Java Nedir?

selsoft

build better, deliver faster

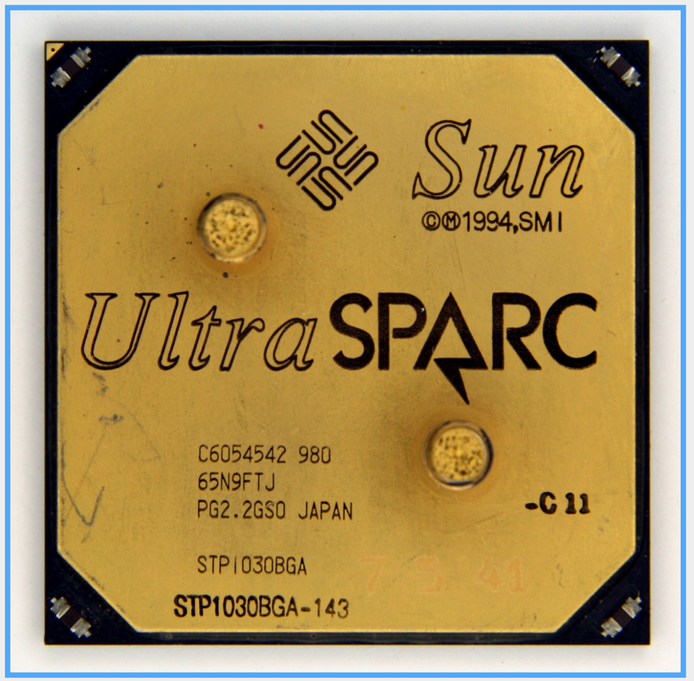
selsoft

build better, deliver faster

Java'nın Kısa Tarihi

Sun Microsystems

- Sun Microsystems, 1982 yılında iki mühendis ve iki işletmecinin ortak girişimi olarak kuruldu.
- Sun, Solaris işletim sistemi ve Sparc CPU ve makinaları (workstation) ile kurumsal Unix platformlarında başarılı bir markaydı.



Green Team

- 1991'de Sun, Patrick Naughton, Mike Sheridan ve James Gosling önderliğindeki küçük bir takımla bir AR-GE projesi başlattı.
- "**Green Team**" (**Yeşil Takım**) adlı bu takımın amacı ICT (Information & Communication Technology) dünyasına yönelik AR-GE yapmaktı.

*7 - Star-Seven

- 18 aylık bir çalışmadan sonra 1992 yazında “***7**”, (**Star Seven**) isminde, dokunmalı ekrana sahip bir kontrol cihazı geliştirdirler.
- “*7”, TV, video oynatıcısı ve müzik seti gibi pek çok ev cihazını kontrol edebiliyordu.
  - https://www.youtube.com/watch?v=1CsTH9S79qI

Oak Programlama Dili

- Gosling, projede önce C++'ı kullanıp, ona bazı eklentiler yapmasına rağmen sonrasında yepyeni bir dil geliştirmenin daha iyi olacağını düşündü.
- **Oak** ismini verdiği bu yeni dil, C/C++'a çok benzer söz dizimine sahip, nesne-merkezli ve platformdan bağımsızdı.
  - Basit olması için, C/C++'ın pek çok özelliğini dışarıda bırakacak şekilde tasarlanmıştı.

Yön Değişikliği

- Green Team’in projesi ticari olarak başarılı olamadı.
- Ama takımın ileri gelenleri o sırada, geliştirdikleri altyapının ve Oak dilinin Internet’e çok uygun olduğunu farkettiler.
- Ve hemen yönlerini değiştirip, ismi daha sonra HotJava olarak değiştirilen Java-tabanlı bir tarayıcı geliştirdiler: WebRunner

Gosling’in Hikayesi

Gosling explains: "*We had already been developing the kind of `underwear' to make content available at the same time the Web was being developed. Even though the Web had been around for 20 years or so, with FTP and telnet, it was difficult to use. Then Mosaic came out in 1993 as an easy-to-use front end to the Web, and that revolutionized people's perceptions. The Internet was being transformed into exactly the network that we had been trying to convince the cable companies they ought to be building. All the stuff we had wanted to do, in generalities, fit perfectly with the way applications were written, delivered, and used on the Internet. It was just an incredible accident. And it was patently obvious that the Internet and Java were a match made in heaven. So that's what we did.*"

Oak'tan Java'ya

- Java, Sun tarafından **23 Mayıs 1995**'te genel amaçlı bir programlama dili olarak piyasaya sunuldu.
  - Pek çokları tarafından "**Web'in dili**" olarak algılandı,
  - Netscape'in web yenilikleriyle birlikte gelişti.
- Oracle'ın 2010 yılında Sun'ı satın almasıyla Java, Oracle'ın sahipliğinde hayatına devam etmektedir.

selsoft

build better, deliver faster

Java'nın Temel Özellikleri

Java ve C/C++

- Java, söz dizimi (syntax) açısından C++'a çok benzeyen ve temelde Smalltalk ve C++ kültürünü benimseyen bir dildir.
- Java, 90lı yıllarda en yaygın sistem geliştirme dili olan C/C++'ın yerini alacak şekilde geliştirildi.
- C++'dan daha küçük, daha soyut ve daha yetenekli bir dil olacak tasarlandı.

Şu Anda Java

- 1995 yılında büyük beklentiler ile piyasaya çıkan Java, o günden bugüne bu beklentileri karşıladığı söylenebilir.
- Java, muhtemelen, şu anda dünyada en geniş geliştirici topluluğu tarafından kullanılan, en büyük kapalı ya da açık kaynak kodlu kütüphanelere sahip, en çok kullanılan yazılım geliştirme dilidir.

Java Nedir?

Sun, 1995'te Java'yı sunarken yayınladığı ve "Java'nın babası" James Gosling'in kaleme aldığı **The Java Language - A White Paper**'da Java'yı şöyle tanıtıyordu:

Java: A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language.

Java: Basit, nesne-merkezli, ağlarda yetenekli, yorumlanan, sağlam, güvenli, mimari olarak tarafsız, taşınabilir, yüksek performanslı/başarımlı, çok kanallı, dinamik bir dil.

Java Basittir

- Java, ataları olan C ve C++'tan daha basittir.
- Çünkü onların pek çok karmaşık özelliğini budamıştır:
  - Pointer aritmetiği,
  - Programcıya bağımlı bellek yönetimi (memory management),
  - İşlemci çoklu kullanımı (operator overloading),
  - Platforma bağımlı yapılar ve diğerleri.

Küçük, Soyut ve Güçlü

- Budanan ve yeni gelen özellikler ile Java, C/C++'tan daha küçük ama daha güçlü bir dil oldu.
- Soyut, nesne-merkezli ve güvenilir yapısı, Java'yı özellikle genel amaçlı kullanım ve iş uygulamaları için rahat kullanılabilir bir seçenek yaptı.

Diğer Özellikleri

- **Nesne-merkezli** (object-oriented), **ağlarda yetenekli** (network savvy), **sağlam** (robust), **güvenli** (secure) ve **çok kanallı** (multi-threaded) gibi özellikleri, 90'lı yılların şartlarında ve C/C++'a göre yorumlamak daha anlamlıdır.
- Bugünlerde bir dilin bu özelliklere sahip olması sıradan bir durum olarak görülür.

Dinamik Dil

- Java, **dinamik tipli** (**dynamically-typed**) değildir, aksine **statik tipli** (**statically-typed**) bir dildir.
- Java'nın dinamik olmasıyla kastedilen, tipleri çalışma zamanında (run-time) yükleyebilmesidir.
  - Dolayısıyla Java'da bir programın kullanacağı tipler, uygulamanın derlenmesi ya da ayağa kalkması sırasında hazır olması gerekmez.
  - Bu durum, webin tabiatına daha uygundur.

Mimari Olarak Tarafsız

- **Mimari olarak tarafsız** (architectural-neutral) olması, Java'nın herhangi bir platforma (donanım ve işletim sistemi) bağımlı olmaması demektir.
- **Java, standartlar üzerine kurulmuştur**,
  - Örneğin, `int` her platformda 32-bittir.
- Ayrıca, Java ve JVM, platformlarla alakalı sadece en genel ön kabullere sahiptir.

Taşınabilir

- Java'nın platformlardan bağımsız olması ve yorumlanan (interpreted) tabiatı, onu aynı zamanda **taşınabilir** (portable) kılmaktadır.
- Derlenmiş (compiled) Java kodu, `.class` dosyasında bytecode olarak saklanır.
  - Bytecode, platformdan bağımsızdır, JVM'in olduğu her ortamda çalıştırılabilir.

