Erdianti Wiga Putri Andini
13522053 – K1

TUGAS BACA
TYPE ERASURE

Generic dalam C# memastikan keamanan tipe dengan mengizinkan collection seperti `List<Person>` hanya untuk menyimpan objek `Person`, menghilangkan kebutuhan untuk pengecoran tipe pada waktu runtime. Ini dicapai melalui kompiler C# yang menghasilkan kode khusus untuk setiap tipe generic dalam perakitan .NET, meningkatkan kinerja dan memberikan manfaat metadata seperti IntelliSense.

Diperkenalkan dalam C# 2.0, generic tidak kompatibel dengan versi sebelumnya (1.0 dan 1.1), memerlukan konversi kode untuk interaksi dengan collection generic yang lebih baru, meskipun ini jarang terjadi karena sifat progresif pengembangan perangkat lunak.

Java juga menggunakan generic, seperti `ArrayList<Person>`, untuk mencegah penambahan tipe yang salah ke dalam collection. Namun, Java menggunakan penghapusan tipe untuk menghilangkan informasi tipe spesifik pada runtime, yang mempertahankan kompatibilitas ke belakang tetapi memerlukan pengecoran eksplisit saat mengambil objek, mengakibatkan penurunan kinerja.

Berbeda dengan C# dan Java, templat C++ berfungsi serupa dengan generic C# tetapi dikompilasi menjadi kode khusus, pada dasarnya membuat kelas kustom. Tidak seperti C# dan Java, C++ tidak bergantung pada mesin virtual, memungkinkan fleksibilitas dan kinerja yang lebih besar tanpa perlu boxing atau unboxing.

Di C# dan Java, generic memerlukan definisi metode eksplisit untuk tipe kelas untuk mendikte perilaku seperti penambahan, biasanya diberlakukan melalui antarmuka. Namun, C++ tidak memiliki batasan ini dengan templatnya, memungkinkan operasi apa pun yang didukung oleh tipe yang disertakan, sehingga menawarkan fleksibilitas pemrograman yang lebih besar tanpa kendala mesin virtual.

Implementasi generic atau templat setiap bahasa mencerminkan arsitektur sistem dan filosofi desainnya, menyeimbangkan keamanan tipe, kinerja, dan kompatibilitas ke belakang. Penghapusan tipe dalam Java adalah mekanisme yang memastikan batasan tipe pada waktu kompilasi sambil membuang informasi tipe pada runtime, memfasilitasi kompatibilitas ke belakang dengan versi Java yang lebih tua. Proses ini penting bagi pengembang yang berurusan dengan generic di berbagai komponen sistem atau mempertahankan kompatibilitas dengan aplikasi Java warisan.

Misalnya, metode generic di Java yang memeriksa elemen array awalnya ditulis dengan tipe `E`. Kompiler Java menggantikan `E` dengan `Object` pada waktu kompilasi untuk memastikan keamanan tipe, menghilangkan informasi tipe generic setelah kompilasi.

Erdianti Wiga Putri Andini
13522053 – K1

Penghapusan tipe mempengaruhi generic tingkat kelas dan metode. Pada tingkat kelas, parameter tipe generic seperti `E` dalam `Stack<E>` diganti dengan `Object` atau tipe terikat pertama selama kompilasi.

Jika parameter tipe generic terikat, seperti `E extends Comparable<E>`, itu digantikan dengan `Comparable` oleh kompiler. Penggantian ini memastikan kompatibilitas dengan versi Java sebelum generic.

Penghapusan tipe tingkat metode serupa menggantikan tipe yang tidak terikat dengan `Object` dan tipe terikat dengan kelas terikat pertama mereka, seperti `Comparable`. Kasus tepi dalam penghapusan tipe dapat menyebabkan kesalahan runtime, seperti penyisipan yang salah dari `String` ke dalam `IntegerStack`, yang tidak menyebabkan kesalahan waktu kompilasi karena penghapusan tipe tetapi mengakibatkan `ClassCastException`.

Untuk mengatasi masalah tersebut, kompiler Java membuat metode jembatan—metode sintetis yang mempertahankan polimorfisme tipe generic setelah penghapusan. Metode jembatan ini memastikan konsistensi tanda tangan metode di seluruh implementasi kelas generic.

Misalnya, kelas `IntegerStack` yang memperluas generic `Stack<Integer>` akan memiliki metode jembatan untuk `push(Object)` yang secara internal memanggil `push(Integer)`, memastikan konsistensi tipe meskipun penghapusan menjadi `Object`.

Mekanisme ini memungkinkan generic Java untuk berintegrasi dengan lancar dengan kode non-generic yang lebih tua, meskipun itu memperkenalkan beberapa overhead kinerja dan potensi kesalahan tipe runtime karena hilangnya informasi tipe. Sumber daya komprehensif seperti Spesifikasi Bahasa Java menawarkan informasi terperinci tentang penghapusan tipe dan generic bagi mereka yang mencari wawasan lebih dalam.