

Java'ya Giriş

# Ders 01

Alper Sarıtaş

Ders Metni

## İçindekiler

<b>1.1</b>	<b>Java'yı Kavrama .....</b>	<b>3</b>
1.1.1	Sanal Java Makinesi.....	3
<b>1.2</b>	<b>Java ile Dizgilemeye Giriş .....</b>	<b>5</b>
1.2.1	Bütünleşmiş Geliştirme Ortamı.....	5
1.2.2	Temel Veri Türleri .....	6
<b>1.3</b>	<b>Java İçin Çalışma Ortamını Oluşturma.....</b>	<b>7</b>
1.3.1	Java Geliştirme Takımı .....	8
1.3.2	Visual Studio Code .....	8
1.3.3	Diğer Kullanışlı Belgeler.....	8



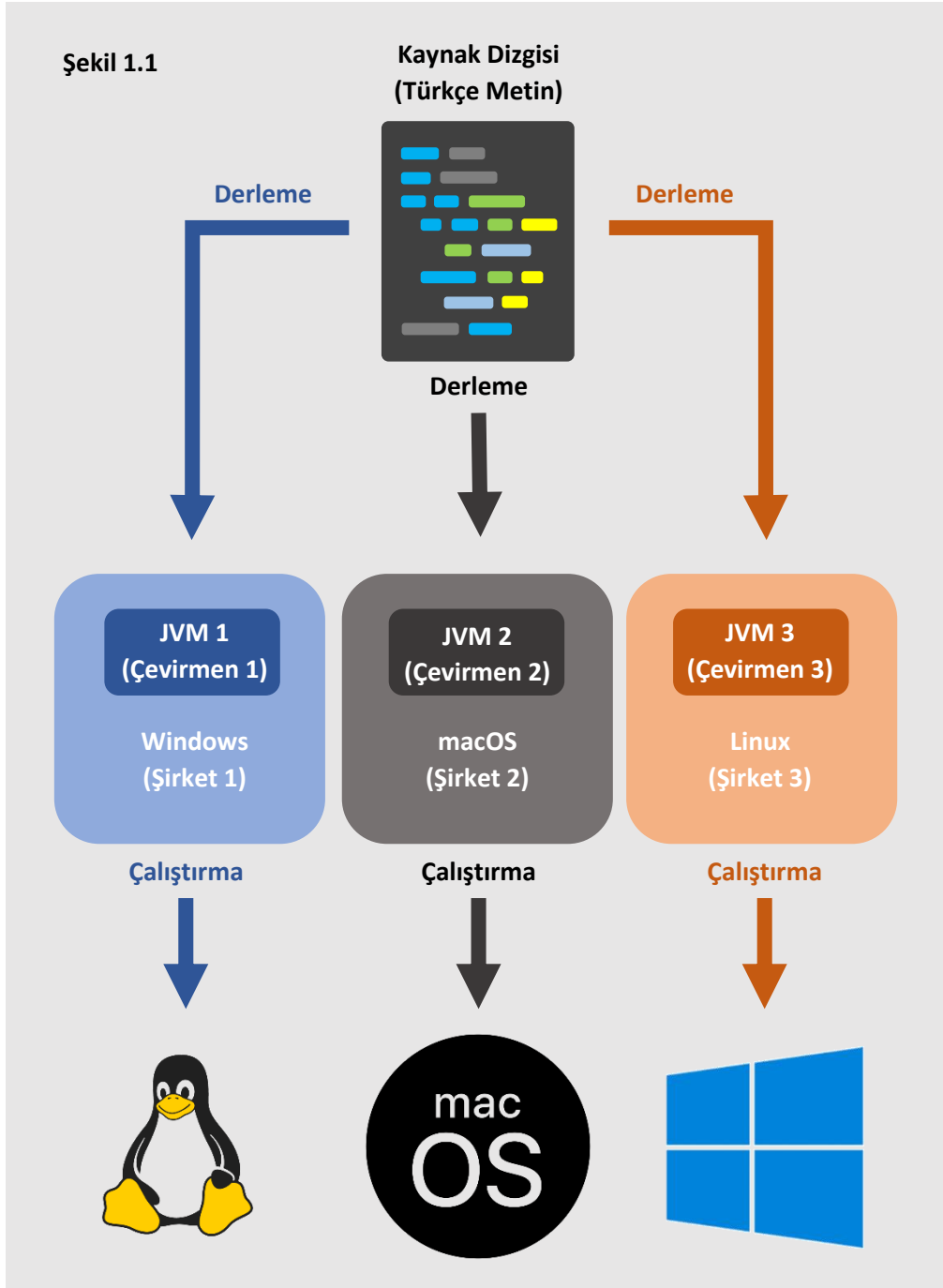
## Java'ya Giriş

### 1.1 Java'yı Kavrama

Java ilk olarak internet için tasarlandı. Daha sonra dizgileri (code) kullanıcılara sunuldu, yani açık kaynak haline geldi. Bu sayede en çok kullanılan ve bilinen dizgileme dili haline geldi. Neden mi? Çünkü dili geliştirenlerin amacı, performansta düşüş yaşamayan, güvenli ve basit bir nesne tabanlı dizgileme dili yapmak, bunun yanı sıra işletim dizgelerinden (sistem) veya platformlardan bağımsız olarak çalışabilmesini sağlamak. Bu ne demek? Nesne tabanlı kavramı önemli çünkü bu özellik, Java'yı C dilinden ayıran bir özellik ve aynı zamanda nesne tabanlı dizgileme, bir yazılım kavramı. Bu da onu işletim dizgelerinden bağımsız, yani her yerde çalışabilir yapıyor. Bir kere yazıyorsunuz ve her işletim dizgesinin kendi yazılımı tarafından derleniyor ve çalıştırıyor. Bu konuyu ileride işleyeceğiz.