WORA ve WOTA

- Başından bu yana **Bir Kere Yaz Her Yerde Çalıştır** (**Write Once, Run Anywhere, WORA**) Java'nın taşınabilirlik konusundaki hedefidir.
  - Tabi olarak bu prensibin geçerli olması için Java programcısı da platforma özel kod yazmamalıdır.
- Ama pratikte geçerli olan, **Bir Kere Yaz Her Yerde Test Et ☺** (**Write Once, Test Anywhere, WOTA**) prensibidir.

Yüksek Başarımlı

- Yüksek başarımlı (high-performance) ile aslen, tüm bu özelliklerine rağmen performans açısından iyi bir seviyede olduğu/olabileceği vurgulanmıştır.
  - Aslen Java'nın performansı ilk yıllarında iyi değildi.
  - Fakat özellikle 2. sürümünden itibaren HotSpot JVM ile performansı iyileşti.
- Her halükarda Java geliştirenlerin performans konusunda bilgili ve dikkatli olmaları gereklidir.

Java Nesne-Merkezlidir - I

- **Java nesne-merkezli bir dildir.**
- Nesne-merkezli diller, prosedürel/yordamsal dillerden çok farklıdırlar.
- Yordamsal dillerde en önemli kavram/soyutlama yordam/prosedür/fonksiyondur ve bu yapılar, bir problemi alt problemlere bölüp, her birini adım adım (step-wise decomposition) tanımlamakta kullanılır.
- Fakat nesne-merkezli diller tamamen nesne kavramı üzerine otururlar.

Java Nesne-Merkezlidir - II

- Nesne, insan zihninin kendisine yöneldiği, özellik ve davranışlara sahip, fiziksel olan ya da olmayan herhangi bir olgudur.

- Yazılımda her tip nesne için, verisi ile davranışını bir paket haline getirip sarmalayan ya da kapsülleyen (encapsulation) ve adına genelde **sınıf** (**class**) denen kalıplar (şablonlar) oluşturulur.

Sınıf = Veri + Davranış

- Nesneler, sınıflardan üretilmiş çalışma zamanı yapılarıdır.

Derleme ve Yorumlama - I

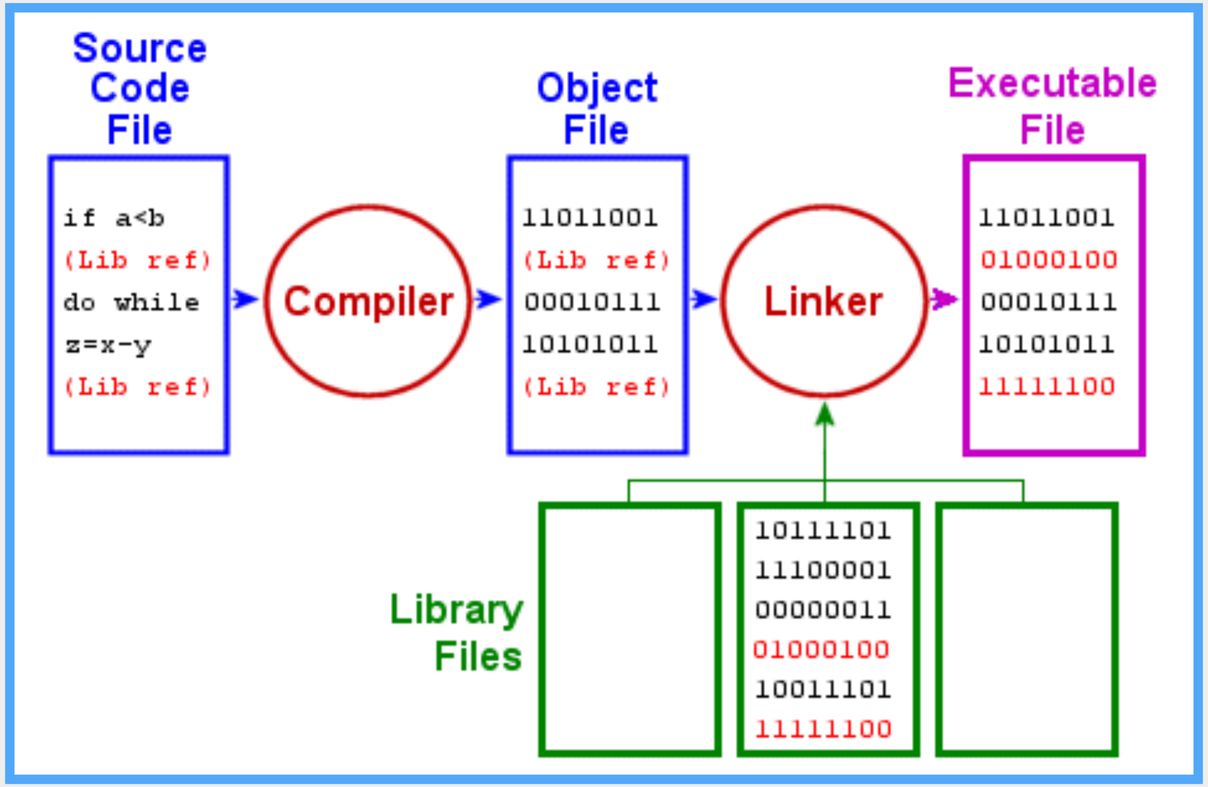
- **Derleme** (**compilation**) ve **yorumlama** (**interpretation**), programlama dilleri dünyasının en temel kavramlarındandır.
- Diller de sıklıkla derlenen (compiled) ve yorumlanan (interpreted) olarak ikiye ayrılır.
  - Compiled: Fortran, C, C++, Pascal, Haskell, Rust, Go
  - Interpreted: JavaScript, PHP, Python, Ruby

