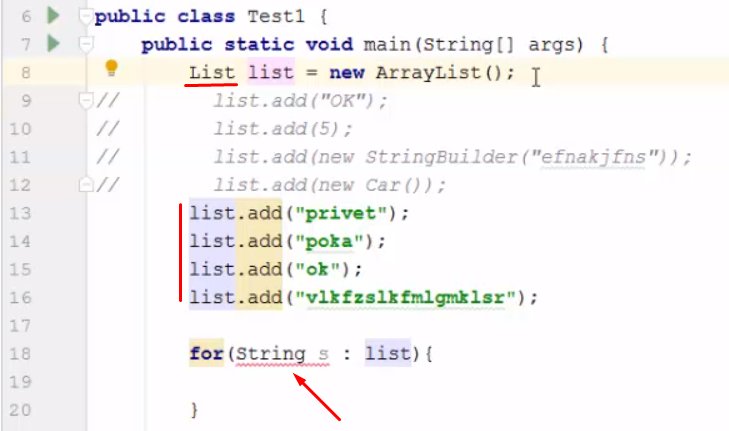
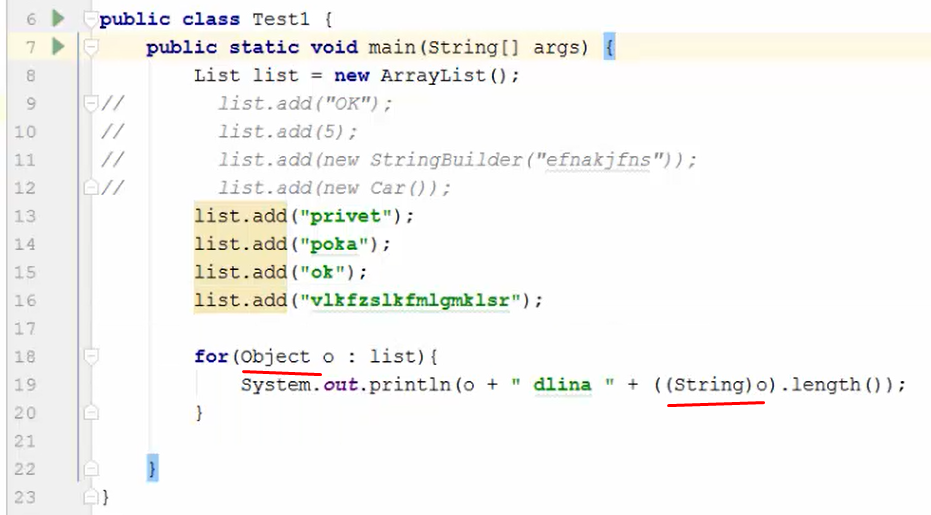


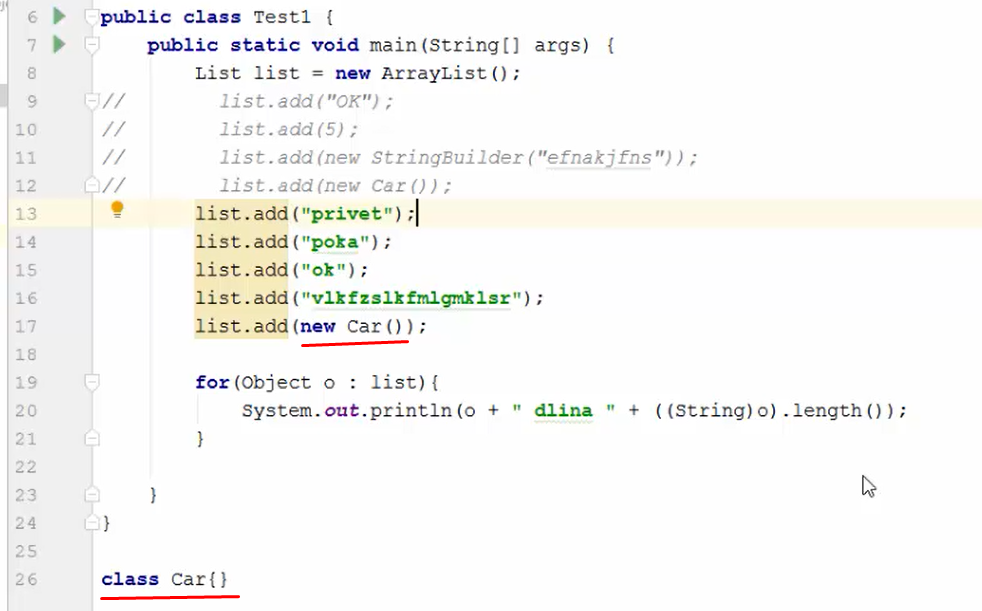
Generics ishlatmasdan List classga masalan String qo’shsak, uni keyin foreach bilan loop qilmoqchi bo’lsak, u holda xatolik beradi sababi ArrayList da generics ishlatmasak, default holatda type Object bo’lib, istalgan typedagi qiymat kiritish mumkin bo’ladi:



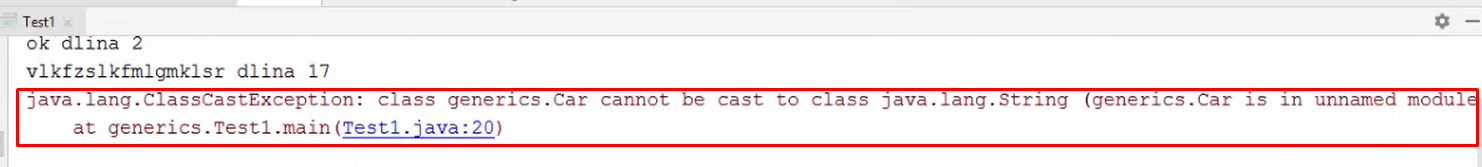
Yuqoridagi xatolikni yo’qotish uchun esa, Object ni ishlatish kerak, va length ni olish uchun esa String ga cast qilish kerak:



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



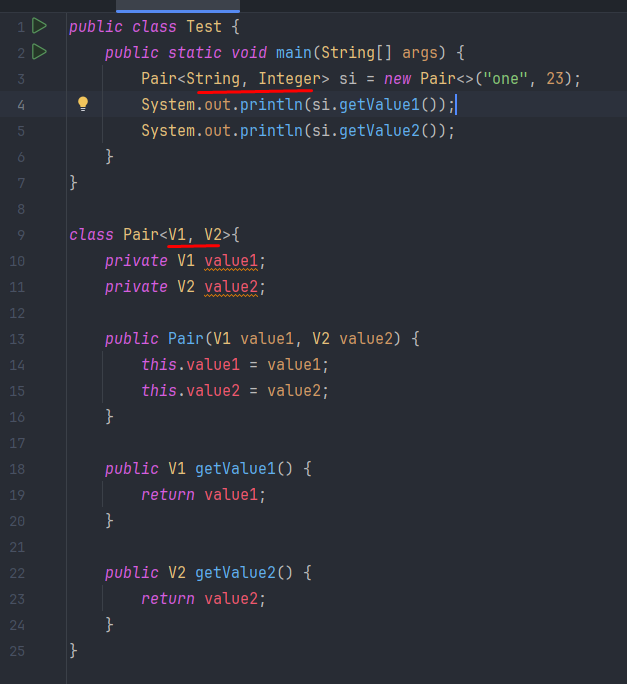
Lekin runtime da xatolik beradi, sababi **Car** classini **length** nomli fielddi yo’q.



