Garbage Collector: Programın çalışması sırasında kullanılmayan bellek alanlarını otomatik olarak serbest bırakmaya ve bellek sızıntılarını engellemeye yarayan bir bellek yönetim sistemidir. Çöp toplayıcı olarak adlandırılır. Garbage Collector bellek yönetimini üstlenir ve kullanıcıların manuel bellek yönetimi yapmalarına gerek kalmaz.

Garbage Collector'ın Çalışma Prensibi:

GC, program çalışırken belleği izler ve kullanılmayan nesneleri otomatik olarak tespit eder. Bu tespit işlemi sonrasında, bu nesnelerin işgal ettiği bellek alanını serbest bırakır.

Bellek Yönetimindeki Rolü:

Otomatik Bellek Yönetimi:

Garbage Collector, bellek yönetimini otomatikleştirir, yani kullanıcıların belleği manuel olarak serbest bırakmalarına gerek yoktur. Bu, bellek sızıntılarını önler ve yazılımın kararlılığını artırır.

Bellek Sızıntılarının Önlenmesi:

GC, erişilemeyen nesneleri tespit edip otomatik olarak temizleyerek bellek sızıntılarını engeller. Bu sayede programcılar bellek yönetimi konusunda fazla endişelenmezler.

Performans ve Verimlilik:

GC, performans üzerinde doğrudan etkili olabilir. Çünkü zaman zaman çalışarak bellek toplama işlemi gerçekleştirir. Bu nedenle, çok sık çöp toplama yapılırsa programın performansı düşebilir.
Bununla birlikte, bellek alanını düzenli olarak temizleyerek uzun vadede bellek verimliliğini artırır.
GC'nin Çalışma Zamanı:

Garbage Collector, belirli aralıklarla veya bellek tükendiğinde çalışır. Çöp toplama işlemi **otomatik olarak tetiklenir** ve genellikle aşağıdaki şartlar altında çalışır:

Bellek tükendiğinde veya belirli bir sınır aşıldığında GC devreye girer.
Genç nesne toplama (Young Generation): Yeni nesneler için çöp toplama.
Yaşlı nesne toplama (Old Generation): Uzun süre bellekte kalan nesneler için çöp toplama.

Bu işlemler sırasında GC, bazı nesneleri yerinde temizler (yerinde sıkıştırma), bazılarını ise tamamen serbest bırakır.

Generic Yapıları ve Avantajları:

Generic Yapıları: Bağımsız kod yazmayı sağlayan bir özelliktir. Generic yapılar, türlerin parametre olarak alındığı sınıflar ve metodlarla esnek programlamaya olanak tanır. Bu özellik, belirli bir türün ne olduğunu belirlemek yerine, tür bağımsız olarak işlem yapmanıza imkan verir.

Avantajları:

Tür Güvenliği:

Generics, kodun çalışma zamanında değil, derleme zamanında tür güvenliğini sağlar. Bu, yanlış türlerin kullanılmasını engeller ve hataların önüne geçer.

Performans İyileştirmesi:

Generics, Boxing ve Unboxing gibi tür dönüşümleri gerektirmediğinden, bellek ve işlem maliyetlerini düşürür. Bu, yazılımın daha verimli çalışmasını sağlar.

Esneklik ve Yeniden Kullanılabilirlik:

Generic yapılar, farklı türlerle kullanılabilen kod parçacıkları oluşturmanıza olanak tanır. Bu, aynı sınıf veya metodu birden fazla türle çalışacak şekilde kullanmayı sağlar.

Kodu Temiz ve Anlaşılır Yapma:

Tür bağımsız kod yazmak, kodun daha temiz, bakımı daha kolay ve hatalara karşı daha dayanıklı olmasına yardımcı olur.

Generic Sınıf ve Metotlarla Esnek Programlama:

Generic sınıflar ve metotlar: tür bağımsız olarak çalışan yapılar sağlar. Bu tür yapılar, aynı kodu farklı veri türleriyle kullanmanıza olanak tanır. Kullanıcı, türü belirtmek yerine türün parametre olarak geçmesini sağlar.

Esnek Programlama:

Generic Sınıflar: Tür parametreleri alarak, her türle çalışabilen sınıflar yazmanızı sağlar. Bu sınıflar, belirli bir tür yerine herhangi bir tür ile çalışabilir.
Generic Metodlar: Benzer şekilde, tür parametresi alan metodlar, yalnızca belirli türlerle değil, birçok farklı türle işlem yapabilir.

Bu, tekrar kullanılabilirliği artırır ve aynı kodu farklı veri türleriyle kullanma esnekliği sağlar.

Boxing ve Unboxing:

Boxing ve **Unboxing**, değer türlerinin nesne türlerine (veya tam tersi) dönüştürülmesi işlemleridir.

Boxing:

Değer türündeki bir değişkenin, bir nesne türüne dönüştürülmesidir. Bu, değerin kutuya yerleştirilmesi gibi bir işlemi ifade eder. Boxing işlemi genellikle bellek ve işlem gücü açısından ek yük oluşturur.

Unboxing:

Boxing işleminde kutuya yerleştirilen değerin, orijinal değer türüne geri dönüştürülmesidir. Bu, değerin yeniden çıkarılması ve orijinal türüne dönüştürülmesidir.

Performansa Etkisi:
Boxing ve Unboxing işlemleri, ek bellek kullanımı ve işlem maliyeti yaratır. Değer türleri nesne türlerine dönüştürülürken, bu işlemler bellek kopyalamalarına yol açar.
Generics kullanıldığında, bu tür dönüştürmeler yapılmaz, çünkü türler doğrudan işlenir. Bu da performansı artırır, çünkü daha az bellek kullanımı ve işlem süresi gereklidir.