



ARTIGO - JAVA GENERICS

A parametrização de tipos foi introduzido no JAVA SE 5, e o principal objetivo dessa funcionalidade é limitar o uso excessivo de *casting* de tipos no código, *casting* é uma forma de converter um tipo de dado em outro, por exemplo

```
double d = 57.17;
    int i = (int)d; // casting explicito de double para int
```

Já utilizamos *Generics* nas interfaces do tipo `Collection`, por exemplo `List`, vamos ver que o uso de *Generics* adiciona mais segurança na manipulação de tipos em nosso código. No exemplo abaixo criou-se duas listas, uma com *Generics* `caixa1` e a outra sem declarar o tipo que a variável lista vai manipular `caixa2`, neste caso a lista vai aceitar elementos de qualquer tipo.

Quando isso acontece o compilador classifica a mesma como uma *Raw type*, pois segundo a documentação Oracle (<https://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html#jls-4.8>):

"The reference type that is formed by taking the name of a generic type declaration without an accompanying type argument list."

Traduzindo para o Português significa que *Raw type* é um tipo de referência construído por instanciar uma **declaração do tipo**

genérica sem a lista de tipos de argumento. Existe outras classificações para definir uma *Raw Type*, mas que não está no escopo deste artigo.

Perceba na figura 1 que o compilador apresentou um *Warning* ao declararmos `caixa2` mas não impediu a execução do programa, inclusive podemos fazer *casting* ao obtermos o elemento da lista (linha 19). Note também que a lista `caixa2` contém objetos tanto do tipo `TelefoneFixo` quanto `TelefoneCelular`:

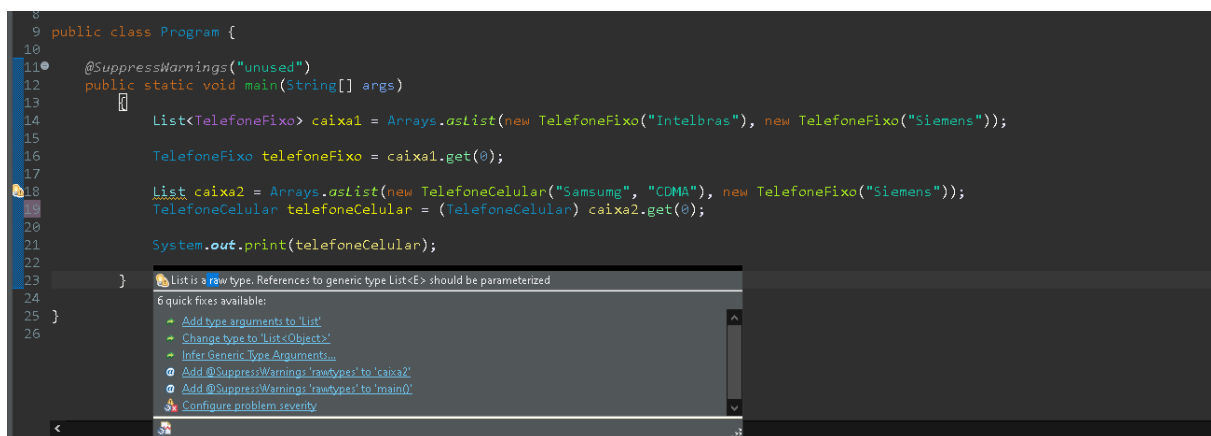


FIGURA 1

O que acontece se executarmos esse código, veja a figura 2 abaixo:

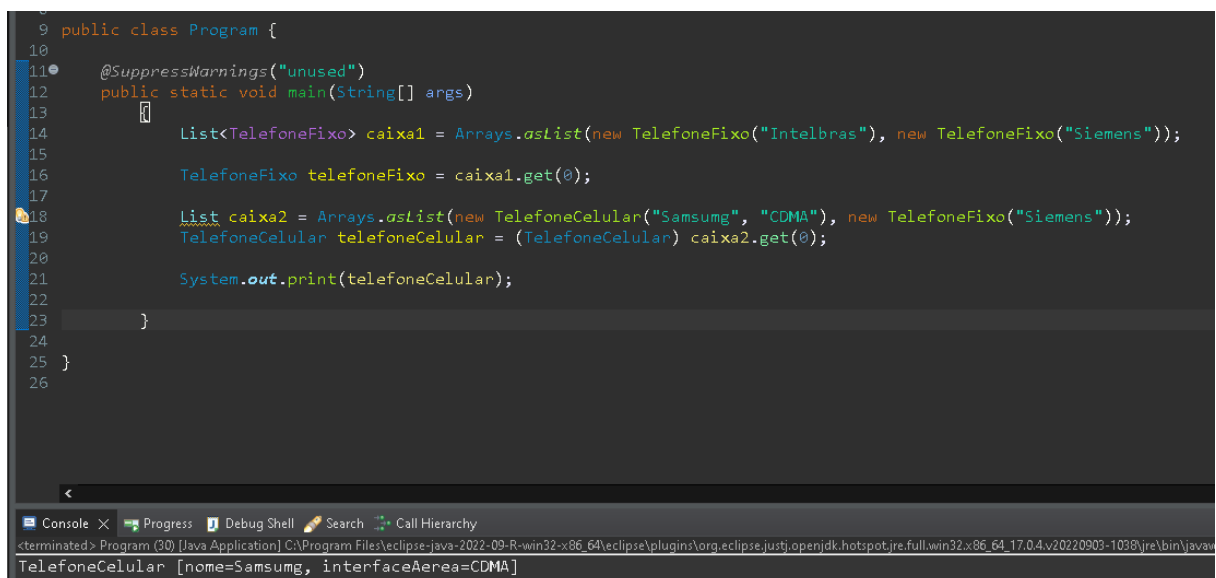
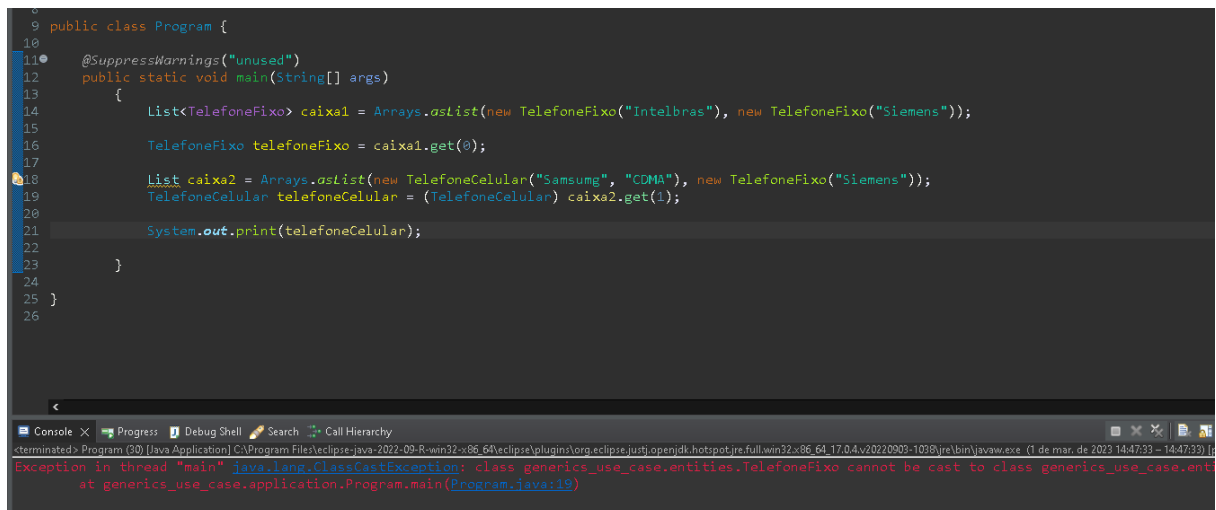


