

Java (programlama dili)

**Java**, [Sun Microsystems](https://tr.wikipedia.org/wiki/Sun_Microsystems) [mühendislerinden](https://tr.wikipedia.org/wiki/M%C3%BChendislik) [James Gosling](https://tr.wikipedia.org/wiki/James_Gosling) tarafından geliştirilmeye başlanmış [açık kaynak kodlu](https://tr.wikipedia.org/wiki/A%C3%A7%C4%B1k_kaynak_kodlu_yaz%C4%B1l%C4%B1m), nesneye yönelik, platform bağımsız, yüksek verimli, çok işlevli, yüksek seviye, hem [yorumlanan](https://tr.wikipedia.org/wiki/Yorumlanan_programlama_dili) hem de derlenen bir dildir.[[2]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-2)

Java, Sun Microsystems'den James Gosling tarafından geliştirilen bir programlama dilidir ve 1995 yılında Sun Microsystems'in çekirdek bileşeni olarak piyasaya sürülmüştür. Bu dil [C](https://tr.wikipedia.org/wiki/C_(programlama_dili)) ve [C++](https://tr.wikipedia.org/wiki/C%2B%2B)'dan birçok sözdizim türetmesine rağmen bu türevler daha basit nesne modeli ve daha az düşük seviye olanaklar içerir. Java uygulamaları [bilgisayar mimarisine](https://tr.wikipedia.org/wiki/Bilgisayar_mimarisi) bağlı olmadan herhangi bir [Java Sanal Makinesi](https://tr.wikipedia.org/wiki/Java_sanal_makinesi) (Java Virtual Machine - *JVM*) üzerinde çalışabilen tipik [bytecode](https://tr.wikipedia.org/wiki/Bytecode" \o "Bytecode)'dur (sınıf dosyası).

Java'nın sık kullanılan sloganlarından biri olan, çevirisi "bir defa yaz, her yerde çalıştır" olan "write once, run anywhere - *WORA*",[[3]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-3) Java'nın; derlenmiş Java kodunun, Java'yı destekleyen bütün platformlarda tekrar derlenmeye ihtiyacı olmadan çalışabileceğini ima eder.[[4]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-:0-4) 2016 yılında bildirilen 9 milyon geliştiricisi ile, özellikle istemci sunucu web uygulamaları için olmak üzere, kullanımda olan en popüler programlama dillerinden birisidir.[[5]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-5)[[6]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-:1-6)[[7]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-7)[[8]](https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-8)

Java ilk çıktığında daha çok küçük cihazlarda kullanılmak için tasarlanmış ortak bir düzlem dili olarak düşünülmüştü. Ancak düzlem bağımsızlığı özelliği ve tekbiçim kütüphane desteği [C](https://tr.wikipedia.org/wiki/C_programlama_dili) ve [C++](https://tr.wikipedia.org/wiki/C%2B%2B)'tan çok daha üstün ve güvenli bir yazılım geliştirme ve işletme ortamı sunduğundan, hemen her yerde kullanılmaya başlanmıştır. Özellikle kurumsal alanda ve mobil cihazlarda son derece popüler olan Java özellikle J2SE 1.4 ve 5 sürümü ile masaüstü uygulamalarda da yaygınlaşmaya başlamıştır.

Java'nın ilk sürümü olan Java 1.0 (1995) Java Platform 1 olarak adlandırıldı ve tasarlama amacına uygun olarak küçük boyutlu ve kısıtlı özelliklere sahipti. Daha sonra düzlemin gücü gözlendi ve tasarımında büyük değişiklikler ve eklemeler yapıldı. Bu büyük değişikliklerden dolayı geliştirilen yeni düzleme Java Platform 2 adı verildi ama sürüm numarası 2 yapılmadı, 1.2 olarak devam etti. 2004 sonbaharında çıkan Java 5, geçmiş 1.2, 1.3 ve 1.4 sürümlerinin ardından en çok gelişme ve değişikliği barındıran sürüm oldu. Java SE 8 ise Java teknolojisinin günümüz sürümüdür. 13 Kasım 2006'da Java düzlemi [GPL](https://tr.wikipedia.org/wiki/GPL) ruhsatıyla açık kodlu hale gelmiştir.