Derleme (Compilation) - I

- Genel olarak derleme (compilation), kaynak kodun **derleyici** (**compiler**) tarafından bir formdan başka bir forma çevrilmesidir, çeviri (translation) de denir.
  - C kodunun okunarak karşı gelen Java kodunun oluşturulması bir derleme ya da çevirmedir.
- Özelde ise kaynak kodun, **çalıştırılabilir** (**executable**) makina koduna çevrilmesidir.
  - Örneğin C++ kodundan çalıştırılabilir (exe, out) kod üretmektir.

Derleme (Compilation) - II

Kaynak kod, önce derleyici tarafından derlenir, sonra kullandığı kütüphaneler ile ilişkilendirilip, sonuç olarak çalıştırılabilir (executable) makina kodu üretilir.



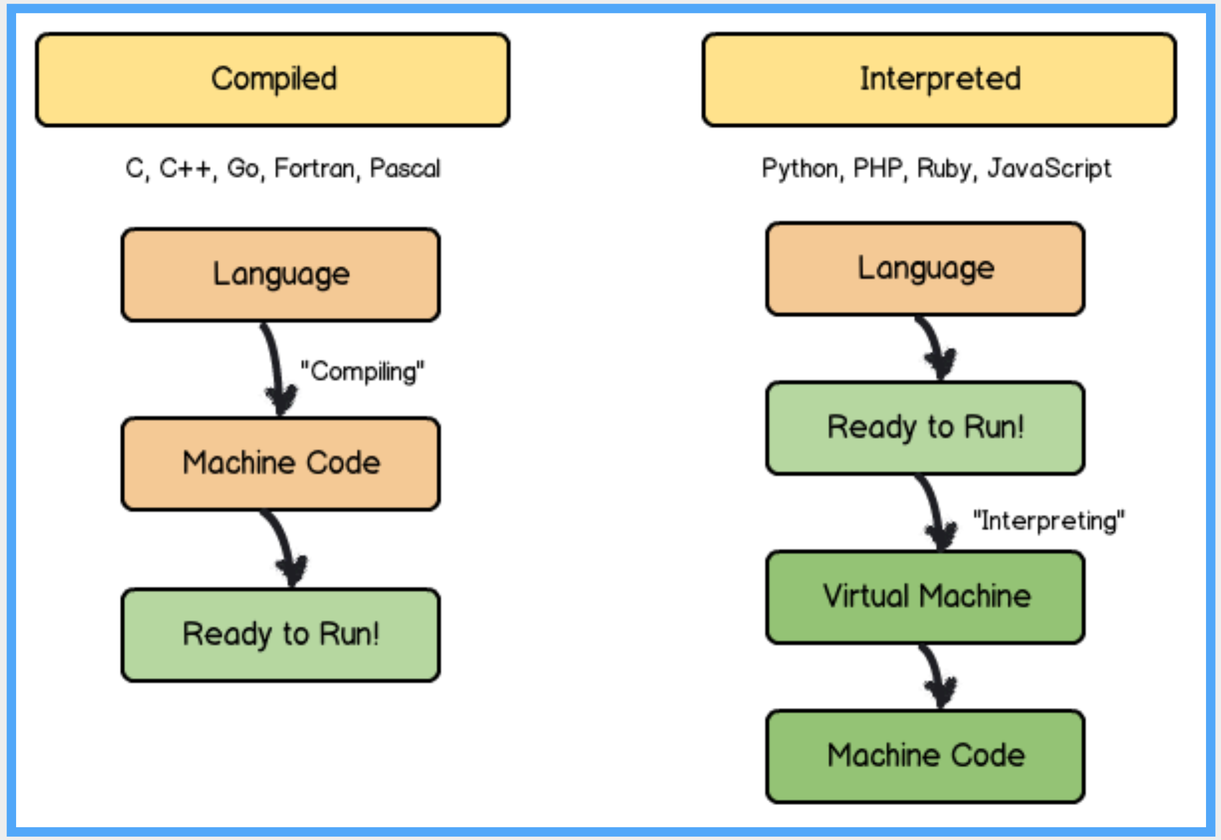
C++ Kodunu Derleme ve Çalıştırma

- `MonteCarloPI.cpp` dosyasındaki C++ kodunun GNU derleyicisi ile derlenip çalıştırılması.
  - MacOS üzerinde `g++ -O3 -std=c++17 -o MonteCarloPI.out MonteCarloPI.cpp` komutu ile çalıştırılabilir `MonteCarloPI.out` kodu üretilir ve çalıştırılır.

```cpp
#include <iostream>
#include <stdlib.h> // srand
#include <ctime>

using namespace std;

# define PI_L  3.14159265358979323846264338327950288 4L

int main() {
    cout.precision(10);
    int n;
    int dotsInCircle = 0;
    cout << "Please enter number of points: " << endl;
    cin >> n;
    clock_t start = clock();
    for (int i = 0; i < n; i ++) {
        double x = static_cast<double>(rand()) /
            static_cast<double>(RAND_MAX);
        double y = static_cast<double>(rand()) /
            static_cast<double>(RAND_MAX);
        double distance = x * x + y * y;
        if (distance <= 1.0)
            dotsInCircle ++;
    }
    clock_t end = clock();
    float time =  ((float) end - start) / CLOCKS_PER_SEC;
    double myPi = (double) 4 * dotsInCircle / n;

