به نام خدا

# برنامه‌سازی پیشرفته

دانشگاه شهید بهشتی · دانشکده مهندسی و علوم کامپیوتر

دکتر مجتبی وحیدی اصل

## آشنایی با جاوا

آراس ولی‌زاده

# فهرست مطالب

1. چرا جاوا؟
2. شاخص‌های محبوبیت زبان‌های برنامه‌نویسی
3. خصوصیات زبان جاوا
4. ابزارهای لازم
5. سرفصل‌های درس
6. نکات مهم موفقیت در درس

# مقدمه

به درس **برنامه‌سازی پیشرفته** خوش اومدید :) در این درس با زبان **جاوا**، یکی از پرکاربردترین زبان‌های برنامه‌نویسی جهان، آشنا می‌شویم. در این دوره، از مفاهیم پایه‌ای شروع می‌کنیم و تا مباحث پیشرفته‌ای مانند شیءگرایی، مدیریت استثناها و برنامه‌نویسی چندریسمانی پیش خواهیم رفت. هدف ما این هستش که شما نه تنها با سینتکس و ساختار زبان جاوا آشنا بشید، بلکه نحوه تفکر برنامه‌نویسی حرفه‌ای و حل مسئله با رویکرد شیءگرا را نیز فرا بگیرید. این مهارت‌ها پایه‌ای محکم برای ورود به دنیای توسعه نرم‌افزار و کسب تجربه در بازار کار فراهم خواهد کرد.

**نکته:** این دوره نیازمند تمرین مداوم و حل مسائل عملی است. پس آماده باشید تا با کدنویسی زیاد، مهارت‌های خود را تقویت کنید!

# چرا جاوا

علاوه بر یادگیری ساختار و ویژگی‌های جاوا، با مفاهیم **برنامه‌نویسی شیءگرا** (OOP) که محبوب‌ترین متدولوژی توسعه نرم‌افزار است نیز آشنا خواهید شد.

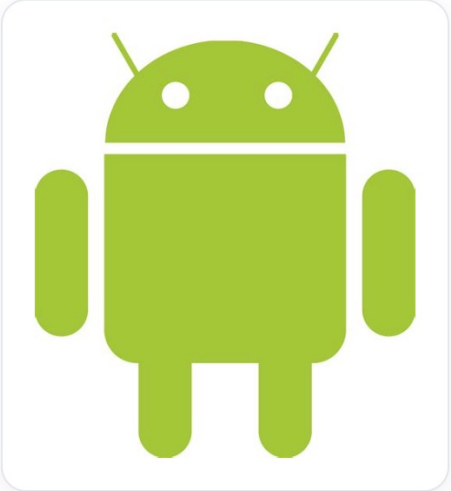
در ادامه، نگاهی بر برنامه‌نویسی در **فریم‌وُرک فلاتر** نیز خواهیم داشت تا بتوانید از دانش خود در توسعه اپلیکیشن‌ها استفاده کنید.

در اکوسیستم نرم‌افزار امروز، **جاوا** همچنان بین زبان‌های پراستفاده قرار دارد: بر اساس نظرسنجی سال ۲۰۲۵ استک‌اُورفلو، حدود **۲۹–۳۰٪** از توسعه‌دهندگان حرفه‌ای در یک سال گذشته با جاوا کار کرده‌اند، و جاوا در شاخص TIOBE نیز در جمع ۴ زبان برتر است.

جاوا زبان رسمی توسعه اندروید می‌باشد. بسیاری از نرم‌افزارهای تحت وب client-server به جاوا نوشته می‌شوند و میلیون‌ها برنامه‌نویس جاوا در این حوزه فعال هستند.

از منظر بازار، هزینه مصرف‌کننده در اپ‌های موبایل در سال ۲۰۲۴ حدود **۱۵۰ میلیارد دلار** بوده (۱۳٪ رشد نسبت به ۲۰۲۳) که نشان می‌دهد فرصت اقتصادی اکوسیستم اپ بسیار بزرگ است؛ بخش قابل‌توجهی از این اپ‌ها به‌صورت بومی یا هیبرید برای اندروید توسعه می‌یابند.