FIGURA 2

Que beleza hein! Imprimiu o elemento 0 da lista sem problemas, mas foi por pura sorte pois este elemento é do tipo `TelefoneCelular`, o que acontece se quisermos imprimir o 2o elemento da lista, a figura 3 responde sua pergunta:



```
9 public class Program {
10
11     @SuppressWarnings("unused")
12     public static void main(String[] args)
13     {
14         List<TelefoneFixo> caixa1 = Arrays.asList(new TelefoneFixo("Intelbras"), new TelefoneFixo("Siemens"));
15
16         TelefoneFixo telefoneFixo = caixa1.get(0);
17
18         List caixa2 = Arrays.asList(new TelefoneCelular("Samsung", "CDMA"), new TelefoneFixo("Siemens"));
19         TelefoneCelular telefoneCelular = (TelefoneCelular) caixa2.get(1);
20
21         System.out.print(telefoneCelular);
22
23     }
24 }
25
26
```

Console X Progress Debug Shell Search Call Hierarchy

terminated> Program (30) [Java Application] C:\Program Files\Eclipse-java-2022-09-R-win32-x86_64\eclipse\plugins\org.eclipse.justjopenjdk.hotspot.jre.full.win32-x86_64.17.0.4.v20220903-1038\jre\bin\javaw.exe (1 de mar. de 2023 14:47:33) [j

Exception in thread "main" java.lang.ClassCastException: class generics_use_case.entities.TelefoneFixo cannot be cast to class generics_use_case.ent

at generics_use_case.application.Program.main(Program.java:19)

FIGURA 3

Runtime exception!! Obviamente o programa não sabe o que fazer quando vc quer atribuir a uma variável do tipo `TelefoneFixo` outra variável do tipo `TelefoneCelular`. Apesar de vc poder operar com *Raw types*, não existe uma forma do compilador saber antes do programa ser executado se as os tipos de variáveis são compatíveis, e portanto o controle deve ser feito por quem gera o código, convenhamos é adicionar um risco (desnecessário) de ocorrer problemas.

E ao trabalharmos com a lista `caixa1` ? Como definimos que tipo de variável a lista vai conter, o próprio compilador vai fazer essa verificação lançando um erro. Veja na figura 4 abaixo que o compilador abriu o bico! Não permitiu atribuir a variável pois existe uma incompatibilidade de tipo, a variável `List<TelefoneFixo> caixa1` aceita apenas variáveis do tipo `TelefoneFixo`

```

10
11 • @SuppressWarnings("unused")
12 public static void main(String[] args)
13 {
14     List<TelefoneFixo> caixa1 = Arrays.asList(new TelefoneFixo("Intelbras"), new TelefoneFixo("Siemens"));
15
16     TelefoneFixo telefoneFixo = caixa1.get(0);
17
18     List caixa2 = Arrays.asList(new TelefoneCelular("Samsung", "CDMA"), new TelefoneFixo("Siemens"));
19     TelefoneCelular telefoneCelular = (TelefoneCelular) caixa1.get(1);
20
21     System.out.print(telefoneCelular);
22
23 }
24
25 }
26

```

Cannot cast from TelefoneFixo to TelefoneCelular

FIGURA 4

Portanto o uso de *Generics* evita o uso excessivo de casts e também problemas de incompatibilidade de tipos, tornando seu código mais robusto. Este artigo é apenas uma introdução ao uso de *Generics* e portanto está longe de extinguir o tema, a documentação oficial

<https://docs.oracle.com/javase/tutorial/java/generics/index.html> pode lhe ajudar a entender melhor como utilizar este recurso do Java.