    cout << "PI:            " << PI_L << endl;
    cout << "Calculated PI: " << myPi << endl;
    cout << "Time is: " << time << " s." <<endl;

    return 0;
}
```

Yorumlama (Interpretation)

- Yorumlama ile kaynak kod, başka bir forma çevrilmeden, **yorumlayıcı** (**interpreter**) tarafından, doğrudan makina kodundaki bloklar çağrılarak çalıştırılır yani yorumlanır (interpretation).
- Derlemede olduğu gibi bir çıktı kodu içeren bir dosya yoktur.



Python Kodunu Çalıştırma

- `MonteCarloPI.py` dosyasındaki Python kodunun, Python yorumlayıcısı ile çalıştırılması.
  - Derlenmiş C++ kodu ile yorumlanan Python kodunun performansını aynı sayıda nokta için, çalışma süresi açısından karşılaştırın.

```
import random as r
import math as m
import time

# Total number of point to throw.
s = input("Please enter number of points: ")
n = int(s)

dotsInCircle = 0

start = time.clock()
# Iterate for the number of darts.
for i in range(0, n):
  # Generate random x, y in [0, 1].
  x = r.random()
  y = r.random()
  # Increment if inside unit circle.
  if (x*x + y*y) < 1.0:
      dotsInCircle += 1

end = time.clock()

# inside / total = pi / 4
pi = (float(dotsInCircle) / n) * 4

print "PI:             " , m.pi
print "Calculated PI: " , pi
seconds_elapsed = (end - start)
print "Time is ", seconds_elapsed, " s."
```

Derleme ve Yorumlama - II

- Derlenen kodlar, çok daha etkindirler çünkü hem hatadan temizlenmişlerdir hem de çalışma-zamanına daha az iş bırakmışlardır.
  - Ama makina koduna derlenmiş kod platforma bağımlıdır dolayısıyla aynı kaynak kod her farklı platform için ayrı ayrı derlenmelidir.
  - Ayrıca kaynak koduna yapılacak herhangi bir değişiklik tekrar derlemeyi gerekli kılar.