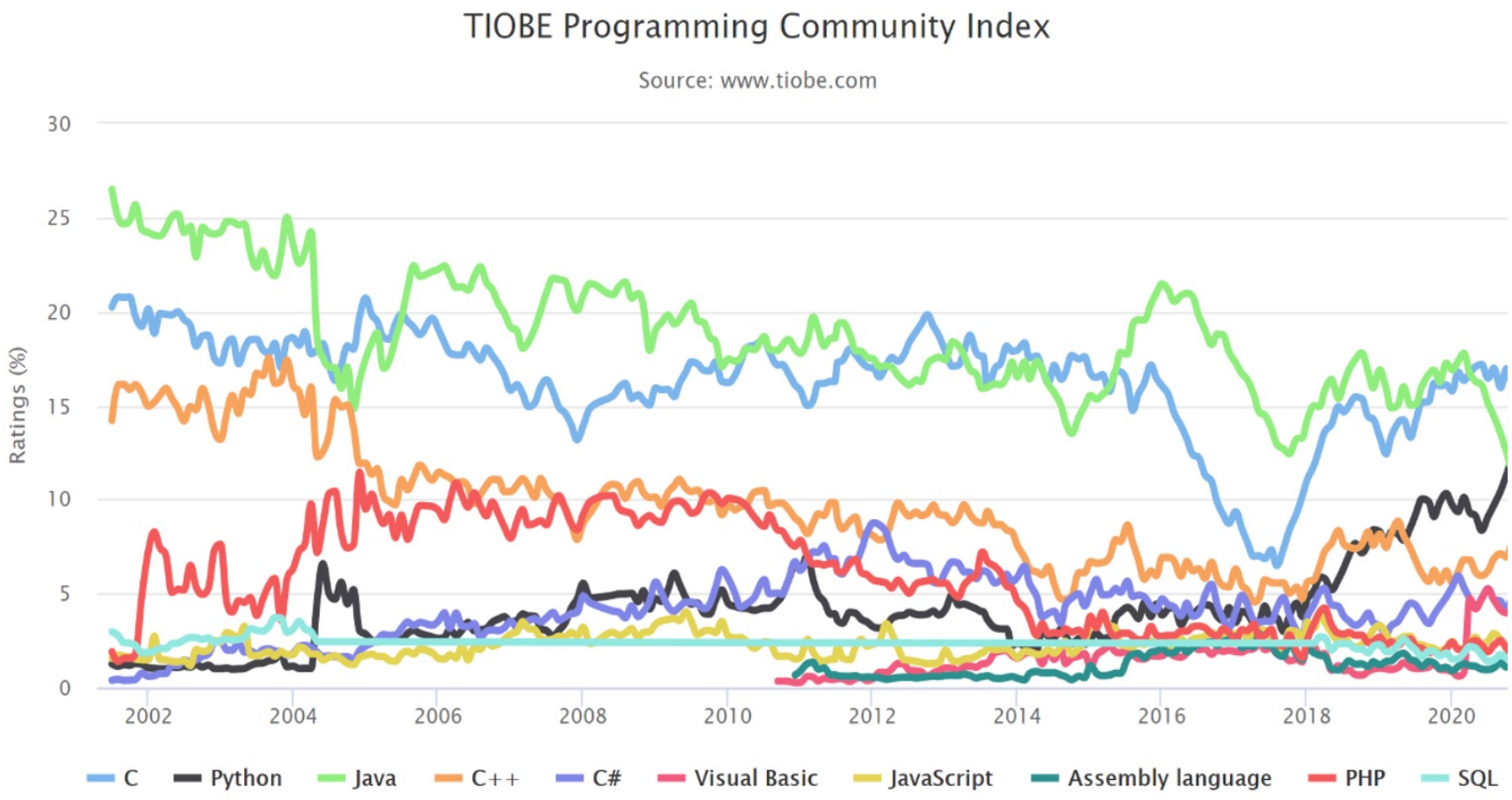
در سمت سرور، جاوا همچنان ستون فقرات بسیاری از سامانه‌های وب enterprise و کلاینت–سرور است (به‌ویژه با اکوسیستم Spring)، و به‌دلیل پایداری ABI/JVM و ابزار بالغ، برای سرویس‌های در مقیاس سازمانی انتخاب رایجی محسوب می‌شود.



# چرا جاوا محبوب است؟

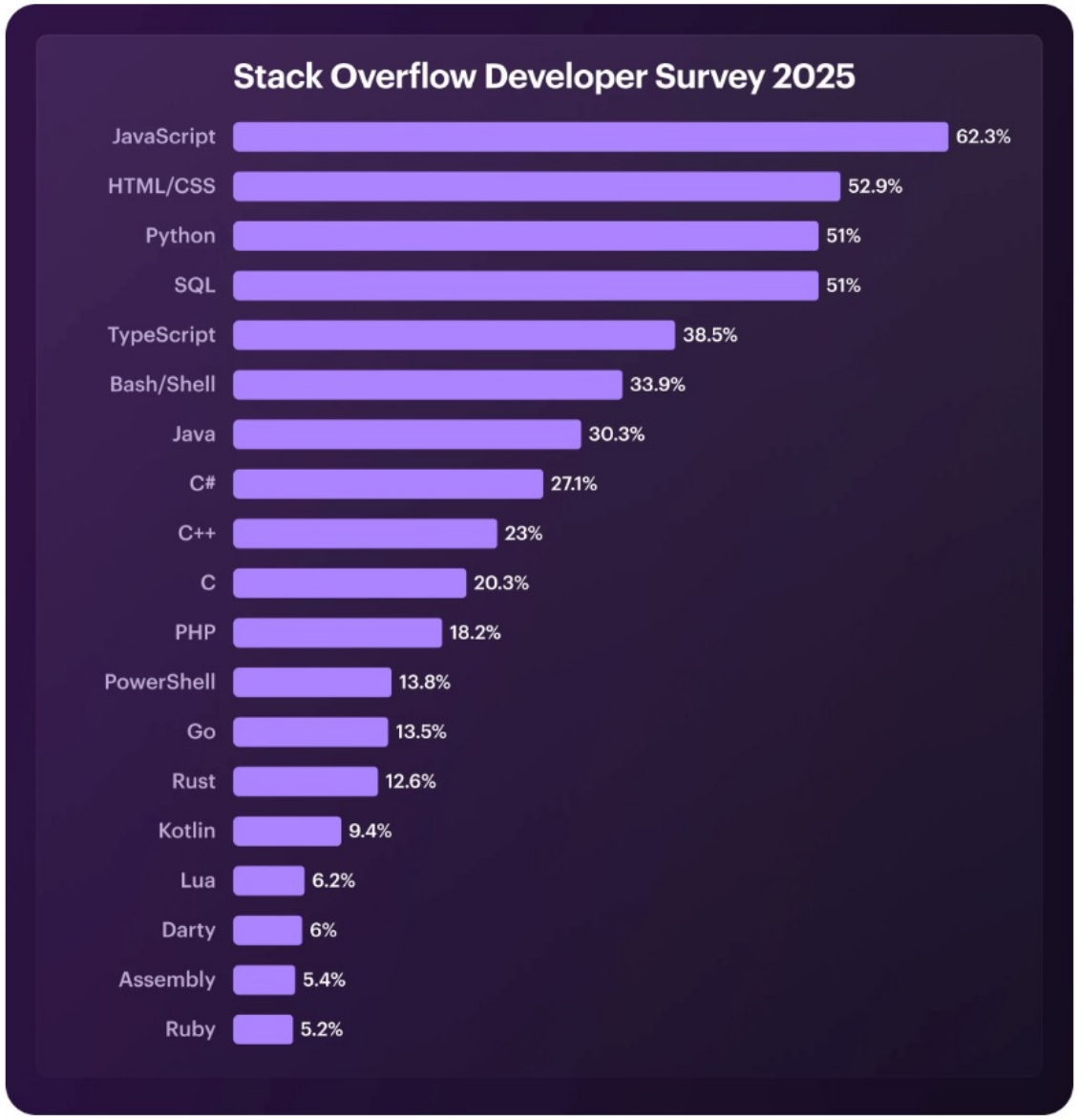
محبوبیت یک زبان برنامه‌نویسی را می‌توان از چند شاخص مهم سنجید:

- تعداد آگهی‌های شغلی که به دنبال برنامه‌نویس مسلط به آن زبان هستند.
- میانگین حقوق و مزایای پرداختی به توسعه‌دهندگان آن زبان
- میزان فروش و تنوع کتاب‌ها و منابع آموزشی مرتبط
- تعداد کدها و پروژه‌های متن‌باز و نمونه‌کدهای منتشرشده در اینترنت
- حضور فعال در جوامع و انجمن‌های برنامه‌نویسی

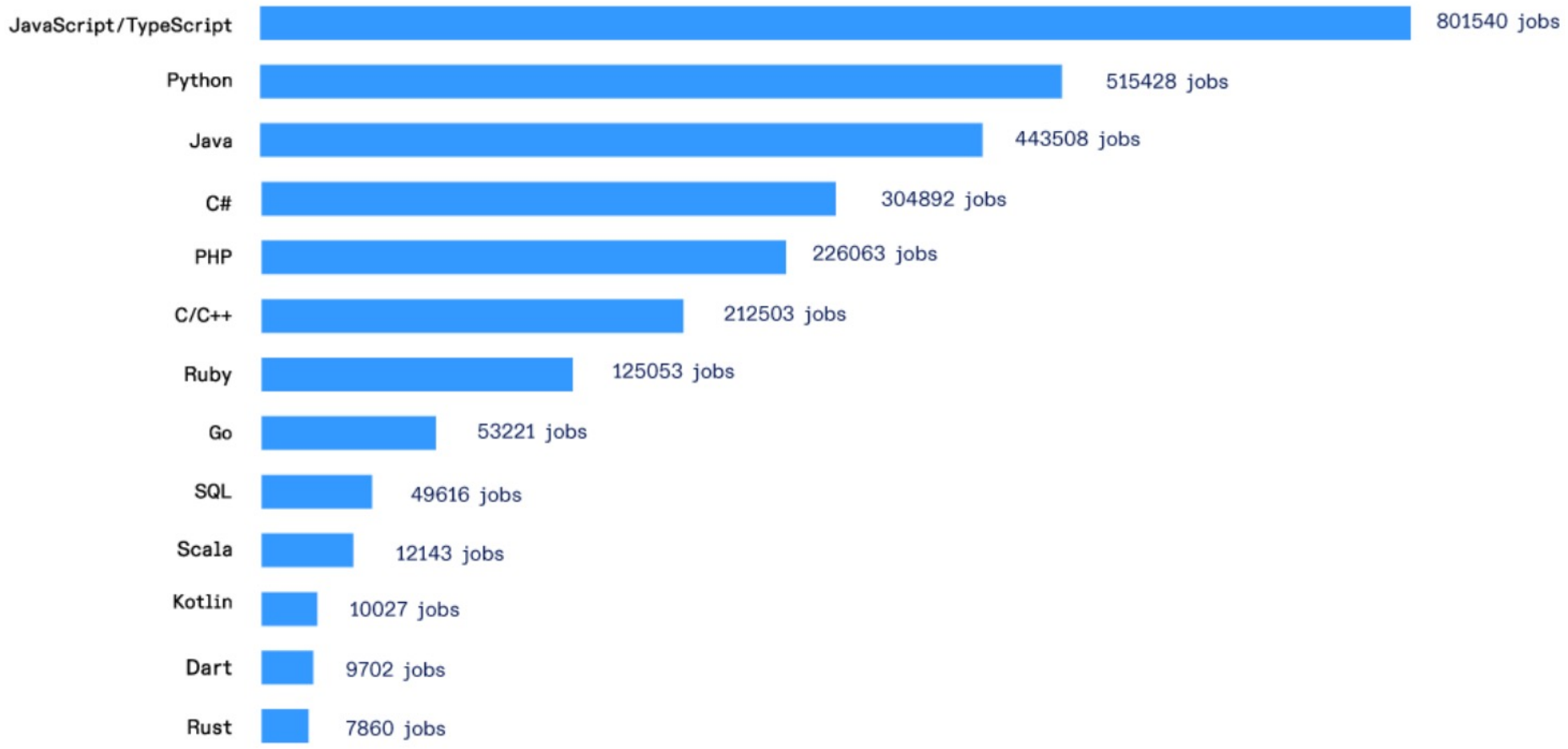

TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
C
Python
Java
C++
C#
Visual Basic
JavaScript
Assembly language
PHP
SQL

# The TIOBE Ranking 2025

| FEB 2025 | FEB 2024 | LANGUAGE | RATINGS | CHANGE |
|---|---|---|---|---|
| 1 | 1 | Python | 23.88% | +8.72% |
| ^ 2 | 3 | C++ | 11.37% | +0.84% |
| ^ 3 | 4 | Java | 10.66% | +0.79% |
| v 4 | 2 | C | 9.84% | -1.14% |
| 5 | 5 | C# | 4.12% | -3.41% |


Stack Overflow Developer Survey 2025
JavaScript 62.3%
HTML/CSS 52.9%
Python 51%
SQL 51%
TypeScript 38.5%
Bash/Shell 33.9%
Java 30.3%
C# 27.1%
C++ 23%
C 20.3%
PHP 18.2%
PowerShell 13.8%
Go 13.5%
Rust 12.6%
Kotlin 9.4%
Lua 6.2%
Darty 6%
Assembly 5.4%
Ruby 5.2%