## **Çalışma mantığı**

Bir Java yazılımı şu şekilde geliştirilir;

* Yazılımcı Java kodunu yazar.
* Bu kod bir Java [derleyicisi](https://tr.wikipedia.org/wiki/Derleyici) ile derlenir. Derleme sonucunda **"bytecode"** adı verilen bir tür sanal makine kodu ortaya çıkar. Platform bağımsızlığını sağlayan bytecode'dur. Çünkü bir kere bytecode oluştuktan sonra yazılım, sanal makine içeren tüm işletim sistemlerinde çalışabilmektedir.
* Bu bytecode [**Java Sanal Makinesi**](https://tr.wikipedia.org/wiki/Java_sanal_makinesi)**(Java Virtual Machine - *JVM*)** tarafından işletilir. Bu aşama, her bir bytecode komutunun teker teker yorumlanması ile icra edilebileceği gibi, **anında derleme ([Just-in-time compilation](https://tr.wikipedia.org/w/index.php?title=Just-in-time_compilation&action=edit&redlink=1" \o "Just-in-time compilation (sayfa mevcut değil)) - *JIT*)** kullanılarak da gerçekleştirilebilir.

## **Bazı kavramlar**

Nesneye yönelik yazılımlama mantığı;

### **Anında Derleme**

Java ilk çıktığında bytecode işletme hızı çok iyi değildi. Yerine göre sistemin öz yazılımlarından 5-10 kat yavaş çalışıyordu. Bu nedenle bazı yazılım geliştirme şirketleri JIT yani "Just-in-time compilation", "anında derleme" araçları üretmeye başladılar. Yapılan şey bytecode'u sanal makinenin kurulu olduğu gerçek sistemin diline anında derleme yaparak dönüştürmesiydi. Bu sayede verimde ciddi artışlar sağlandı. Ama 2000 yılından sonra geliştirilen sanal makinelerde (HotSpot gibi) JIT'in işlevi VM içinde yer almaya başlamış, işlemci hızı ve bellek miktarının dramatik biçimde artması ile dış JIT yazılımları popülerliğini kaybetmiştir. Bugün hâlen birkaç ürün (Excelsior JET gibi) pazarda bulunsa da genellikle bu yöndeki ihtiyaç azalmıştır.

### **Java API**

Java API, Java yazılımlarında kullanılan yazılım kütüphanelerine genel olarak verilen isimdir. Java API ile disk, grafik, ağ, veri tabanı, güvenlik gibi yüzlerce konuda kullanıcılara erişim imkânı sunulur. Java API J2SDK'nın bir parçasıdır.

### **Atık Veri Toplayıcı**

Atık veri toplama teknolojisi (Garbage Collection) Java'dan önce de var olan ama Java ile adını duyurmuş ve yaygın olarak kullanılmaya başlanmış bir kavramdır. C++, C gibi dillerin en büyük engellerinden birisi dinamik bellek yönetimidir. Yazılımda gösterici (işaretçi - *pointer*) kullanarak dinamik olarak bellek ayırdıktan sonra o bellek ile işiniz bittiğinde mutlaka ayrılan belleği bellek yöneticiye özel altyordamlar yardımıyla (delete, free...) iade etmeniz gerekir. Yoksa bellek sızıntısı (*memory leak*) oluşur ve bu bir süre sonra yazılımın ve işletim sisteminin beklenenden farklı davranmasına yol açabilir. Sızıntıların saptanması oldukça güçtür ve bulunması zor hatalara yol açar. Bu nedenle bugünün tüm büyük C ve C++ yazılımları az da olsa bellek sızıntısı içerir (işletim sistemleri dahil).

Atık veri toplayıcı sayesinde Java'da bir nesne oluşturulduktan sonra o nesne ile işiniz bittiğinde hiçbir şey yapmanız gerekmez: [Sanal makine](https://tr.wikipedia.org/wiki/Sanal_makine) akıllı bir biçimde kullanılmayan bellek bölümlerini belirli aralıklarla ya da uyarlamalı yöntemlerle otomatik olarak temizler ve sisteme iade eder. Bu işleme çöp toplama (atık toplama - *garbage collection*) adı verilir. Çöp toplama sistemlerinin yapısı oldukça karmaşıktır ve geçen yıllar içinde büyük gelişmeler kaydedilmiştir. Çöp toplayıcının varlığı Java'da bellek sızıntısı olmayacağı anlamına gelmez, ama bellek sızıntıları daha ender olarak ve farklı şekillerde karşınıza çıkar ve genellikle tedavi edilmesi daha kolaydır.

### **Jar**

Jar (Java Arşivi - *Java Archive*), aslında bir tür sıkıştırma formatıdır. Jar ile derlenen Java kodları ile oluşan yazılımın paketlenip taşınması kolay bir hale getirilir. Jar dosyaları temelde bytecode blokları içerir. Jar dosyaları genellikle kütüphane oluşturmada ya da uygun biçimde hazırlanırsa işletim sisteminden doğrudan çalıştırılabilecek bir şekilde kullanılabilir (çalıştırılabilir jar - *executable jar*). Jar dosyalarının içeriğini sıkışıtırma yazılımları ya da java yazılım geliştirme araçları ile inceleyebilirsiniz. Java 1.5 ile yeni bir tür jar oluşturma metodu da kullanıma girdi. Pack200 adı verilen hiper-sıkıştırma (*hyper-compression*) algoritması ile jar dosyaları daha küçük boyutlara indirgenebiliyor. Ancak bu teknoloji daha çok ağ üzerinden yapılan transferlerde kullanılıyor. Daha fazla ayrıntı için: [buraya bakınız](http://ta-java.blogspot.com/2011/08/arsiv-dosyalar.html) 24 Ocak 2012 tarihinde [Wayback Machine](https://tr.wikipedia.org/wiki/Wayback_Machine" \o "Wayback Machine) sitesinde [arşivlendi](https://web.archive.org/web/20120124051343/http:/ta-java.blogspot.com/2011/08/arsiv-dosyalar.html).

### [**AWT**](https://tr.wikipedia.org/wiki/AWT)**ve Swing**

[AWT](https://tr.wikipedia.org/wiki/AWT), ilk Java ile birlikte geliştirilen temel grafik arayüz oluşturma kütüphanesine verilen isimdir. AWT işletim sisteminin doğal grafik yapılarına erişimi sağlar. Ancak Java 2 platformu ile birlikte AWT yetersiz görülmüş ve çok daha geniş ve gelişmiş özelliklere sahip Swing kütüphanesi sisteme eklenmiştir. Özellikle çok platform destekleyen yazılımlarda kullanıcı arayüzü geliştirme aracı olarak swing hâlen önemini korumaktadır. Swing önceleri işletim sisteminin kullandığı donanım grafik hızlandırma araçlarını kullanmadığından yavaşlığı ile eleştirilere hedef olmuştu. Özellikle Java 1.4 ile Swing, hem genel olarak sanal makinenin hızlanması ve kısmen donanım hızlandırmayı kullanması ile bu kötü şöhretinden sıyrılmaya başladı. [AWT](https://tr.wikipedia.org/wiki/AWT) hâlen Swing'in bir alt katmanında, temel iki boyutlu grafik işlemlerinde kullanılmaya devam ediyor.

### **SWT**

SWT Swing'e bir alternatif olarak [IBM](https://tr.wikipedia.org/wiki/IBM) tarafından geliştirilen bir gösterim sistemidir. Swing'den en büyük farkı çalıştığı işletim sisteminin grafik kütüphanesi ve komutlarını kullanmasıdır. Bu nedenle SWT uygulamaları Swing'e göre çoğu yerde daha hızlı ve işletim sistemindeki diğer uygulamaları andıran bir şekilde çalışmasını sağlar. Swing'in Java 1.5 ile performans açığını kapattığı iddia edilse de SWT'nin de artık Java camiasında kabul görmüş bir sistem olduğu açıktır. SWT'nin dezavantajı ise Java'nin bir parçası olmamasıdır. Yani SWT uygulamaları SWT kütüphanesi ile birlikte dağıtılmaktadır. Ayrıca SWT farklı işletim sistemlerinde farklı olgunluk ve performansta işlemektedir ve özellikle Windows dışı sistemlerde henüz yeterince olgunlaşamamıştır. En bilinen SWT uygulamaları, ünlü Java yazılım geliştirme aracı (*IDE*) [Eclipse](https://tr.wikipedia.org/wiki/Eclipse_(yaz%C4%B1l%C4%B1m)" \o "Eclipse (yazılım)) ve [BitTorrent](https://tr.wikipedia.org/wiki/BitTorrent" \o "BitTorrent) uygulaması [Azureus](https://tr.wikipedia.org/wiki/Azureus" \o "Azureus)'tur.