Derleme ve Yorumlama - III

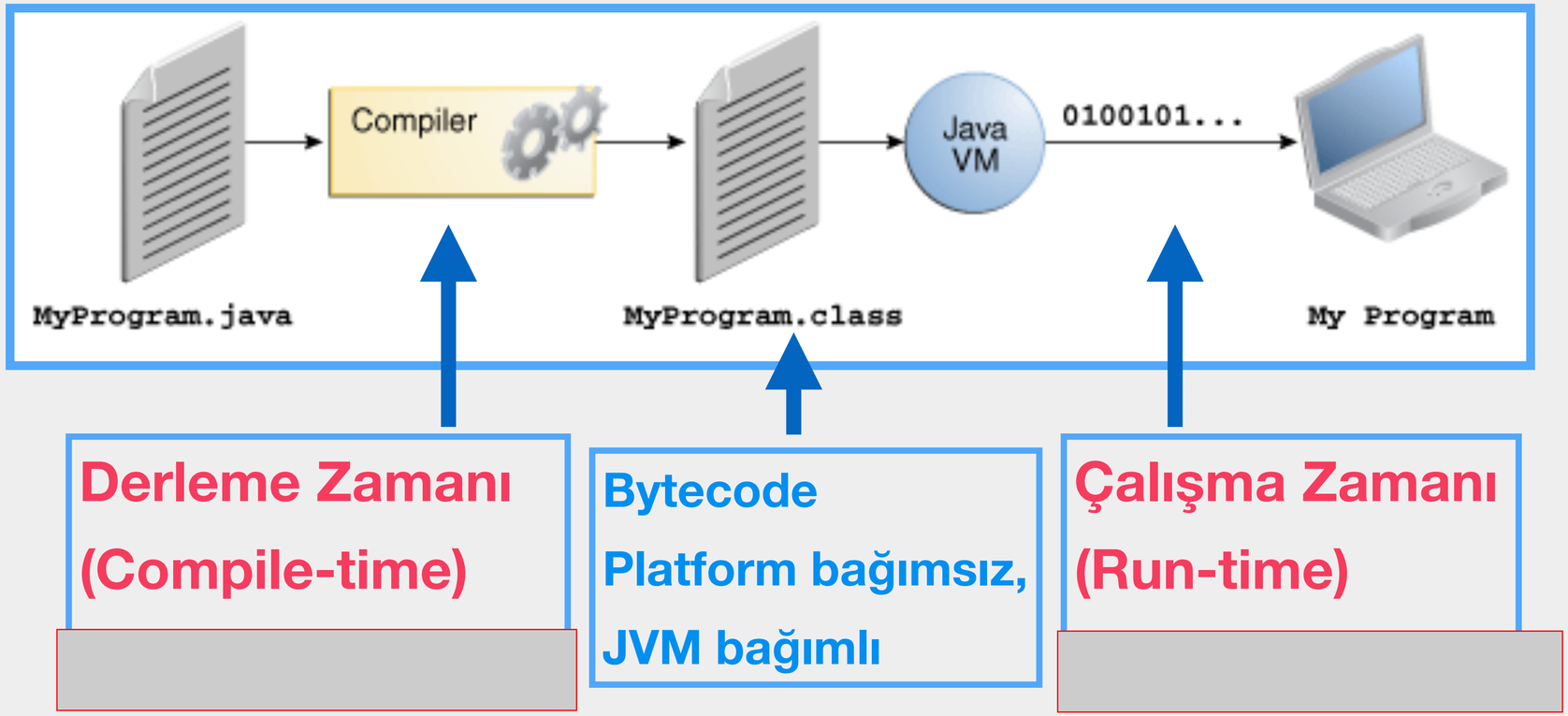
- Yorumlama, kodu daha kırılgan ama dinamik ve platformdan bağımsız yapar.
  - Hatalar çalışma zamanında ortaya çıkar,
  - Kaynak koda yapılan değişiklikler çalışma zamanında doğrudan görünür.
- Diller arasındaki bu farklar **Just-in-Time Compiler** teknolojisi sayesinde azalmıştır.

Java Derlenir ve Yorumlanır

- **Java hem derlenir (compiled) hem yorumlanır (interpreted).**
- Java kaynak kodları (`java` uzantılı dosyalar) doğrudan makina koduna derlenmez, **bytecode**lardan oluşan `class` uzantılı dosyalara derlenir.
- Bytecodelar ise çalışma zamanında JVM tarafından yorumlanır.

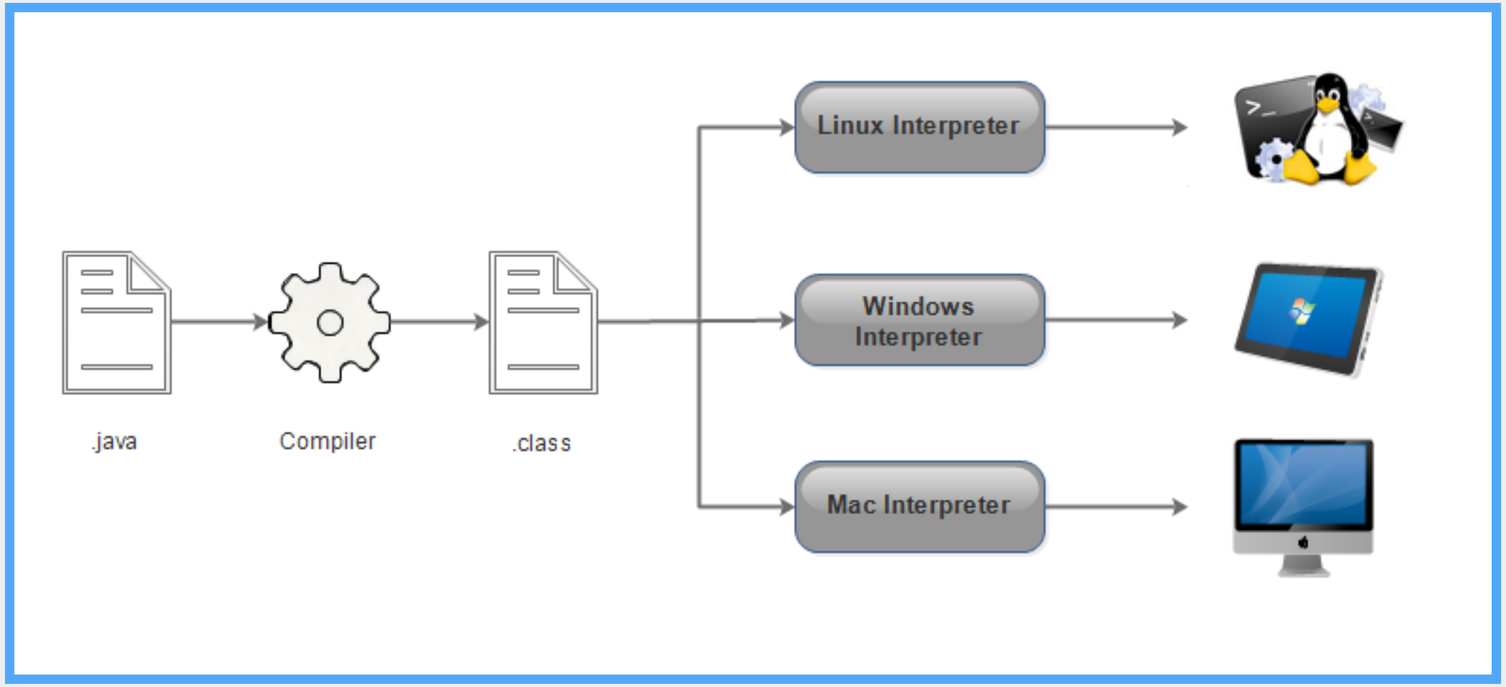
Derleme ve Çalışma Zamanları

- Kaynak kodun derlendiği zamana **derleme zamanı** (**compile-time**), çalıştırıldığı zamana ise **çalışma zamanı** (**run-time**) denir.



Bytecodeun Yorumlanması

- Bytecodelardan oluşan `class` dosyası, platformdan bağımsızdır ve JVM'in olduğu herhangi bir platformda JVM tarafından yorumlanır, çalıştırılır.



Bytecode

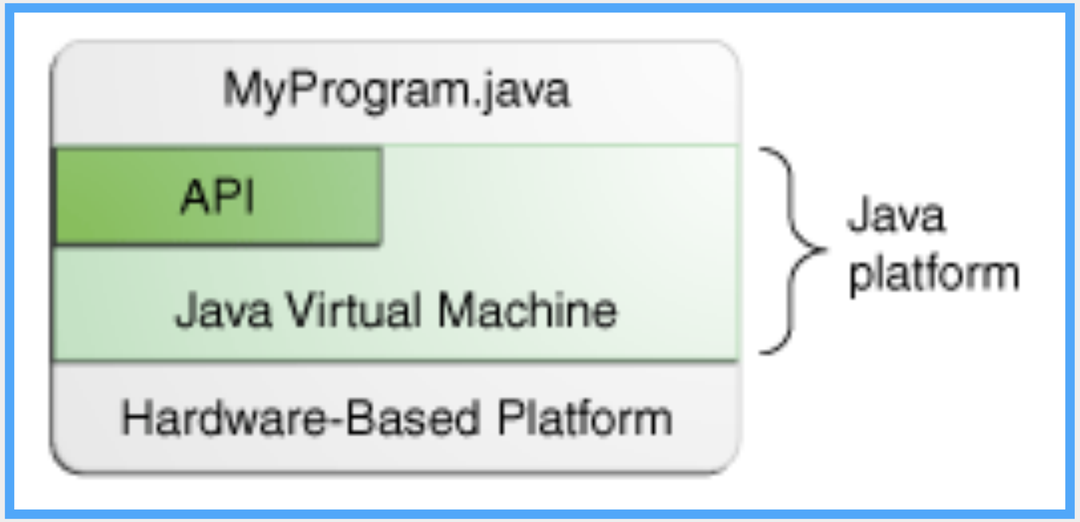
- **Bytecodelar, JVM'in komutlarıdır**,
  - Bir byte uzunluğunda oldukları için böyle adlandırılmıştır.
- Bytecode, herhangi bir işlemciye özel değildir ya da başka bir ifadeyle sadece JVM'e özeldir,
- Bytecodelar, JVM tarafından çalışma zamanında platforma özel makina dili komutlarına çevrilir, yorumlanır.
  - Bu şekilde Java'nın platformdan bağımsız ve taşınabilir olması sağlanır.

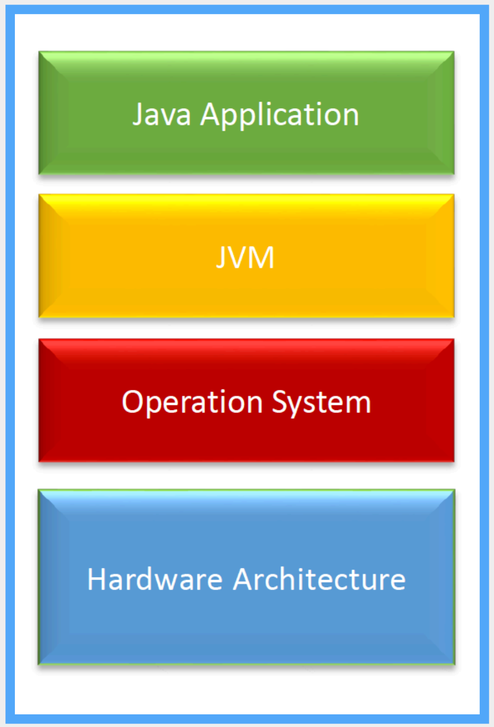
JVM, Java Virtual Machine - I

- **JVM (Java Virtual Machine ya da Java Sanal Makinası), sanal bir makinadır.**
  - Derlenmiş Java kodlarını, bytecode, çalıştıran sistemdir,
  - Genelde içinde Just-in-Time Compiler barındırır,
  - JRE'nin (Java Runtime Environment) ana parçasıdır,
  - Genel olarak C/C++ ve Java ile yazılır,
  - Java'nın çalışması beklenen her platformda kurulu olmalıdır.

Java Platform

- JVM'i, geliştiriciye bakan yüzü standart olan ama platforma bakan yüzü, onunla konuşabilecek şekilde, geliştiriciyi platformdan soyutlayan bir ara katman olarak düşünebilirsiniz.





Soru ve Cevap Zamanı!

selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

İlk Java Kodumuz

selsoft

build better, deliver faster