# Most Demanded Programming Languages by Number of Jobs

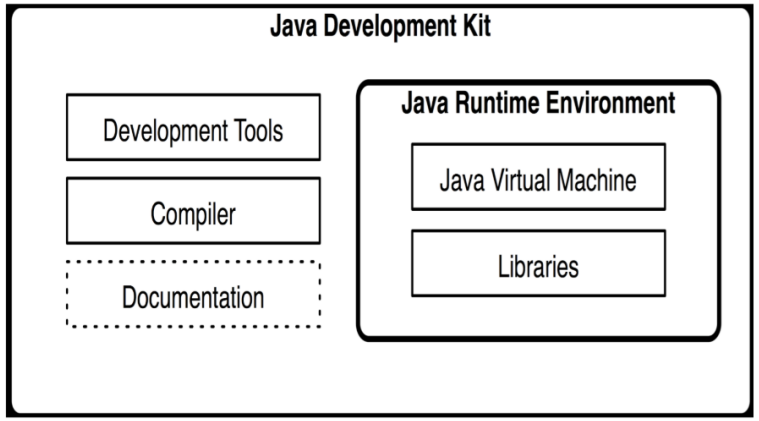


# Java Development Kit (JDK)

برای نوشتن، **کامپایل** و **اجرای** برنامه‌های جاوا به **JDK** نیاز داریم؛ داشتن **JRE** به‌تنهایی فقط اجرای برنامه را ممکن می‌کند.

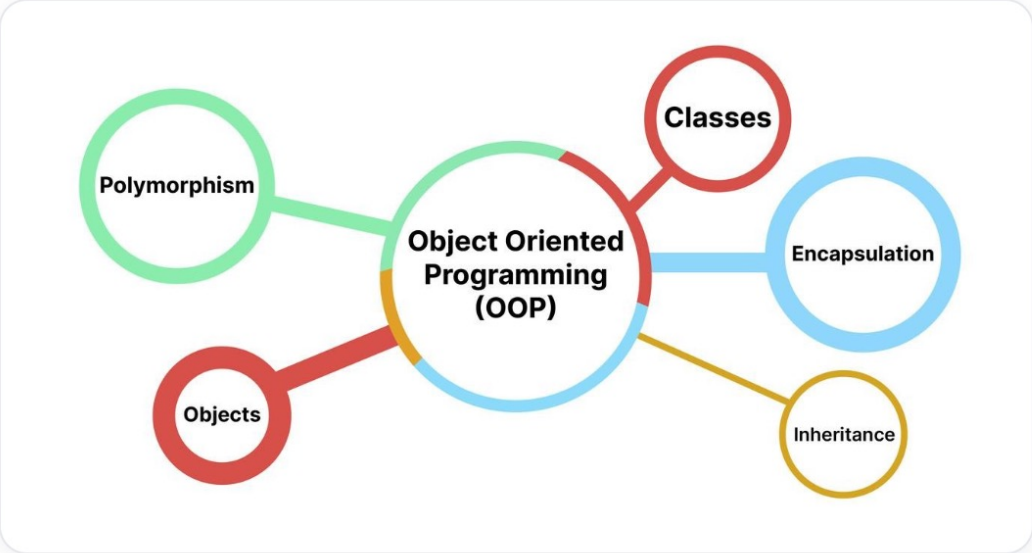
- **Development Tools** — ابزارهای توسعه مانند `javac` (کامپایل)، `java` (اجرا)، دیباگر و ابزار پکیج/مستندسازی.
- **Compiler** — تبدیل کد منبع `.java` به بایت‌کد `.class` قابل‌اجرا روی JVM.
- **Documentation** — مستندات زبان و APIها برای توسعه سریع‌تر و درست‌تر.
- **JRE** — شامل:
  - **JVM** برای اجرای بایت‌کد.
  - **Libraries** کتابخانه‌های استاندارد برای ورودی/خروجی، شبکه، کالکشن‌ها و ...

جمع‌بندی: **JDK = ابزارهای توسعه + JRE**، در حالی‌که **JRE = فقط اجرای برنامه**.

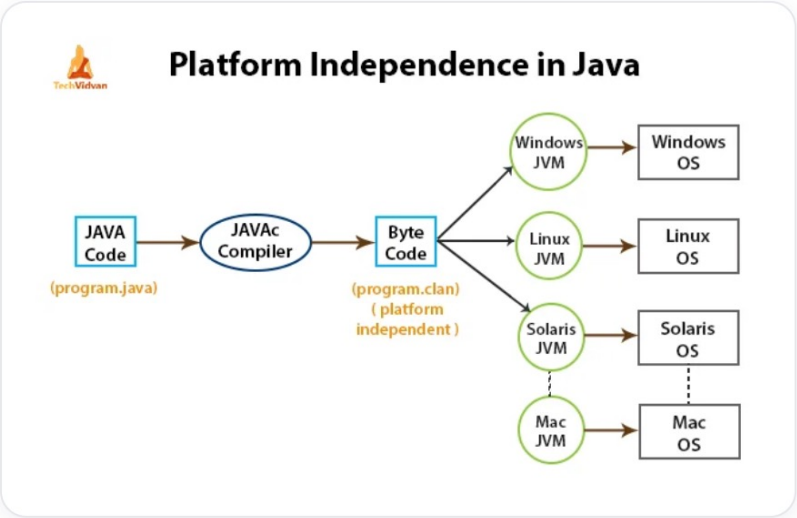


# خصوصیات زبان جاوا

- **سادگی:** سینتکس آشنا شبیه C اما بدون اشاره‌گر؛ مدیریت حافظه خودکار با Garbage Collection.
- **شیءگرایی (Object-Oriented):** مدل‌سازی دنیای واقعی با کلاس/شیء، تعامل بهتر اجزا، و **استفاده‌مجدد از کد**.



- **قابل‌حمل و مستقل از معماری:** یک کد جاوا روی سامانه‌های مختلف بدون تغییر اجرا می‌شود.
- **ترکیب کامپایل و تفسیر:** در جاوا ابتدا کد منبع توسط کامپایلر به قالبی به نام `Bytecode` تبدیل می‌شود. این کد میانی مستقل از سیستم‌عامل است و توسط **ماشین مجازی جاوا (JVM)** در زمان اجرا تفسیر یا به کد بومی ترجمه می‌شود. این فرآیند باعث می‌شود برنامه بدون تغییر روی سیستم‌عامل‌ها و معماری‌های مختلف اجرا شود (Platform-Independent).