#### 1.1.1 Sanal Java Makinesi

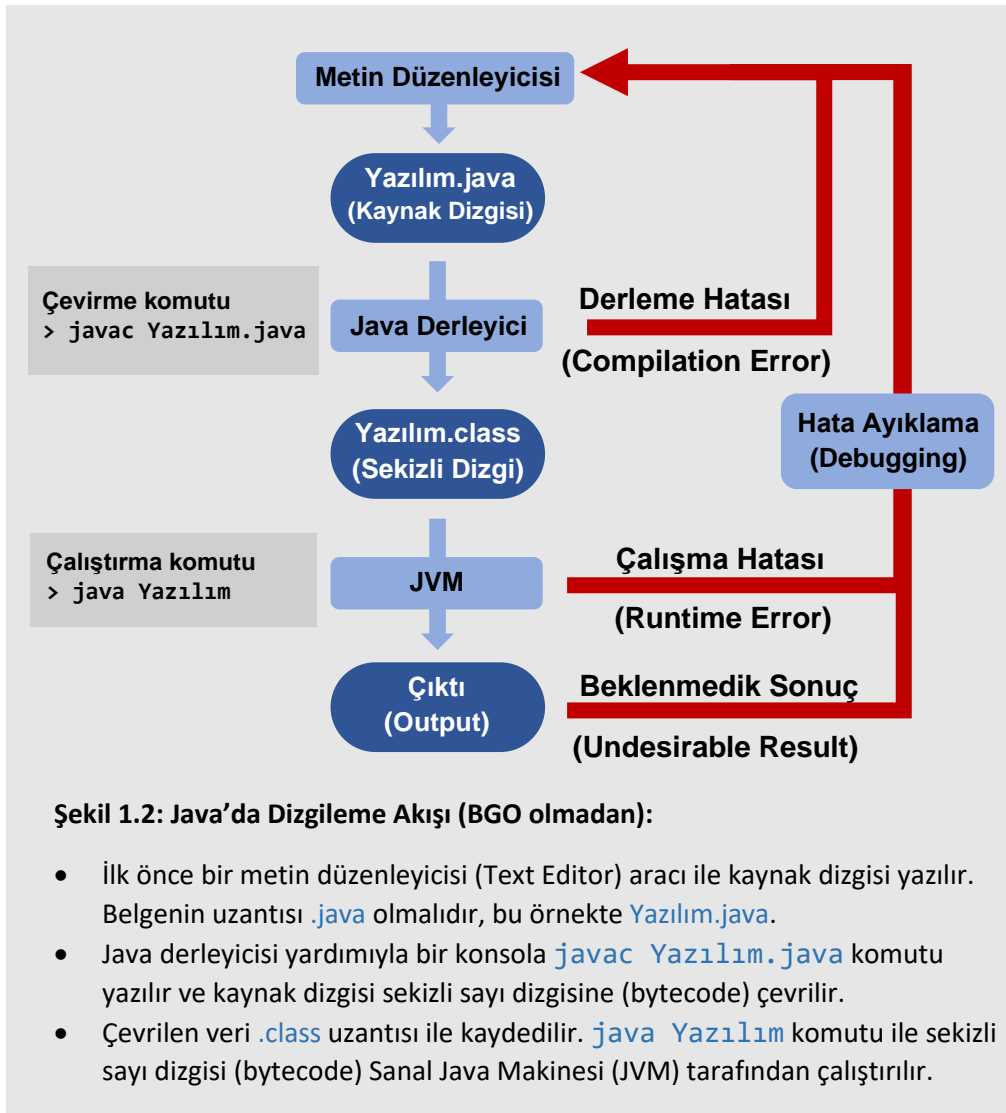
Java'yı diğer dillerden ayıran özellik Sanal Java Makinesidir (JVM ya da Java Virtual Machine). JVM teknik bir kavram o yüzden detaylarına daha sonra değineceğiz ama yüzeysel bir tanımlama gerekli ki ileride yapacağımız veya vereceğimiz bir çok kararın neden olduğunu daha kolay anlayalım. Sanal Java Makinesini çevirmenlere benzetebiliriz. Üç tane çevirmen düşünelim, sırasıyla İngilizce, Almanca ve Rusça dillerini bilsinler ve Türkçeden çeviri yapabilsinler. Bu üç çevirmen farklı şirketlerde çalışıyorlar ve sadece çalıştıkları ortama alışkınlar bu yüzden sadece kendi şirketlerinde çalışabiliyorlar. Artık bu benzetmeleri teknik kavramlarına çevirebiliriz. Türkçe dili Java ile yazdığımız kaynak dizgisi, çevirmenler ise her biri farklı işletim dizgesinde çalışan JVM'ler, bu şirketler işletim dizgeleri. Örnek olması açısından Windows, MacOS ve Linux diyebiliriz. Her JVM sadece kendi işletim dizgesinde çalışabiliyor ve Türkçe metni yani Java ile yazdığımız kaynak dizgisini kendi işletim dizgesinin anlayabileceği bir dile çeviriyor ve bilgisayar yazılım (program) çalıştırıyor. Tabiki bu kadar basit değil bilgisayarla anlaşabilmek bu işlemler daha kapsamlı ve detaylı. Biz sadece basitleştirdik. (Şekil 1.1)



JVM bize bazı kolaylıklar sağlıyor: Sınıf yükleyici (class loader), bellek yönetimi (memory management), çöp toplama (garbage collector) ve çalıştırma motoru (execution engine) hizmetlerini sunuyor. Bunların ne olduğunu şimdilik bilmemize gerek yok. İleride dizgilemeye başladığımız zaman nerede işe yaradıklarını yeri geldiğinde anlatacağım.

## 1.2 Java ile Dizgilemeye Giriş

C dilinde ve diğer bir çok dizgileme dilinde olduğu gibi Java’da da önce kaynak dizgisi (source-code) yazılır, ardından bu dizgi derlenir (compile) ve eğer hatasız (error-free) ise çalıştırılır (execute). Her adımda hata olabilir ve olacaktır. Yazılımın çıktısı (output) da buna dahil. Her adımda denetleme döngüleri vardır ki dizginizin hatasız olduğu kesinleşsin ve JVM tarafından çalıştırılabilirsin. (Şekil 1.2’e bakın)



### 1.2.1 Bütünleşmiş Geliştirme Ortamı

Denetleme döngülerinin tekrarını en aza indirmek ve basitleştirmek için Java neredeyse her zaman bütünleşmiş geliştirme ortamında (Integrated Development

Environment - IDE) dizgilenmiştir. IDE'ler genellikle metin düzenleyicisi (Text Editor), derleyici (Compiler), yorumlayıcı (Interpreter), bağlayıcıyı (Linker), hata ayıklayıcıyı (Debugger) ve daha bir çok özelliği bir araya getirir. Bu modülde sade (low-level) bir IDE olan *Visual Studio Code* kullanılacaktır. Bunun nedeni çoğu IDE'lerin (IntelliJ IDEA, Android Studio, Eclipse...) bizim için her şeyi otomatik olarak hazırlamasıdır. Bizim amacımız her şeyi başından anlamaktır. [Şekil 1.1](#) tekrar baktığımızda derleme (compile) hataları genellikle bir IDE tarafından işaretlenir, böylece yazarken doğrudan düzeltilebilir. Normalde Java'yı derlemek ve çalıştırmak için bir terminal kurmak gereklidir ama her ikisi de IDE'ye entegre edilmiştir. Çalışma anı hatalarının (runtime error) veya yanlış bir sonucun nedenini bulmak için dizgiyi adım adım hata ayıklayıcısı (debugger) ile gözden geçirebiliriz. Bununla ilgili ayrıca bir eğitim olacaktır.


### 1.2.2 Temel Veri Türleri

Veri türleri, ne tür değişkenler olduğunu, hangi işlemler yapılabildiğini, hafızada nasıl depolandığını ve değişkenlerinin temsilinin nasıl görüldüğünü, yani hangi bitin (bit) hangi anlama geldiğini ve kaç tane olduğunu belirler. Öte yandan temel veri türleri (Primitive Data Types), veri türlerinin daha basit hali değildir, veri türlerinin yapı taşlarıdır. Aşağıdaki tabloda Java'daki temel veri türlerini görebilirsiniz:

Veri Türleri	İçeriği	Karşıladığı Değer
boolean	1 bit doğruluk değeri	true, false
byte	8 bit atanmış tam sayı	$\{-2^7, 2^7 - 1\}$ , $2^7 = 128$
char	16 bit Evrensel dizgi	Evrensel dizgi (Unicode) Karakterleri
short	16 bit atanmış tam sayı	$\{-2^{15}, 2^{15} - 1\}$ , $2^{15} = 32768$
int	32 bit atanmış tam sayı	$\{-2^{31}, 2^{31} - 1\}$ , $2^{31} = 2147483647$
long	64 bit atanmış tam sayı	$\{-2^{63}, 2^{63} - 1\}$ , $2^{63} = 9223372036854775808$
float	32 bit kesirli sayı	$\pm 2^{127} \approx 10^{38}$ , 7 ondalık basamak
double	64 bit kesirli sayı	$\pm 2^{1023} \approx 10^{308}$ , 15 ondalık basamak

Eğer bu veri türleri atanmamış ise yani bir değer verilmemiş ise otomatik olarak boolean için false ya da tam ve ondalıklı sayılar için 0 ya da char için null değerini alır. En sık kullanılan veri türleri booleani int, double ve String (ileride öğreneceğiz) sınıfıdır ama bu veri türlerinin kullanımının artması bir yazılım mühendisinin amacından sapmasına neden olabilir. Mühendisin amacı kaynak tüketimini ve tamamlanma süresini en az hata payıyla mümkün olduğunca azaltmaktır. Yani basit olarak düşündüğümüzde dizginizdeki verilerin boyutları ne kadar küçük oldursa ve


aynı şekilde komut sayınız ne kadar az olursa ürününüz o kadar verimli olur. 2014 yılında Youtube'un verdiği bu karar bir örnektir:

 YouTube ▶ Öffentlich 01.12.2014

We never thought a video would be watched in numbers greater than a 32-bit integer (=2,147,483,647 views), but that was before we met PSY. "Gangnam Style" has been viewed so many times we had to upgrade to a 64-bit integer (9,223,372,036,854,775,808)!

