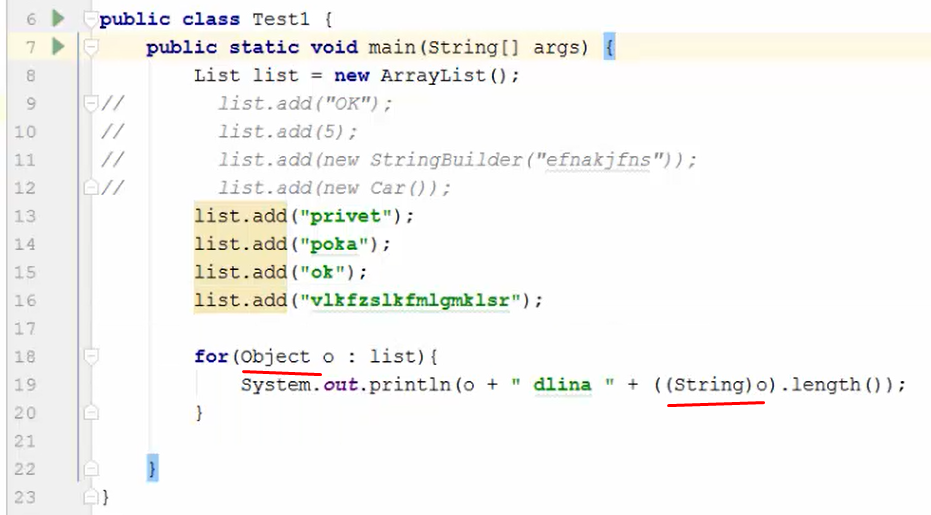


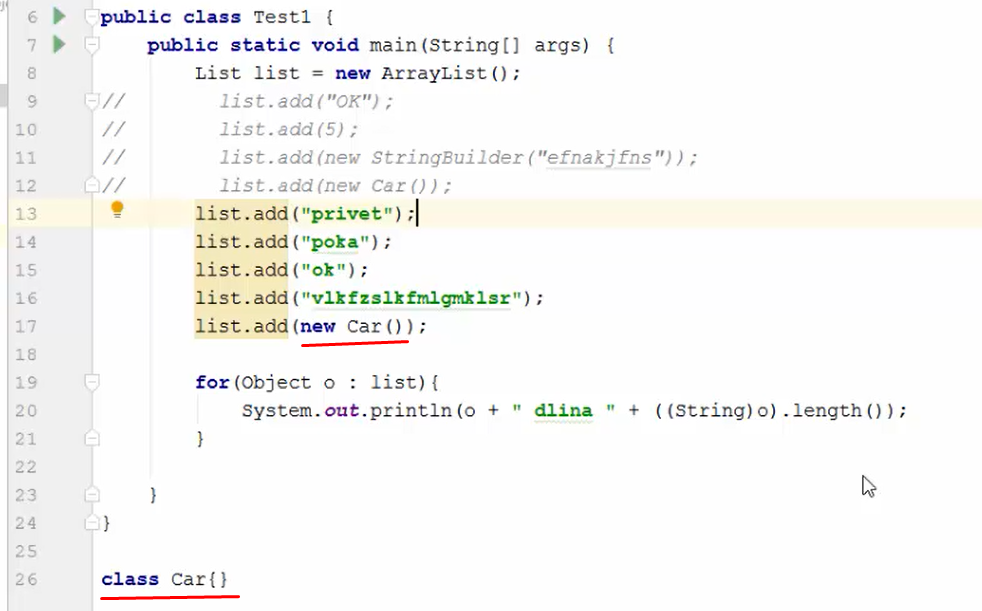
Generics ishlatmasdan List classga masalan String qo’shsak, uni keyin foreach bilan loop qilmoqchi bo’lsak, u holda xatolik beradi sababi ArrayList da generics ishlatmasak, default holatda type Object bo’lib, istalgan typedagi qiymat kiritish mumkin bo’ladi:



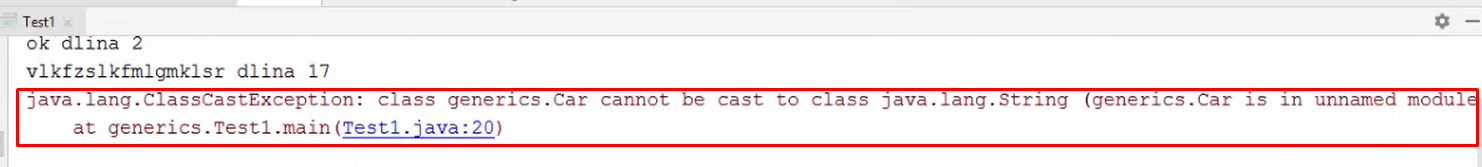
Yuqoridagi xatolikni yo’qotish uchun esa, Object ni ishlatish kerak, va length ni olish uchun esa String ga cast qilish kerak:



Generics ishlatish type-safety ga juda katta ta’sir qiladi. Masalan yuqoridagi **list** imizga **Car** objectini qo’shamiz:



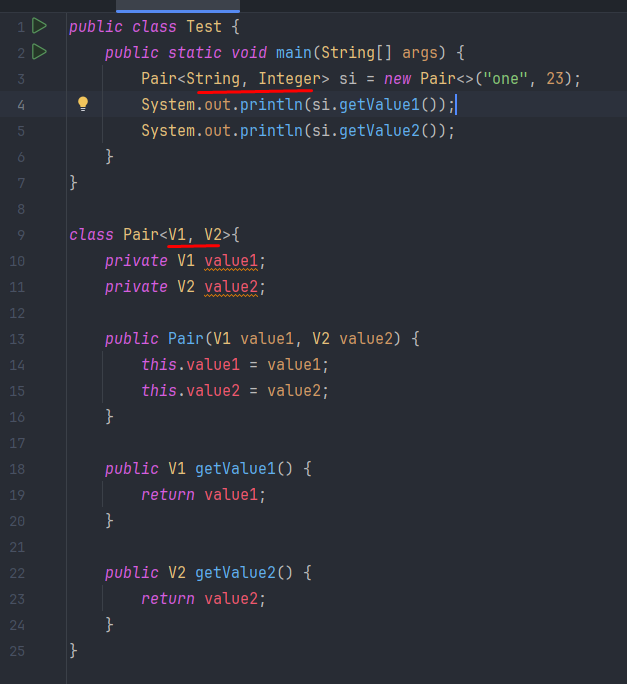
Lekin runtime da xatolik beradi, sababi **Car** classini **length** nomli fielddi yo’q.



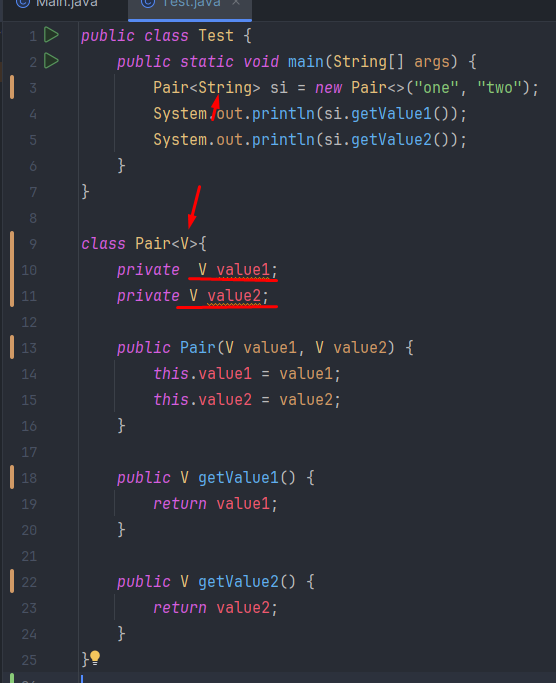
Bitta parameter oladigan Generics class:



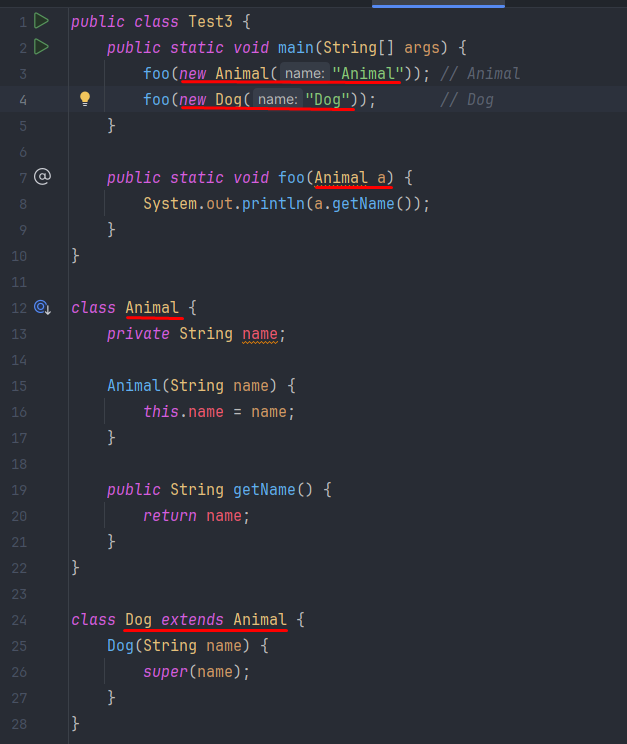
Ikkita parameter oladigan generics class:



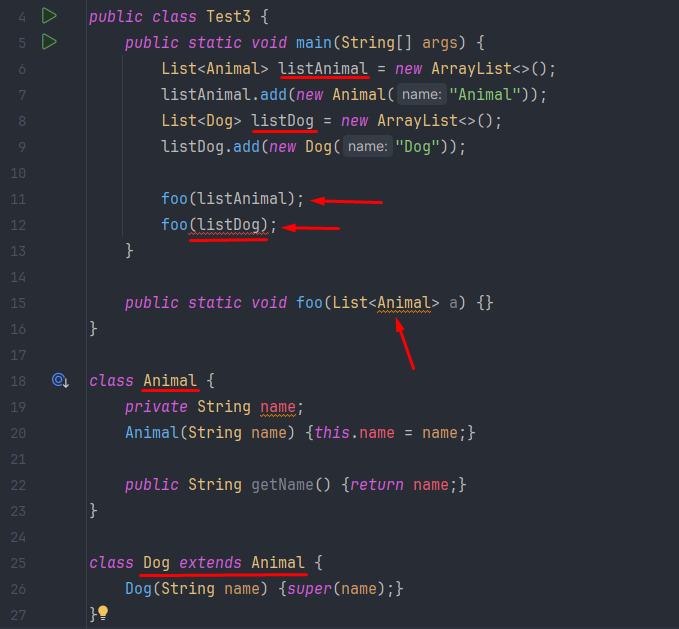
Agar Generic class ikkita bir xil type qabul qilsa, yuqoridagi kabi 2 ta parameter yozmasdan, 1 ta parameter yozish mumkin:



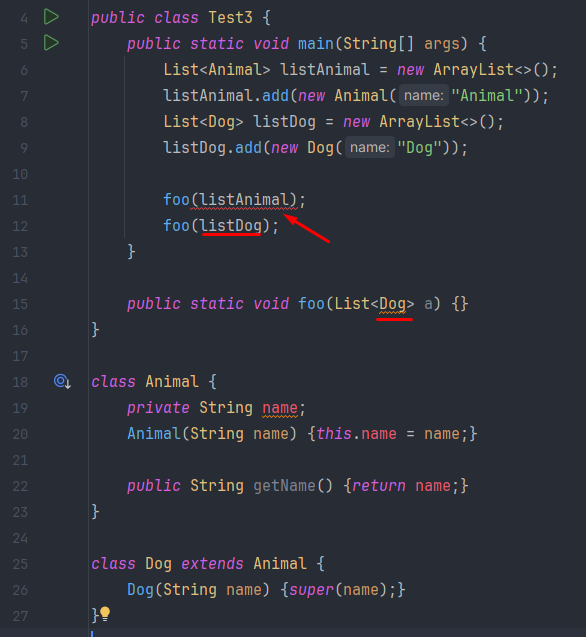
Javada Genericslarda Wildcards degan tushuncha bor. Buni tushunish uchun avval pastdagi misolni tushunaylik. 7-qatordagi **foo()** m-d parameterga Animal classni qabul qiladi. Shuning u-n foo() m-dga Animal va Dog objectlarini berishimiz mumkin, sababi Dog classi Animal classni bolasidir:



Lekin biz shu ishni Genericslarda qilsak, ya’ni 15-qatordagi foo() methodga parametri Animal classni qabul qiladi, lekin biz unga Dog classni objectini bera olmaymiz. Bu holat Genericslarda ishlamaydi, xatolik beradi:



Xattoki foo() m-d ham Dog classni parameter sifatida qabul qilsa, unga Animal objectni jo’nata olmaymiz xatolik beradi. 11-qatorda xatolik beradi:



Bundan qutulish uchun bizga Wildcard degan tushuncha yordam beradi. 13-qatorda biz Wildcarddan foydalanganmiz. “**? extends Animal**” bu wildcardni bildiradi. Bu yerda **?** belgi istalgan type deganidir. Sintaksisni o’qilishi: faqat Animal classdan meros olgan , **?** istalgan type dir. Bu degani Animal class va Animal classdan meros olgan istalgan type dir:

