**Java Nedir?**

Java, 1995 yılında Sun Microsystems (şu anda Oracle) tarafından geliştirilen bir programlama dilidir. Java'nın en önemli özelliklerinden biri, yazılımın bir kez yazılıp her platformda çalışabilmesi anlamına gelen "bir kez yaz, her yerde çalıştır" (Write Once, Run Anywhere - WORA) ilkesidir. Bu, Java'nın platform bağımsızlık hedefini gerçekleştirmesini sağlar. Java kodları, Java Virtual Machine (JVM) tarafından çalıştırılır ve JVM sayesinde yazılan kod farklı işletim sistemlerinde değişiklik yapılmadan çalışabilir.

**Java’nın Avantajları**

1. **Platform Bağımsızlık**: Java, derlendikten sonra *bytecode* denilen bir ara kod oluşturur. Bu bytecode, herhangi bir JVM olan sistemde çalışabilir. Bu nedenle, Java kodu platformdan bağımsız olarak çalışır.
2. **Nesne Yönelimli Programlama (OOP)**: Java, OOP ilkelerini tamamen destekleyen bir dildir. Bu sayede kodun modülerliği, yeniden kullanılabilirliği ve bakım kolaylığı sağlanır.
3. **Güvenlik**: Java, güvenlik özellikleriyle tanınır. JVM, uygulamaların izole bir ortamda çalışmasını sağlar ve Java'nın hafıza yönetimi gibi özellikleri, güvenliği artırır.
4. **Çoklu İş Parçacığı Desteği**: Java, çoklu işlem yapmayı destekleyen bir dildir. Aynı anda birden fazla iş yapmayı (multithreading) destekleyerek yüksek performanslı uygulamalar geliştirmenizi sağlar.
5. **Geniş Kütüphane Desteği (Java API)**: Java’nın standart kütüphaneleri, veri yapıları, koleksiyonlar, giriş/çıkış işlemleri, grafik arayüzler ve ağ programlaması gibi pek çok alanda güçlü bir altyapı sağlar.
6. **Topluluk ve Destek**: Java'nın geniş bir geliştirici topluluğu vardır. Bu nedenle, Java ile ilgili kaynak, kütüphane ve çerçevelere kolayca ulaşabilirsiniz.

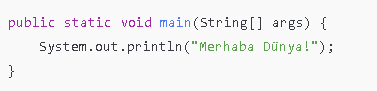
**Java’nın Yapısı**

Java uygulamalarının genel yapısı belirli bileşenlerden oluşur:

1. **Sınıflar ve Nesneler**: Java, sınıfları (classes) ve nesneleri (objects) kullanan nesne yönelimli bir dildir. Her sınıf, özellikler (değişkenler) ve davranışlar (metotlar) içerir.
   * Örnek bir sınıf yapısı:



1. **Paketler**: Java, kodun modülerliğini sağlamak için paketler (packages) kullanır. Paketler, sınıfları organize eder ve çakışmaları önler.
   * Örneğin, java.util paketinde veri yapıları, koleksiyonlar gibi sınıflar bulunur.
2. **Metotlar**: Java’da her işlem bir metodun içinde tanımlanır. main metodu, Java programlarının başlangıç noktasıdır.



1. **Hafıza Yönetimi**: Java, otomatik bellek yönetimi sağlar. *Garbage Collector* sayesinde kullanılmayan nesneler otomatik olarak bellekten kaldırılır.
2. **Java Virtual Machine (JVM)**: Java’nın derlenmiş bytecode’larını çalıştıran sanal makinedir. JVM sayesinde platform bağımsızlık sağlanır.

**Java’nın Kullanım Alanları**

Java, geniş bir kullanım alanına sahiptir:

1. **Masaüstü Uygulamaları**: Swing ve JavaFX gibi GUI kütüphaneleri sayesinde masaüstü uygulamaları geliştirmek mümkündür.
2. **Web Uygulamaları**: Java, servlets ve JSP (JavaServer Pages) gibi teknolojiler ile web tabanlı uygulamaların geliştirilmesinde sıkça kullanılır. Spring ve Hibernate gibi popüler Java çerçeveleri de web geliştirme için güçlü araçlardır.
3. **Mobil Uygulamalar**: Android uygulama geliştirmede en çok tercih edilen dildir. Android’in temel dili Java’dır ve bu sayede Java bilgisi ile Android uygulamaları geliştirilebilir.
4. **Gömülü Sistemler**: Java, çeşitli donanım sistemleri üzerinde gömülü yazılımların geliştirilmesinde de kullanılır.
5. **Büyük Veri ve Bilimsel Uygulamalar**: Hadoop gibi büyük veri (Big Data) platformları Java ile geliştirilmiştir. Ayrıca Java, bilimsel ve araştırma amaçlı uygulamalarda da kullanılır.
6. **Oyun Geliştirme**: Minecraft gibi bazı oyunlar Java ile geliştirilmiştir. Oyun geliştirme için güçlü kütüphaneler ve çerçeveler sunar.

Java’nın platform bağımsızlığı, geniş kütüphane desteği, güvenlik özellikleri ve büyük topluluğu ile birçok alanda tercih edilen bir dil olduğunu görebiliriz. Hem giriş seviyesindeki uygulamalar hem de büyük ölçekli kurumsal çözümler için güçlü bir altyapı sunar.

**Eclipse Nedir ?**

Eclipse Java kodlarını yazmamıza ve düzenlememizi sağlayan bir kod editör programıdır.

**Eclipse Kurulumu**

Kendi sitesinden <https://www.eclipse.org/downloads/packages/> adresine giriyoruz ve açılan ekranda "**Eclipse IDE for Java Developers**" sekmesinde tıklıyoruz. Sağ taraftaki **Download Links** menüsü altından kendi işletim sistemimize uygun programı indiriyoruz ve sonrasında basit bir şekilde kuruyoruz.

**Java JDK Nedir ?**

Java Geliştirme Kiti (JDK), JVM (Java Sanal Makinesi) ve JRE (Java Çalışma Zamanı Ortamı) ile birlikte Java programlamada kullanılan üç temel teknoloji paketinden biridir. Bu üç teknolojiyi birbirinden ayırmak ve aralarındaki bağlantıları bilmek önemlidir.

* JVM, programları yürüten Java platform bileşenidir.
* JRE, Java'nın JVM'yi oluşturan diskteki parçasıdır.
* JDK, geliştiricilerin JVM ve JRE tarafından çalıştırılabilen ve çalıştırılabilen Java programları oluşturmalarına izin verir.

Java'da yeni olan geliştiriciler genellikle Java Geliştirme Kiti'ni ve Java Runtime Environment'ı karıştırır. Fark, JDK'nın Java tabanlı yazılım geliştirmek için bir araç paketi, JRE ise Java kodunu çalıştırmak için bir araç paketi olmasıdır.

JRE, Java programlarını çalıştırmak için bağımsız bir bileşen olarak kullanılabilir, ancak aynı zamanda JDK'nın bir parçasıdır. JDK bir JRE gerektirir çünkü Java programlarını çalıştırmak onları geliştirmenin bir parçasıdır.

**JDK Kurulumu**

Java'yı geliştirmek için, bilgisayarımıza Java Development Kit paketini kurmamız gerekmektedir. Bunun için Google'a Java JDK 8 yazıp ilk çıkan adrese tıklamanız yada bu adresten <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>JDK 8'i indirebilirsiniz.

**Eclipse’in Önemli Terimleri**

Eclipse; *Perspective, View ve Editor’ü* içinde bulundurur. View ve Editor; Perspective’in altında gruplanmıştır.

**Workspace**

Workspace; proje dosyalarının hard diskte tutulduğu konumdur. Projeleriniz,resimleriniz ve diğer yaptıklarınız çalışma ortamında tutulur ve saklanır.

Çalışma ortamınızı Eclipse başlarken veya menüden seçebilirsiniz.

(*File* → *Change Workspace*→*Other*)

**View ve Editor Bölümleri**

*Part’lar* sizin veriyi yönlendirdiğiniz ve değiştirdiğiniz kullanıcı arayüzleridir

*View;* veri üstünde çalışmak için kullanılır. Bu veriler hiyerarşik durumda bulunabilir.Eğer veri view üzerinden değiştirildiyse, bu değişiklik direkt olarak ilgili veri yapılarına uygulanır. View’a bir örnek Java Packet Explorer verilebilir. Java Packet Explorer, Eclipse projelerinizin dosyalarını incelemenize olanak sağlar. Eğer Java Packet Explorer içindeki bir veriyi değiştirirseniz; örnek olarak ismini değiştirmek; o dosyanın ismi direkt olarak dosya sisteminde de değişir.

Java’da nesneler ve sınıflar, nesne yönelimli programlamanın (OOP) temel yapı taşlarıdır. Sınıflar, nesnelerin özelliklerini ve davranışlarını tanımlarken, nesneler sınıflardan üretilmiş örneklerdir. Bu konuları anlamak, Java ile daha modüler, yeniden kullanılabilir ve anlaşılır kodlar yazmamıza olanak tanır.

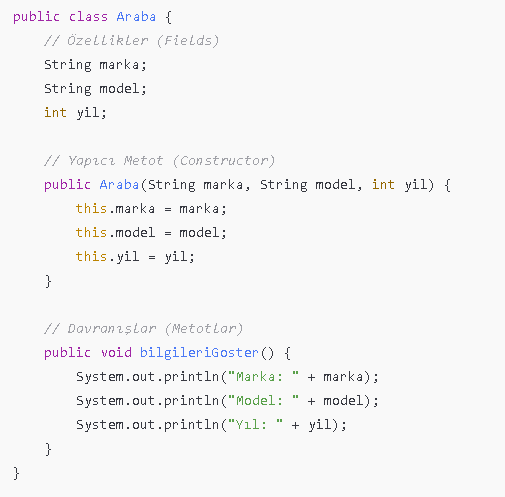
**Nesneler ve Sınıflar**

Sınıf (Class): Bir sınıf, kendi türünde desteklenen bir nesnenin, niteliklerini/davranışlarını betimleyen bir şablon olarak tanımlanabilir.

Nesne (Object): Nesneler durumlar ve davranışlara sahiptir. Örnek: Bir köpek; renk, isim, doğurganlık gibi niteliklere sahip olmakla birlikte havlama, sallama ve yemek yeme gibi davranışlara da sahiptir. Nesne, bir sınıfın oluşumudur(instance).

Bir sınıf tanımlamak için class anahtar kelimesi kullanılır. Sınıfın adı büyük harfle başlamalıdır ve sınıfın özellikleri ile metotları süslü parantez {} içinde tanımlanır.

Örnek bir sınıf:



**Nesne Oluşturma ve Kullanma**

Bir sınıftan nesne oluşturmak için new anahtar kelimesi kullanılır. Oluşturulan nesne, sınıfın tanımladığı özellik ve davranışları kullanabilir.

Örnek nesne oluşturma ve metot çağırma:





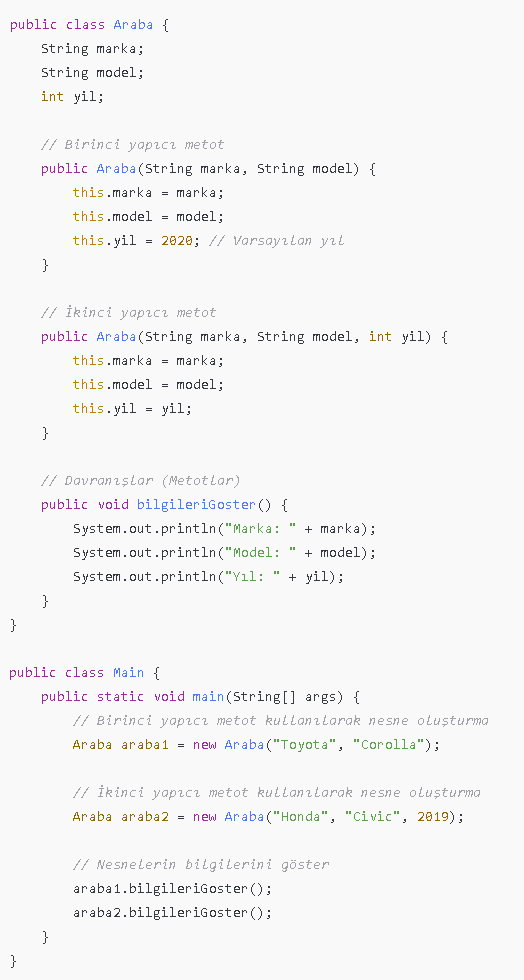
**Yapıcı Metotlar (Constructors)**

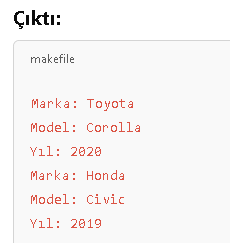
Sınıfların en önemli alt konularından biri de Constructor’lardır. Her sınıfın bir constructor’ı vardır. Eğer, bir sınıf için açıkça bir constructor yazmazsak, java derleyicisi bu sınıf için varsayılan bir constructor oluşturur.Yeni bir nesne oluşturulan her zaman, en azından bir constructor çalıştırılacaktır. Constructor’ların ana kuralı, sınıfı ile aynı ismi taşımasıdır. Bir sınıfı birden çok constructor’ı içerebilir.



**Birden Fazla Yapıcı Metot (Constructor Overloading)**

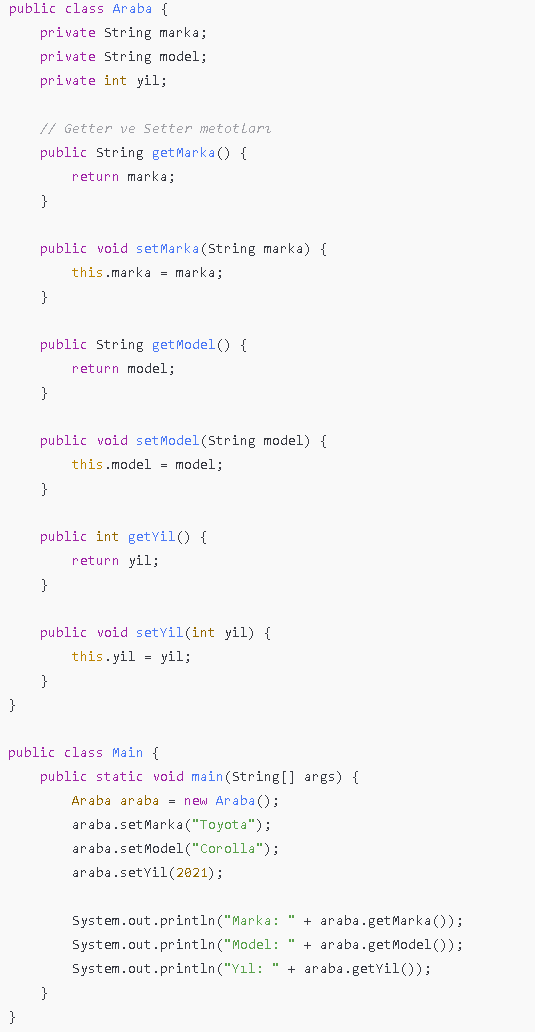
Java’da bir sınıf birden fazla yapıcı metot tanımlayabilir. Bu sayede, nesne oluşturulurken farklı parametrelerle işlem yapılabilir.

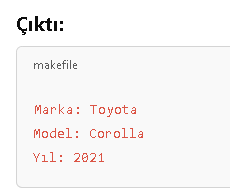




**Nesneye Yönelik Özellikler (Encapsulation)**

Java’da nesneye yönelik programlamanın temel ilkelerinden biri olan *enkapsülasyon*, sınıf özelliklerini gizlemek ve güvenli bir şekilde erişimi sağlamaktır. Genellikle sınıf özellikleri private olarak tanımlanır ve bu özelliklere dışarıdan erişim getter ve setter metotları ile sağlanır.





Bu kodda, Araba sınıfının özellikleri (marka, model, yil) private olarak tanımlanmıştır, böylece doğrudan erişime kapalıdır. Ancak, get ve set metotları sayesinde bu özelliklere kontrollü bir şekilde erişim sağlanır. Bu yöntem, **enkapsülasyon** ilkesini uygulayarak daha güvenli ve sürdürülebilir bir yapı sunar.