Hover over the counter in PSY's video to see a little math magic and stay tuned for bigger and bigger numbers on YouTube.

[Übersetzen](#)



PSY - GANGNAM STYLE (강남스타일) M/V

Bu, dizgi verimliliği kavramı ile açıklanır ve dizginizi verimli hale getirmek sizi iyi bir mühendis yapar. Bu eğitim amacı ise sizin iyi bir mühendis olabilmeniz için gereken altyapıyı size kazandırmaktır.

### 1.3 Java İçin Çalışma Ortamını Oluşturma

Bu bölümde Java ile dizgilemeye başlamak için gerekli ortamı adım adım oluşturacağız. Öncelikle Java Geliştirme Takımını (Java Development Kit - JDK), ardından *Visual Studio Code* IDE'sini indireceğiz. Ardından çok yararlı belgelerin (documentation) bağlantılarını (link) gözden geçireceğiz.

### 1.3.1 Java Geliştirme Takımı

Masaüstümüzde yeni bir metin dosyası oluşturup Java komutlarını yazmaya başlayabiliriz ama Java bir betik dili (Scripting Language) yani insanların anlayabileceği türden komutlar kullanılan dillerden biri olduğu için bu yazdığımız kaynak dizgisini bilgisayarın anlayabileceği bir biçime dönüştürmemiz gerekmektedir. Eğer yeterli zamanınız varsa makine dilini (Machine Language) öğrenip yazdığınız Java komutlarını sıfır-bir tertibine (kombinasyon) dönüştürebilirsiniz. İşleminiz bittiğinde ürününüz buna benzeyecek:

```
0011 0000 0000 0000 0010 001 0 0000 1111 0101 100 100
1 00000 0101 101 101 1 00000 0110 010 001 000000 0000
010 0 0000 1100 0101 011 010 1 00001 0000 010 0 0000
1001 0001 101 101 1 00001 0000 111 0 0000 1010 0001
100 100 1 00001 0001 001 001 1 00001 0000 111 0 0000
0011 1011 100 0 0001 0000 1011 101 0 0001 0001 1111
0000 0010 0101 0100 0000 0000 0000 0011 0010 0000
```

Çok basit bir işlem yapan bu ikili sayı sıralaması bile uzman olmayan bir insanın hızlıca yapabileceği türden bir iş değildir. İşte bu yüzden Java'yı geliştirenler yazılım mühendisleri zamanlarını buna harcamasınlar diye Java Geliştirme Takımını (JDK) geliştirmişler. JDK, kaynak dizgisini Java Çalışma Anı Ortamının (JRE) çalıştırabileceği biçime dönüştürüyor. JDK içinde JRE, yorumlayıcı (Interpreter), derleyici (Compiler), belgelyici (Archiver), belge üretici (Documentation Generator) ve bazı diğer geliştirme araçlarını barındırıyor. O yüzden bu araç takımlarının en son çıkan sürümlerini bilgisayarımıza indirip yüklememiz gerekiyor.

Bağlantı: [Java Geliştirme Takımı \(JDK\)](#)

### 1.3.2 Visual Studio Code

Bölüm [1.2.1](#)'de de bahsettiğimiz gibi kullanacağımız metin düzenleyicisi (Text Editor) VS Code olacaktır. Neyse ki bu konuda kişinin kendi tercihi söz konusudur. Bu yüzden en sevdiğiniz metin düzenleyicisini kullanmakta özgürsünüz. Eğitimde örnekler VS Code üzerinden verilecektir o yüzden kullanılan uygulamaya bağlı sorunları çözmek sizin sorumluluğunuzdadır.

Bağlantı: [Visual Studio Code \(IDE\)](#)

### 1.3.3 Diğer Kullanışlı Belgeler

- [JDK İndirme Rehberi](#)
- [VS Code İndirme Rehberi](#)
- [Java SE Belgeleri](#)