### **HotSpot Teknolojisi**

Sun Microsystems tarafından geliştirilen Java sanal makinesi, HotSpot adı verilen özel bir teknolojiyi içinde barındırır. HotSpot, yani "sıcak nokta", bir yazılımda sürekli olarak tekrarlanan ve üzerinden geçilen kod bölümlerine verilen bir isimdir. HotSpot sanal makinesi şu anda iki ayrı modda sanal makinenin çalışmasını sağlamaktadır: İstemci (*Client*) ve Sunucu (*Server*) modları. İstemci modunda bytecode büyük ölçüde daha başlangıçta JIT ile sistemin öz makine koduna dönüştürülerek işletilir ama çalışma anında daha fazla iyileştirme işlemi gerçekleştirilmez. Bir uygulamanın hızı istemci modunda zaman içinde bu nedenle değişmez. Sunucu modunda ise sanal makine başlangıçta bytecode'unu sistemin öz koduna dönüştürmekte acele etmez. Bu nedenle sunucu modu başlangıçta istemci modundan oldukça yavaştır. Yazılım çalışmaya devam ettikçe sanal makine yazılımdaki sıcak noktaları tespit edip bytecode'u sadece JIT ile makine koduna dönüştürmekle kalmaz, ayrıca oldukça yoğun bir iyileştirmeye de tabi tutar. Sonuçta sunucu modunda uygulamalar zaman içinde hızlanır, uzun soluklu uygulamaların bu nedenle sunucu modunda işletilmesi önerilir. Mustang kod adlı Java SE 6'da istemci modunda da benzeri bir teknolojinin kullanılması bekleniyor. Dolphin kod adlı Java SE 7'de ise bu iki mod arasındaki fark azalacağından sistemin tek modda çalışması bekleniyor.

### **Uygulamacık**

Uygulamacık ([applet](https://en.wikipedia.org/wiki/Applet) 12 Nisan 2011 tarihinde [Wayback Machine](https://tr.wikipedia.org/wiki/Wayback_Machine" \o "Wayback Machine) sitesinde [arşivlendi](https://web.archive.org/web/20110412105045/http:/en.wikipedia.org/wiki/Applet).), uzaktaki sistem üzerinden indirilip internet tarayıcı üzerinde çalıştırılabilen Java uygulamalarına verilen isimdir. Java'nın son kullanıcılar tarafından tanınması uygulamacık sayesinde olmuştur dersek yanlış olmaz. Uygulamacıklar sisteme zarar veremeyecek bir şekilde tasarlanmıştır. İçerisinde uygulamacık olan bir sayfayı açmaya çalıştığınızda tarayıcınız otomatik olarak Java sanal makinesini çalıştırıp ekranın uygulamacığa ayrılan bölümünde uygulamanın çalışmasını sağlar.

Fakat, son yıllarda birçok Java programcısı Uygulamacıklar üzerinden gözünü çekmiştir. Artık [HTML](https://tr.wikipedia.org/wiki/HTML), [CSS](https://tr.wikipedia.org/wiki/CSS) ve [JavaScript](https://tr.wikipedia.org/wiki/JavaScript" \o "JavaScript) sayesinde çok daha hızlı, pratik, güvenli ve kusursuz uygulamalar/web hizmetleri yapılabiliyor olması bunun en önemli sebebidir.

### **WebStart**

WebStart teknolojisi uygulama kurulum, güncelleme ve silme dertlerine deva olmak üzere tasarlanmış bir sistemdir. WebStart teknolojisi kısaca yazılımların uzaktan yerel sisteme güvenli olarak kurulmasını ve korumalı bir alanda çalıştırılmasını sağlar. Applet'lerin bir sonraki adımı olarak görülebilir. Bir WebStart uygulamasını kurmak için internet üzerindeki özel bir bağlantıya tıklamak yeterlidir. Sistem otomatik olarak WebStart sistemini çalıştırıp yazılımı Java cep belleğine indirir. İstenirse masaüstüne kısayol koymasını da sağlar. Daha sonra sistem çevrimiçi (*online*) ya da çevrimdışı (*offline*) olarak çalıştırılabilir ve uzaktaki yazılım güncellendiğinde otomatik olarak -istenirse- yerel makinedeki yazılımın da güncellenmesi sağlanabilir. Kullanıcının özel olarak izin vermesi halinde uygulama yerel sisteme erişim hakkı kazanabilir. Aksi takdirde WebStart uygulamaları sisteme yazma işlemi gerçekleştiremezler.

Bu Java teknolojisi de, günümüzdeki programcıların takibinde değildir.

## **Java yazımı (Sözdizimi)**

Sınıf temelli nesneye yönelik bir dil olan Java, yazım olarak C++ ile benzerlikler arz eder. Java'nın yanında C#, Perl, JavaScript gibi diller de aynı dil ailesine aittir. "{}" şeklinde süslü parantezler içerisindeki bloklar, ++ arttırma ve — azaltma işleçleri bu dilin belirgin özelliklerindendir.

### **Merhaba** Dünya!

*// MerhabaDünya.java*

**public** **class** **MerhabaDünya**{

**public** **static** void main(String[] args) {

System.out.println("Merhaba Dünya!");

}

}

* "MerhabaDünya.java" ile "public class MerhabaDünya" bu kısımda iki isim aynı olmalıdır aksi takdirde uygulama çalıştırılamaz.
* class: Sınıf tanımlayabilmek için class ön eki şarttır.
* public: Sınıfın dışarıdan erişebilir olduğu,
* static: Sınıf tarafından paylaşıldığı,
* void: Bir değer döndürmediği anlamına gelir.
* Public ve Static'e erişim belirleyicisi (access specifier) de denir.
* Void'e dönüş tipi (return type) de denir.
* String args[]: Parantezin içinde yöntemin aldığı parametreleri belirleriz. "String" sınıf adı, "args" da parametre adıdır. "[]" ise args'ın bir dizi (array) olduğunu belirtiyor.
* "System.out.println();" ile **yazımızı yazdırıyoruz ve yeni satıra geçmesini** sağlıyoruz.
* Yazımızı "System.out.print("Merhaba Dunya!");" ile de yazdırabilirdik. Fakat imleç yeni satıra geçmezdi.
* **Java Türkçe karakterleri (C# gibi) "değişken adlarında, sınıf adlarında" da kullanmamıza imkân tanır.**

Java'da yazdığımız yazılımları derlememiz için öncelikle sınıf adı ile aynı adı taşıyan dosya ismine sahip olmamız gerekmektedir. Yukarıdaki örnek yazılımı sınıf ismi olan "MerhabaDünya" ifadesini kullanıp uzantısı ile beraber "MerhabaDünya.java" ismi ile kaydedebiliriz. J2SDK veya benzer bir Java geliştirme ortamı kurulu sistemimizde yazılan uygulama aşağıdaki şekilde derlenebilir.

javac -encoding UTF-8 MerhabaDünya.java

[javac](https://tr.wikipedia.org/wiki/Javac), yazılan programı derleyerek ".class" uzantılı bir dosya üretir. ".class" sınıf dosyaları JVM'de çalışabilecek bytecode'lar içeren sınıf dosyalarıdır. Örnek uygulamayı çalıştırmak için:

java MerhabaDünya

yazabiliriz. Java komutu öncelikle sınıf yolunda "MerhabaDünya" sınıfını arayacaktır. Bulduğu takdirde "MerhabaDünya" sınıfında "main" metodunu arayacaktır. Eğer metot bulunur ise bu metot icra edilecektir.

### **Kapsamlı örnek**

**package** **TekCift**;

**import** **javax.swing.JOptionPane**;

**public** **class** **TekCift** {

*/\*\**

*\* Tam sayı (int ingilizce de integer’ın kısaltmasıdır)*

*\*/*

**private** int kullanicininGirdigi;

*/\*\**

*\* Bu yapıcı fonksiyondur. TekCift objesi yaratılırken çağrılır.*

*\*/*

**public** TekCift() {

}

*/\*\**

*\* Aşağıdaki ana fonksiyondur. Java yorumlayıcısı programı çalıştırmak için*

*\* ilk bu fonksiyonu çağırır.*

*\**

*\* @param argumanlar*

*\* Komut satırı argümanları (kullanılmıyor)*

*\*/*

**public** **static** void main(**final** String[] argumanlar) {

TekCift obje = **new** TekCift();

obje.pencereGoster();

}

*/\*\**

*\* Aşağıdaki fonksiyon "Sayı giriniz" diyaloğunu gösterir*

*\*/*

**public** void pencereGoster() {

**try** {

kullanicininGirdigi = Integer.parseInt(JOptionPane.showInputDialog("Lütfen bir sayı giriniz."));

hesapla();

} **catch** (**final** NumberFormatException e) {

System.err.println("HATA: Geçersiz bir değer girdiniz. Lütfen sayısal bir değer giriniz.");

}

}

*/\*\**

*\* Aşağıdaki fonksiyon kullanıcının girdiği değere göre çift ya da tek*

*\* penceresi gösterir.*

*\*/*

**private** void hesapla() {

**if** ((kullanicininGirdigi % 2) == 0) {

JOptionPane.showMessageDialog(**null**, "Çift");

} **else** {

JOptionPane.showMessageDialog(**null**, "Tek");

}

}

}

* package anahtar kelimesi dosyanın başında paket belirtmek için kullanılır.
* kullanıcı arayüzü gösterebilmek içın javax.swing.JOptionPane kütüphanesine ihtiyacımız var. Dosyanın başındaki import cümlesi tam bu işe yarar.
* TekCift sınıfı kullanıcının girdiği bir tam sayıyı hafızasında tutmak için kullanıcının girdiği değişkenini tanımlıyor. Diğer sınıfların bu değişkeni görmemesi için private (özel) anahtar kelimesini kullanılıyoruz.