- **چندریسمانی (Multithreading):** قابلیتی که به برنامه اجازه می‌دهد چند بخش (Thread) را به‌طور همزمان اجرا کند. این ویژگی باعث می‌شود وظایف سنگین مانند پردازش داده، رندر گرافیک، یا ارتباطات شبکه‌ای بدون ایجاد توقف در عملکرد اصلی برنامه انجام شوند. جاوا ابزارها و کلاس‌های داخلی متنوعی برای مدیریت Threadها، همگام‌سازی (Synchronization) و جلوگیری از تداخل داده‌ها فراهم می‌کند، که در نتیجه کارایی و پاسخ‌گویی برنامه بهبود می‌یابد.
- **امن و قابل اتکا (Secure & Robust):** مدل امنیتی قوی، بررسی خطا در زمان کامپایل/اجرا، عدم دسترسی مستقیم به حافظه.

# اصل WORA

یکی از مهم‌ترین مزیت‌های جاوا اصل **یک بار بنویس، همه‌جا اجرا کن** است. این یعنی شما یک بار کد را می‌نویسید و پس از کامپایل به فایل `.class`، این کد می‌تواند روی هر سیستم‌عاملی که **JRE** روی آن نصب شده است اجرا شود، بدون اینکه نیاز به تغییر در کد منبع باشد.

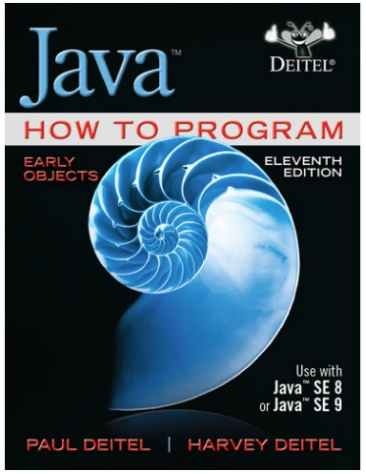
1. نوشتن کد منبع جاوا (Java Source)
2. کامپایل کد منبع توسط کامپایلر جاوا و تولید Bytecode
3. اجرای Bytecode بر روی هر محیطی که JRE دارد (Windows، Linux/Unix، macOS، Android)

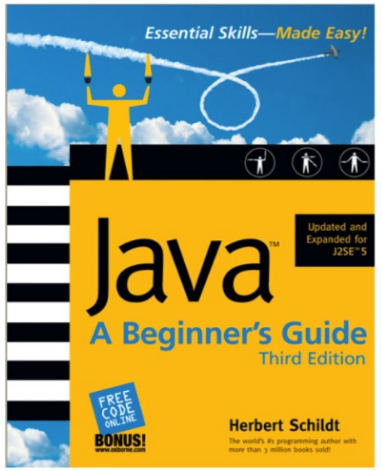
نتیجه این ویژگی، کاهش هزینه توسعه و نگهداری، و امکان توسعه آسان نرم‌افزار برای چندین پلتفرم به صورت همزمان است.

# منابع درس

برای یادگیری بهتر زبان جاوا، منابع متنوعی در دسترس هستند که هر کدام با رویکردی متفاوت به آموزش این زبان پرداخته‌اند.

- **Java: A Beginner’s Guide** – نوشته Herbert Schildt، مناسب برای شروع یادگیری از پایه.
- **Java How to Program** – نوشته Paul و Harvey Deitel، با مثال‌ها و تمرین‌های فراوان.
- **Head First Java** – نوشته Kathy Sierra و Bert Bates، رویکرد تصویری و مفهومی برای یادگیری آسان‌تر.
- **Thinking in Java** – نوشته Bruce Eckel، مناسب برای درک عمیق مفاهیم شیءگرایی و ساختار جاوا.



# اولین برنامه به زبان جاوا

در این بخش، قصد داریم اولین برنامه خود را به زبان جاوا بنویسیم. برای این کار از **Jupyter Notebook** استفاده می‌کنیم. **Jupyter** یک محیط تعاملی و قدرتمند برای نوشتن و اجرای کد، تحلیل داده و مستندسازی است که به‌طور پیش‌فرض با کرنل Python اجرا می‌شود.

چون قصد داریم کدهای جاوا را اجرا کنیم، لازم است یک کرنل جاوا (مثل JBang ) نصب کنیم تا Jupyter بتواند کدهای جاوا را پردازش و اجرا کند.

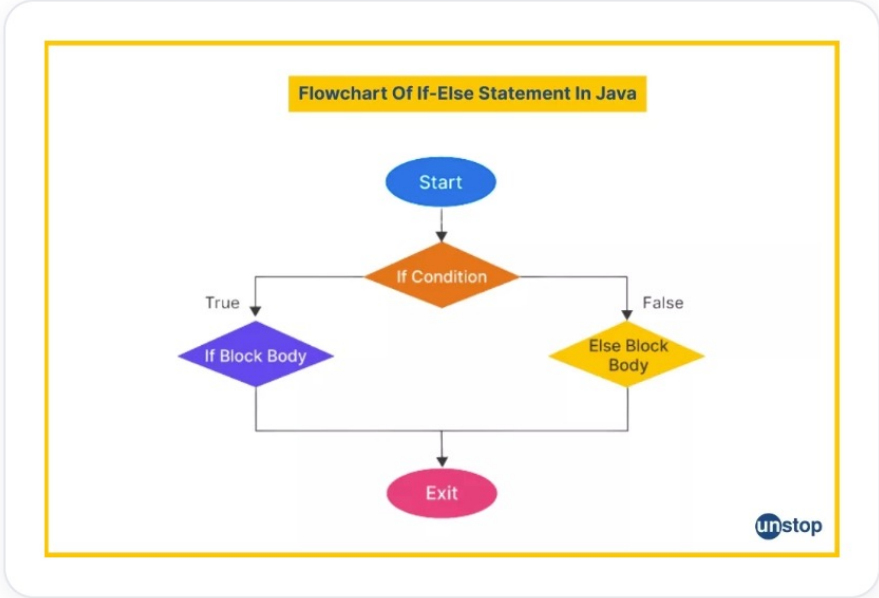
این کار به ما امکان می‌دهد تا در همان محیط تعاملی Jupyter، کد جاوا بنویسیم، اجرا کنیم و خروجی آن را بلافاصله ببینیم. ابتدا cell زیر را با استفاده از کرنل پایتون اجرا کنید و پس از نصب، می‌توانید کرنل جاوا را انتخاب کنید.

```
!pip install jbang
import jbang
jbang.exec("trust add https://github.com/jupyter-java")
jbang.exec("install-kernel@jupyter-java")




// You can also use Intellij Idea to write your codes
public class Welcome3 {
    public static void main(String[] args) {
        System.out.println("Welcome\nto\nJava\nProgramming!");
    }
}
Welcome3.main(new String[]{});
```

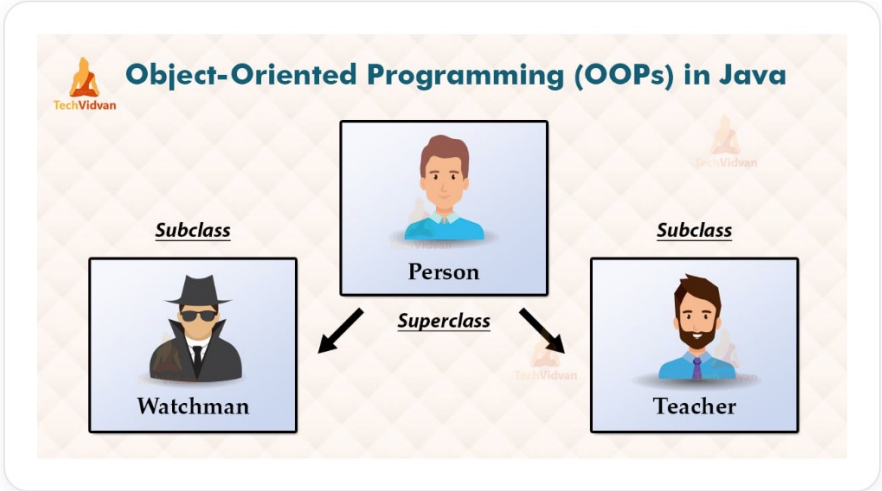
# سرفصل‌های آموزشی دوره جاوا

در این دوره، به صورت گام‌به‌گام مفاهیم کلیدی و پیشرفته زبان جاوا آموزش داده خواهد شد. مباحث شامل:

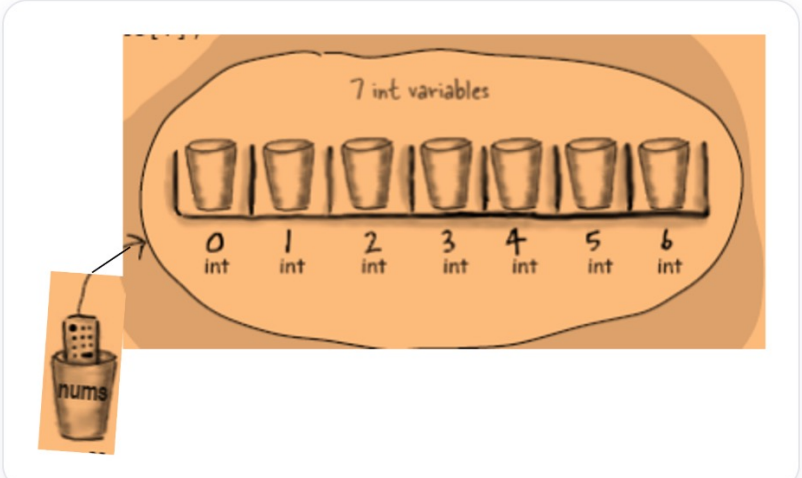
- **ساختارهای کنترلی:** شامل if ساده، switch..case، if..else، حلقه‌های do..while، while، for و دستورات break و continue.



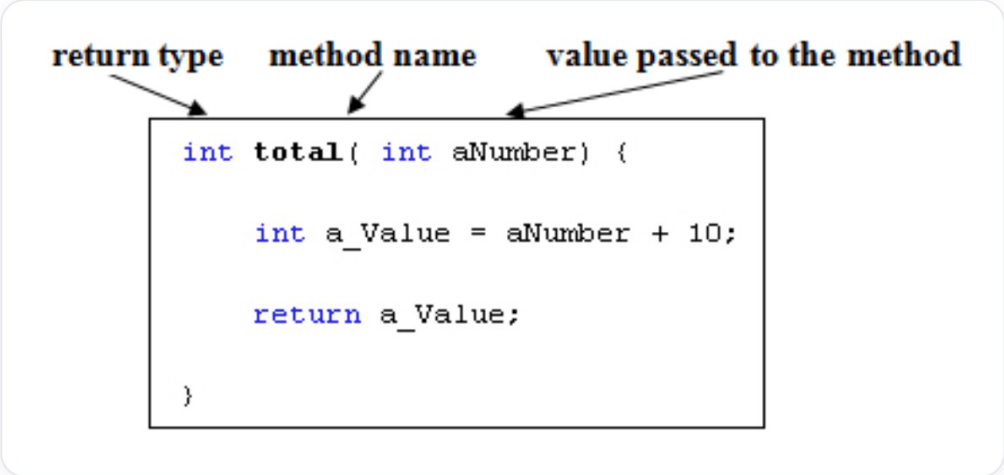
- **مفاهیم شیءگرایی پیشرفته:** آشنایی با ارث‌بری، پلی‌مورفیسم و نحوه استفاده مؤثر از کلاس‌ها، متدها و متغیرها.




- **آرایه‌ها و رشته‌ها:** آشنایی با کلاس String و Regex، کار با آرایه‌های یک‌بعدی و چندبعدی، ایجاد، تغییر و حذف عناصر و ارسال آرایه‌ها به متدها.

