El trabajo con los tipos Genéricos en Java es algo de lo más habitual , pero siempre hay situaciones en las que aparecen dudas. Una de las cosas más importantes que hay que entender hablando de Genéricos es que se trata realmente de “sintaxis sugar” es decir no existen realmente en la JVM. Esto es debido a que en su momento se llego a la conclusión que implementarlos directamente en la máquina virtual era muy costoso. Vamos a ver un ejemplo usando el concepto de Java Generics Erasure, supongamos que tenemos el siguiente código:

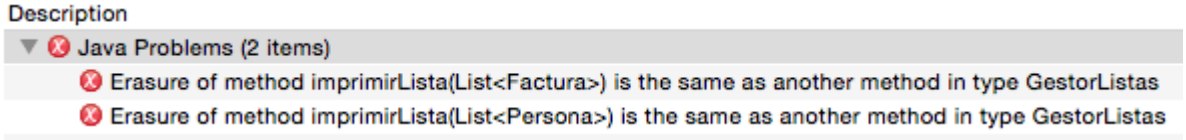


```
package com.arquitecturajava.generics;

import java.util.List;

public class GestorListas {

 public void imprimirLista(List&amp;lt;Persona&amp;gt; listaPersonas)
{

 for (Persona p: listaPersonas) {
 System.out.println(p);
 }

 }
```

```
 public void imprimirLista(List&amp;lt;Factura&amp;gt; listaFacturas)
{

 for (Factura p: listaFacturas) {
 System.out.println(p);
 }

 }

}
```

# Java Generics Erasure Problema

Estamos realizando una sobrecarga estática o overloading , eso sí con la particularidad de que trabajamos con tipos genéricos. Lamentablemente el código no compila y muestra un mensaje.

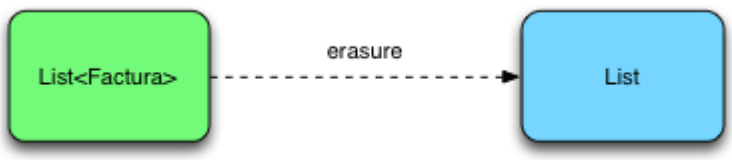



El mensaje tiene mucho que ver con Java Generics Erasure y la sintaxis sugar. Lo que esta pasando realmente es que la JVM no puede trabajar directamente con este código y se ve obligada a convertirle a este otro.

```
package com.arquitecturajava.generics;

import java.util.List;
```

```
public class GestorListas {

 public void imprimirLista( List listaPersonas) {

 for (Object p: listaPersonas) {
 System.out.println(((Persona)p).getNombre());
 }

 }

 public void imprimirLista(List listaFacturas) {

 for (Object f: listaFacturas) {
 System.out.println(((Factura)p).getConcepto());
 }

 }

}
```

Recordemos que dentro de la JVM los genéricos no existen por lo tanto nos encontramos con un problema de sobrecarga ya que los métodos tienen el mismo nombre y mismo tipo de argumentos. Esto es algo que nosotros no vemos en nuestro código pero que cuando el compilador lo modifica para que la JVM pueda entenderlo aparece.




Tengamos muy en cuenta este tipo de situaciones cuando trabajamos con Java Generics y nos ayudará a entender mejor como funcionan realmente las cosas.

Otros artículos relacionados:

Introducción Java Generics

Java Generics WildCard

Java Generics Oracle