

Generics parametreleştirilmiş tipler anlamına gelir ve sınıf, interface veya metod yazarken tek bir data tipine bağlı kalmadan üzerinde işlem yapacağımız  data tipini parametre olarak alabilmemizi sağlar. Bu sayede, farklı data tipleri üzerinde çalışan tek bir sınıf, metod veya interface yazmak mümkündür.

Bu şekilde yazılan sınıflara **generic sınıf**, metotlara **generic metod** denir.

Generics hataları azaltmak ve tip güvenliğini (type-safety) sağlamak amacıyla JDK 5 ile eklenmiştir. Buna rağmen, Java’nın en temel özelliklerinden biridir ve dili temelden etkilemiştir. Bu yüzden, jenerikleri iyi anlamak  Java’yı öğrenmek açısından büyük önem taşır.

Generic class, metod veya interface oluştururken data tipini parametre olarak almak için elmas(diamond) işareti <> kullanılır ve çoklu parametre(multiple parameter) kullanılabilir. Parametreler genelde T, S, U, V gibi harfleri ile gösterilir.

Genericsde primitive(ilkel) data tipleri parametre olarak kullanılamaz.

**Generic Class**

Generic bir sınıfın genel tanımı aşağıdaki gibidir:

class class\_name<T> {

sınıf değişkenleri;

…..

sınıf metodları; }

Sınıf tanımlandıktan sonra, istediğiniz herhangi bir data tipinde aşağıdaki gibi nesneler oluşturabilirsiniz:

class\_name <T> obj = new class\_name <T> ();

Örneğin, integer data tipi için nesne şöyle olacaktır:

class\_name <Integer> obj = new class\_name<Integer>;

**Generic Metod**

Generics ile farklı data tiplerini parametre olarak alan ve döndüren metod yazabilirsiniz.

public static <T> void yazdir(T type){

System.out.print(type); }

Generics ile data tipleri üst veya alt tip sınırlanabilir.

**Java Generics Wildcard**

Javada, wildcard(?) genelde genericsde parametre olarak bilinmeyen data tiplerine atıfta bulunmak için kullanılır.