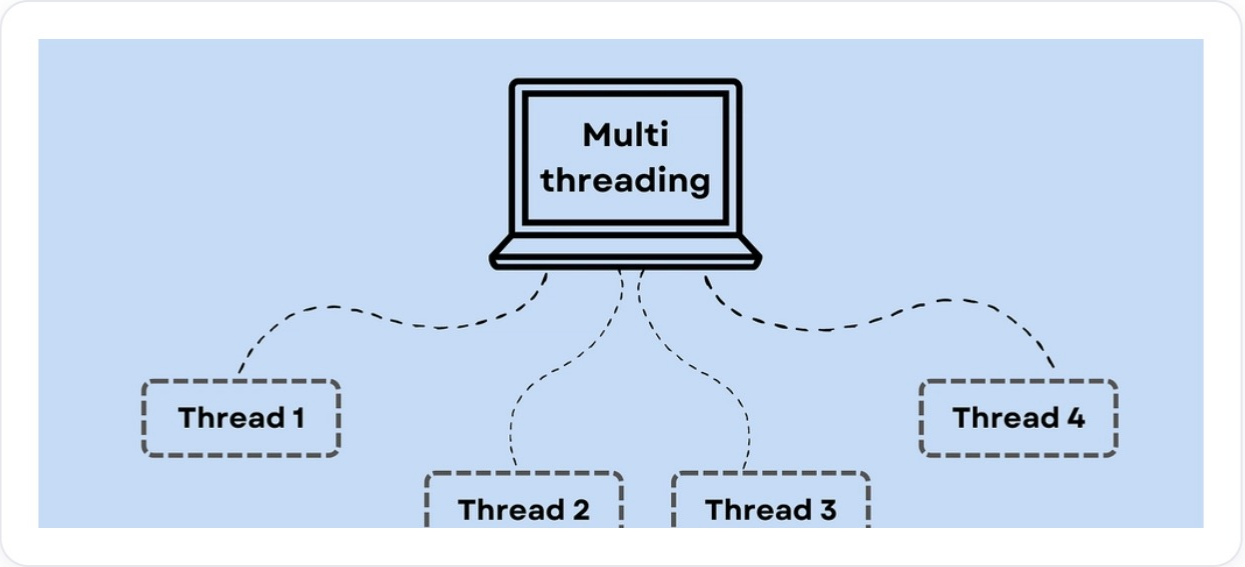


- **متدها و ماژول‌ها:** متدهای ایستا/غیراستا، فلیدهای ایستا/غیر ایستا، تعریف متدهای چندپارامتری، آشنایی با بسته‌های API جاوا، سربارگذاری و بازنویسی متدها.




- **مدیریت استثناها:** سلسله‌مراتب استثناها، استثناهای زنجیره‌ای، تعریف استثناهای جدید و نحوه کنترل خطاها در برنامه.

- **سایر مباحث پیشرفته:** کار با فایل‌ها و ورودی/خروجی در جاوا، مقدمات برنامه‌نویسی شبکه، مفاهیم مقداردهی ایستا و پویا، برنامه‌نویسی چندریسمانی (Multithreading)، کار با مجموعه‌ها (Collections)، برنامه‌نویسی جنریک (Generic Programming) و بسیاری مطالب دیگر.



# چند نکته مهم

- **اهمیت تمرین پیوسته:** تمرین مداوم یکی از کلیدی‌ترین عوامل موفقیت در یادگیری این درس است. توصیه می‌شود علاوه بر تمرین‌های ارائه شده در کلاس، به صورت مستقل نیز مسائل و مثال‌های بیشتری را حل کنید. این کار باعث تثبیت مفاهیم و افزایش مهارت شما در حل مسئله می‌شود.
- **شرط قبولی در درس:** برای پاس کردن این درس، لازم است که دانشجو حداقل ۵۰٪ از مجموع نمره میان‌ترم و پایان‌ترم را کسب نماید.
- **همکاری در انجام تمرین‌ها:** می‌توانید برای فهم بهتر تمرین‌ها با همکلاسی‌های خود مشورت کنید و ایده‌ها را به اشتراک بگذارید. با این حال، اجرای نهایی تمرین‌ها باید توسط خودتان انجام شود تا هم مهارت‌های فردی‌تان تقویت شود و هم اصول اخلاق آکادمیک رعایت گردد. هرگونه کپی‌برداری مستقیم، تخلف محسوب می‌شود.
- **ابزار های هوش مصنوعی در درس ما:** استفاده از ابزار های هوش مصنوعی برای نوشتن کد جایز نیست و درصورت مشاهده تقلب تلقی می‌شود :) همچنین تمارین و پروژه درس توسط سیستم تقلب یاب کوئرا چک می‌شود و درصورت مشاهده درصد های بالا این مورد هم تقلب تلقی می‌شود.
- **کوییزها و ارزیابی مستمر:** در طول ترم تعدادی کوییز (توسط تیم حل‌تمرین) برگزار خواهد شد که تاریخ برگزاری آن‌ها از قبل اطلاع‌رسانی می‌شود. این کوییزها نقش مهمی در ارزیابی پیوسته پیشرفت شما دارند و به شما کمک می‌کنند نقاط ضعف خود را شناسایی و برطرف کنید.
- **مدیریت زمان:** پیشنهاد می‌شود زمان خود را به‌گونه‌ای مدیریت کنید که علاوه بر حضور فعال در کلاس، فرصت کافی برای مطالعه، انجام تمرین‌ها و مرور مطالب داشته باشید. یادگیری این درس نیازمند پیگیری منظم و مستمر است. همچنین حضور در کلاس های حل‌تمرین اجباری می‌باشد.
- **سیلابس درس:** سیلابس کامل درس در گیت‌هاب مربوط به این درس قرار داده شده است. برای اطلاع از بارم‌بندی بخش‌های مختلف و آشنایی با برنامه آموزشی ترم، حتما به آن مراجعه کنید.
- **ارتباط با تیم آموزشی:** در صورت بروز هرگونه مشکل یا نیاز به مشورت، با سرپرست دستیاران آموزشی در ارتباط باشید. همچنین برای ارتباط مستقیم با استاد درس، می‌توانید از طریق ایمیل mo_vahidi@sbu.ac.ir اقدام کنید.

# مطالعه بیشتر: تاریخچه و کاربردهای جاوا

زبان **جاوا** در اوایل دهۀ ۹۰ میلادی در زمانی متولد شد که صنعت نرم‌افزار با یک چالش جدی روبه‌رو بود: هر سیستم‌عامل و سخت‌افزار زبان و محیط توسعۀ مخصوص به خود را داشت و انتقال یک نرم‌افزار از یک پلتفرم به دیگری هزینه و زمان زیادی می‌برد. شرکت **Sun Microsystems** تصمیم گرفت زبانی طراحی کند که مستقل از سخت‌افزار و سیستم‌عامل باشد. نتیجۀ این تلاش ماشین مجازی جاوا (Java Virtual Machine - JVM) و شعار مشهور **«یک‌بار بنویس، همه‌جا اجرا کن»** بود. ایدۀ اصلی این بود که برنامه‌ها به جای اجرا شدن مستقیم روی سیستم‌عامل، روی یک ماشین مجازی اجرا شوند که خودش با هر سیستم‌عامل هماهنگ می‌شود. این نوآوری باعث شد جاوا به سرعت جایگاه ویژه‌ای در توسعۀ نرم‌افزار پیدا کند، به‌خصوص در پروژه‌هایی که نیاز به پایداری، امنیت و قابلیت حمل بالا داشتند.

## پیدایش (۱۹۹۱–۱۹۹۵)

در سال ۱۹۹۱، **James Gosling** به همراه تیمی از مهندسان Sun Microsystems پروژه‌ای با نام رمز Oak را آغاز کرد. هدف اولیه، ساخت زبانی سبک و امن برای دستگاه‌های تعاملی و سیستم‌های توکار (مانند ست‌تاپ‌باکس‌ها) بود. این زبان باید بتواند بدون وابستگی به سخت‌افزار، در محیط‌های مختلف اجرا شود. با رشد اینترنت در اوایل دهۀ ۹۰، تیم توسعه متوجه شد که همین قابلیت «چندسکویی» می‌تواند برای وب نیز بسیار ارزشمند باشد. به همین دلیل، در سال ۱۹۹۵ این زبان با نام جدید **Java** معرفی شد و همراه با مرورگر HotJava و تکنولوژی اپلت (Applet) به جهان عرضه گردید.

## رشد چشمگیر کجا رخ داد؟

- **وب سازمانی (اواخر دهۀ ۹۰ و ۲۰۰۰):** با معرفی Servlet/JSP و سپس **J2EE**، جاوا به هستۀ اصلی سیستم‌های سازمانی در بانکداری، مخابرات و تجارت الکترونیک تبدیل شد.
- **اندروید (از ۲۰۰۸):** عرضۀ Android باعث شد میلیون‌ها توسعه‌دهنده وارد اکوسیستم جاوا شوند و برنامه‌های موبایل خود را با این زبان بنویسند.
- **زیرساخت‌های مقیاس‌پذیر:** فریم‌ورک‌هایی مانند **Spring Boot** و فناوری‌های مبتنی بر JVM (مثل Kafka و Hadoop) راه را برای توسعه سیستم‌های ابری و مقیاس‌پذیر هموار کردند.

## استفادۀ شرکت‌های بزرگ

- **Google:** بخش عمدۀ اکوسیستم اندروید و سرویس‌های سمت سرور.
- **Amazon:** سرویس‌های مقیاس‌پذیر و زیرساخت AWS.
- **Netflix:** سیستم‌های پخش محتوا و معماری میکروسرویس.

# A SHORT HISTORY OF JAVA

## JAVA VERSION HISTORY

### 1997 — JDK 1.1

**Major Changes**

Extensive retooling of the AWT event model and 'inner classes' added to the language: JavaBeans and JDBC.

### 1998 — J2SE 1.2

**Major Changes**

Codename Playground; rebranded as Java 2 and the version name changed to J2SE (Java 2 Standard Edition).

### 2000 — J2SE 1.3

**Major Changes**

Codename Kestrel; bundled with Hotspot JVM, JavaSound, Java Naming and Directory Interface (JNDI) and Java Platform Debugger Architecture.

### 2002 — J2SE 1.4

**Major Changes**

Codename Merlin; first release of the Java platform developed under the Java Community Process as JSR 59. Included regular expressions modeled after Perl.

### 2004 — J2SE 5.0

**Major Changes**

Codename Tiger; originally numbered 1.5 which is still used as its internal version number. Added several new language features such as the for-each loop, generics, autoboxing and var-args.

### 2006 — JAVA SE 6

**Major Changes**

Codename Mustang; bundled with a database manager and facilitates the use of scripting languages with the JVM. Replaced the name J2SE with Java SE and dropped the .0 from the version number.

### 2011 — JAVA SE 7

**Major Changes**

Codename Dolphin; added small language changes including strings in switch. The JVM was extended with support for dynamic languages.

### 2014 — JAVA SE 8

**Major Changes**

Language level support for lambda expressions and default methods and a new date and time API inspired by Joda Time.

### 2017 — JDK 9

**Major Changes**

Project Jigsaw; designing and implementing a standard module system for the Java SE platform, and to apply that system to the platform itself and the the JDK.

PEARSON FRANK. | JAVA, WEB & PHP RECRUITMENT EXPERTS

WWW.PEARSONFRANK.COM

Pearson Frank @PearsonFrank_ Pearson Frank

# خودمون رو بسنجیم

این بخش برای این طراحی شده که در پایان مطالعه این اسلاید، بتونی خودت رو محک بزنی و ببینی آیا مفاهیم رو به خوبی یاد گرفتی یا نه. سوالات زیر رو مرور کن و سعی کن بدون نگاه کردن به متن درس، به اون ها پاسخ بدی.

- **JDK** چیه و یک فایل جاوا چه مراحلی رو طی می‌کنه تا اجرا بشه؟
- جاوا چطوری مستقل از پلتفرم هستش؟
- داخل تمرین‌های درس چه شکلی قراره به هم کمک کنیم؟ :))

## پایان

در صورت هرگونه سوال یا پیشنهاد میتونید با من در ارتباط باشید :)

gmail: arasvalizadeh@gmail.com
telegram: @arasvalizadeh