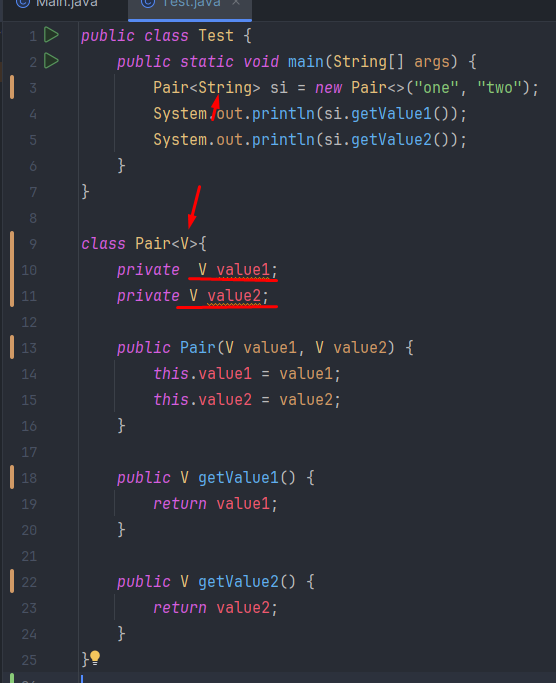
Bitta parameter oladigan Generics class:



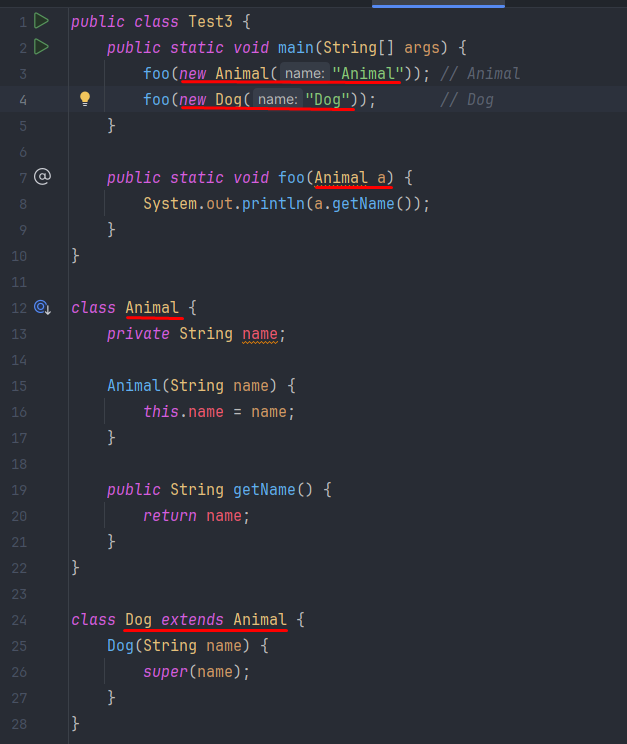
Ikkita parameter oladigan generics class:



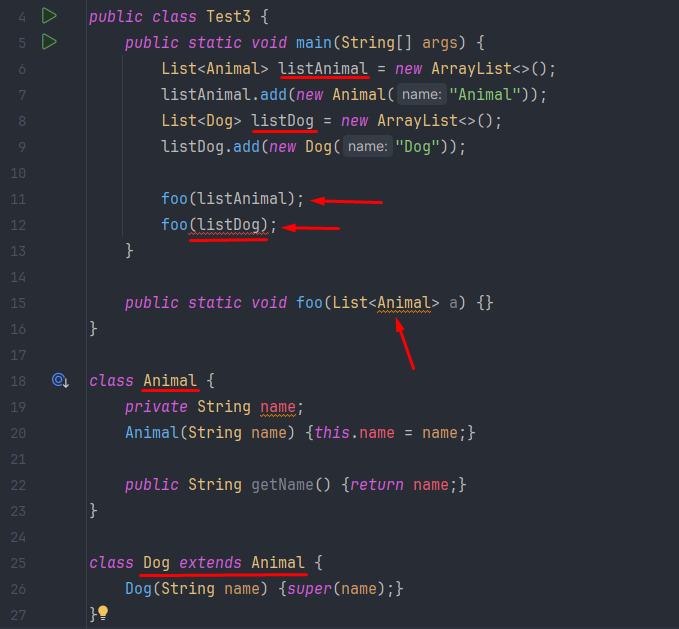
Agar Generic class ikkita bir xil type qabul qilsa, yuqoridagi kabi 2 ta parameter yozmasdan, 1 ta parameter yozish mumkin:



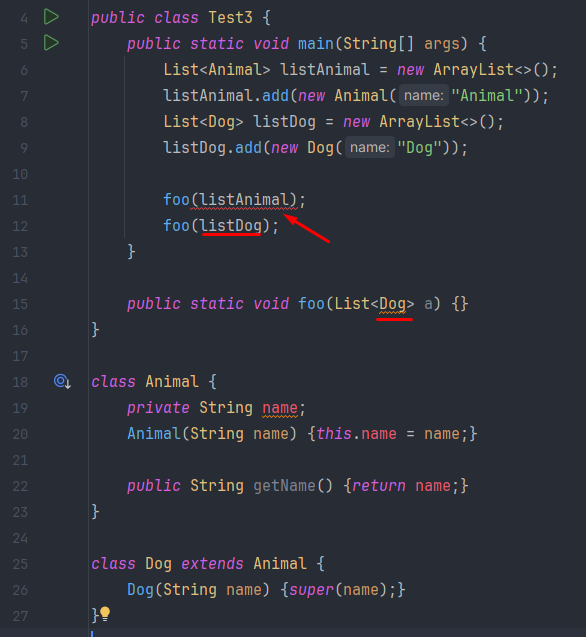
Javada Genericslarda Wildcards degan tushuncha bor. Buni tushunish uchun avval pastdagi misolni tushunaylik. 7-qatordagi **foo()** m-d parameterga Animal classni qabul qiladi. Shuning u-n foo() m-dga Animal va Dog objectlarini berishimiz mumkin, sababi Dog classi Animal classni bolasidir:



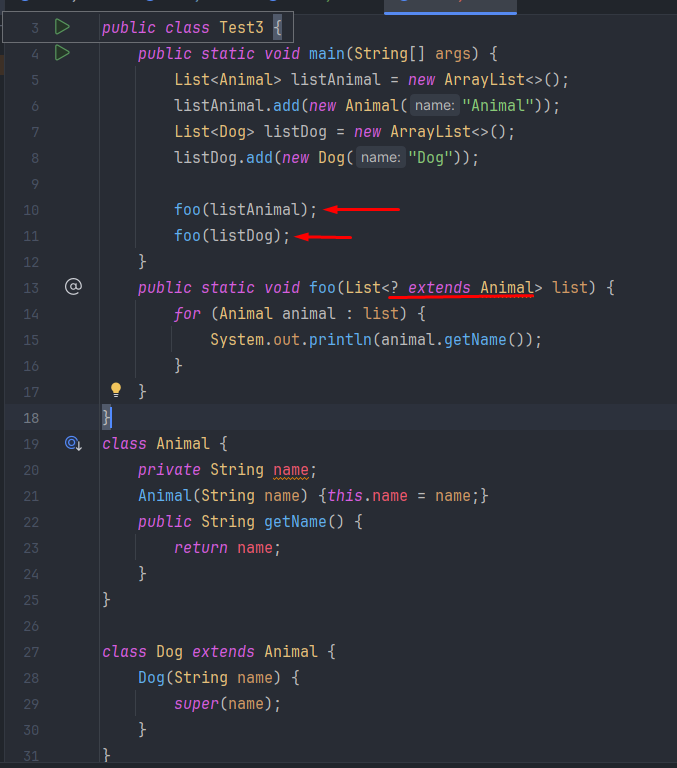
Lekin biz shu ishni Genericslarda qilsak, ya’ni 15-qatordagi foo() methodga parametri Animal classni qabul qiladi, lekin biz unga Dog classni objectini bera olmaymiz. Bu holat Genericslarda ishlamaydi, xatolik beradi:



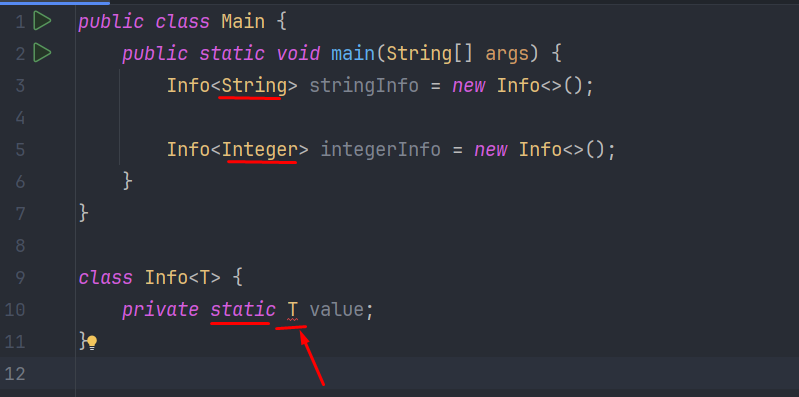
Xattoki foo() m-d ham Dog classni parameter sifatida qabul qilsa, unga Animal objectni jo’nata olmaymiz xatolik beradi. 11-qatorda xatolik beradi:



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:



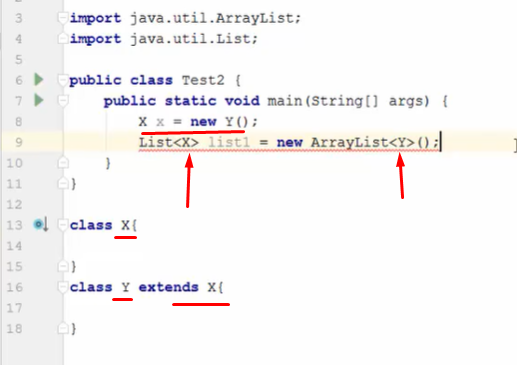
Nega genericslarda static o’zgaruvchini ishlatib bo’lmaydi. Sababi static o’zgaruvchi class load bo’lganda bir marta yaratiladi. Pastdagi misolni ko’rib tushunaylik. Info classni 1-marta ishga tushurganimizda, bu class 3-qatordagi String type bilan chaqiriladi. Biz bilamizki static o’zgaruvchilar 1 marta class load bo’lganda yaratiladi. Demak bizni 10-qatordagi static o’zgaruvchimizni type String bo’ldi. Lekin bizda hali Integer typeli generics turibdi 5-qatorda. Agar uni ham chaqiradigan bo’lsak, u holda static o’zgruvchimiz boshqa yaratilmaydi, shu payti static o’zgaruvchiga Integer type boradi. Bu esa 2 xil typeni static o’zgaruvchiga berganimiz uchun xatolik sodir bo’ladi:



Lekin generics methodlar static bo’lishi mumkin. Chunki static methodlar ham boshqa methodlar kabi har doim chaqirilgand yangidan yaratiladi.

**Generics in Subtyping**

Pastdagi misoldan ko’rish mumkinki, 8-qatordagi ifoda to’g’ri yozilgan, bu qoida polymorphism tushunchasi bo’yicha to’g’ridir, lekin shu qoida genericslarda ishlamaydi. Parent**(X)** typeli List ga Child**(Y)** typeli ArrayListni tenglashtirib bo’lmaydi, xatolik beradi:



Keling nega xatolik berishini ko’raylik pastdagi misoldan. Bizda 10-qatorda List bo’lib, o’ziga parameter sifatida Number classni qabul qiladi. Chap tomonda esa Arraylist bo’lib, Integer classni qabul qiladi. Hamma gap shu chap tomondagi ArrayList objectida bo’lib, bu object faqat Integer typeni qabul qiladi. Shuning uchun unga double yoki float yoki number type qo’shsak xatolik beradi. Demak subtype(child class type)ni qo’shish mumkin emas ekan:

