

**Thread nedir?**

Her bir işlemin altında çalışan alt işlemlere "thread" adı verilir. Aynı anda birden fazla işlem yapmayı sağlayan yapıya thread denir. Bu yapı sayesinde işlemler birbirlerini beklemeden kendi işlemini yapar. Kullanıcı bir form üzerinden web isteği başlattığında web isteği cevap verene kadar kullanıcı form üzerinde işlem yapamayacaktır.

Benzer şekilde ağ programlama işlemleri sırasında karşı taraftan bir cevap beklenmeye alındığında program üzerinde işlem yapılmayacaktır. Bu ve bunun gibi durumlarda thread yapısının kullanımı iyi bir kullanıcı deneyimi için faydalı olacaktır.

Aşağıdaki örnek thread kullanılmadan yapılmıştır.

public class ThreadOrnegi {

public static void main(String[] args) {

        uzunBirIslem();

        System.out.println("Merhaba Thread");

    }

private static void uzunBirIslem() {

        try {

            // Burada uzun bir işlem yapılıyor.

            Thread.sleep(5 \* 1000);

            System.out.println("Uzun işlem sonucu");

        } catch (InterruptedException ex) {

            System.err.println(ex);        }   } }

Örnekte basit bir Merhaba Thread yazısının yazılması için uzunBirIslem metodunun bitmesi beklenmektedir.

**Thread kullanımı**

Thread kullanımı için Thread sınıfını kalıtım almak(extends) veya Runnable arayüzünü uygulamak(implements) olmak üzere iki yöntem kullanılır.

Thread sınıfının kullanımı için Thread sınıfı kalıtım alındıktan sonra run metodu ezilir(override) ve gerekli komutlar yazılır.

 public class ThreadOrnegi extends Thread {

public static void main(String[] args) {

        ThreadOrnegi threadOrnegi = new ThreadOrnegi();

        threadOrnegi.start();

        System.out.println("Merhaba Thread");    }

@Override

    public void run() {

        try {

            // Burada uzun bir işlem yapılıyor.

            Thread.sleep(5 \* 1000);

            System.out.println("Uzun işlem sonucu");

        } catch (InterruptedException ex) {

            System.err.println(ex);

        }    } }

NOT: main metodu da bir thread oluşturur.

Runnable arayüzünün kullanımı için arayüz metotları sınıf tarafından uygulanır(implements).

Arayüz uygulandıktan sonra Thread sınıfının kurucusuna sınıf parametre olarak geçilerek thread çalıştırılır.

 public class ThreadOrnegi implements Runnable {

public static void main(String[] args) {

        Thread t1 = new Thread(new ThreadOrnegi());

        t1.start();

        System.out.println("Merhaba Thread");    }

@Override

    public void run() {

        try {

            // Burada uzun bir işlem yapılıyor.

            Thread.sleep(5 \* 1000);

            System.out.println("Uzun işlem sonucu");

        } catch (InterruptedException ex) {

            System.err.println(ex);

        }    } }

Her ikisi arasında herhangi bir fark yoktur.

NOT: Java çoklu kalıtımı desteklemediğinden Runnable arayüzünün kullanımı faydalı olacaktır.

**Multithreading Nedir?**

Java’da çoklu thread, maksimum CPU kullanımıyla aynı anda iki veya daha fazla iş parçacığı yürütme işlemidir.

Çok iş parçacıklı uygulamalar, iki veya daha fazla iş parçacığının aynı anda çalıştığı yerlerdir; Bu nedenle, Java’da eşzamanlılık olarak da bilinir. Bu çoklu görev, birden fazla işlem CPU, bellek vb. ortak kaynakları paylaştığında yapılır.