جامعة أم القرى
كلية الحاسب الآلي قسم علوم الحاسب
والذكاء الاصطناعي

CS1701T: البرمجة الحاسوبية بلغة Java

الفصل الدراسي : 2026/27 1st

المحاضرة 1

الخوارزميات ومقدمة إلى Java (الجزء 1)

المخطط العام

- البرامج ولغات البرمجة.
- لغة البرمجة Java.

البرامج ولغات البرمجة

البرامج

- الـ برنامج هو مجموعة من التعليمات التي يتبعها الحاسوب، والتي توجهه لتنفيذ المهام التي تريدها منه وإنتاج النتائج التي ترغب في الحصول عليها.

- البرنامج يشبه الوصفة؛ فهو يحتوي على قائمة من المتغيرات (تسمى المكونات) وقائمة من العبارات (تسمى التوجيهات) التي تخبر الحاسوب بما يجب فعله بهذه المتغيرات.
- يطلق على اتباع هذه التعليمات اسم تشغيل أو تنفيذ البرنامج
- نحن نستخدم البرامج بشكل يومي تقريبا (البريد الإلكتروني، معالجات النصوص، ألعاب الفيديو، أجهزة الصراف الآلي، إلخ).
- يخبر المبرمجون الحاسوب بما يجب عليه فعله من خلال البرامج. فبدون البرامج، يكون الحاسوب مجرد آلة فارغة.

- لا تفهم الحواسيب اللغات البشرية، لذا نحتاج إلى استخدام لغات البرمجة للتواصل معها.

- تكتب البرامج باستخدام لغات البرمجة.

كلمات هذه الصفحة

instructions — تعليمات · أوامر برمجية يتبعها الحاسوب

recipe — وصفة · تشبيه لخطوات البرنامج بالتسلسل

statements — جمل برمجية · أوامر فردية داخل البرنامج

produces — ينتج · يخرج نتائج معينة

variables — متغيرات · مواضع تخزين بيانات في الذاكرة

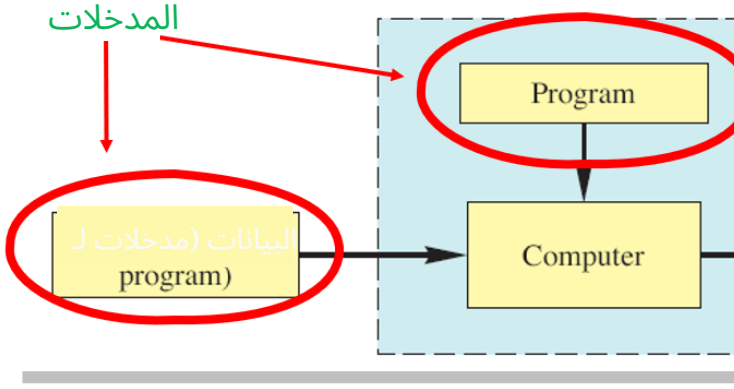
executing — تنفيذ · تشغيل البرنامج أو الأمر

word processors — معالجات النصوص · مثل برنامج وورد

تشغيل برنامج

- عادة، يتلقى الحاسوب نوعين من المدخلات:
  - البرنامج
  - البيانات التي يحتاجها البرنامج.

- الـ مخرجات هي النتيجة (أو النتائج) التي يتم إنتاجها باتباع التعليمات الموجودة في البرنامج.



هذا ما يحدث فعليا عند
شغيل البرنامج.

كلمات هذه الصفحة

input — مدخلات · البيانات التي يتلقاها البرنامج

output — مخرجات · النتيجة التي يظهرها البرنامج

لغات البرمجة

- لغات البرمجة هي مجموعة من التعليمات المكتوبة بطريقة يمكن للحاسوب فهمها

- لماذا نحتاج إلى لغات البرمجة؟
  - جميع مكونات الحاسوب: وحدة المعالجة المركزية (CPU)، والذاكرة، ووحدات الإدخال والإخراج (I/O) تفهم كود الآلة الذي يسمى النظام الثنائي (binary).
    - النظام الثنائي هو سلسلة طويلة من الأصفار والآحاد التي لا يمكن للبشر فهمها!
  - إذا، كيف يمكننا جعل الحاسوب ينفذ مهمة معينة؟
    - تساعدنا لغات البرمجة في القيام بذلك من خلال توفير حل وسيط بين الإنسان والآلة.
- تمتلك كل لغة مجموعة فريدة من الكلمات المفتاحية **(keywords)**)كلمات تفهمها اللغة) وبنية برمجية **(syntax)** خاصة لتنظيم تعليمات البرنامج.

كلمات هذه الصفحة

components — مكونات · أجزاء
الحاسب المادية

intermediate — وسيط · حل بين
طرفين

keywords — كلمات مفتاحية ·
كلمات محجوزة لها معنى خاص للغة

syntax — صيغة اللغة · قواعد كتابة
الكود البرمجي

بنية ودلالات لغات البرمجة

• البنية البرمجية (syntax)والدلالات (semantics) توفران تعريف اللغة.

• تسمى قواعد النحو لأي لغة برمجة بـ البنية ***(syntax)*** الخاصة باللغة (والتي يكتشفها المترجم/الكومبايلر).
تختلف البنية من لغة إلى أخرى.

- على سبيل المثال، جملة while في لغة Java
- while (boolean_expression) statement

• تسمى التفسيرات أو المعاني الخاصة بالجمل بـ الدلالات ***(semantic)*** للغة.

- على سبيل المثال، خروج مؤشر المصفوفة (Array index) عن النطاق المسموح:

```
int[] v = new int[10];
v[10] = 100;      // 10 ليس دليلا (index) قانونيا لمصفوفة (array) مكونة من 10 عناصر
```

- على سبيل المثال،استخدام متغير غير مهيأ (non-initialized variable):

```
int i;
++i;      // المتغير i غير مهيأ
```

مزيد من التفاصيل:

https://newtutorial2012.blogspot.com/2012/07/differentced-between-synataxsemantic.html
https://www.inf.unibz.it/~calvanese/teaching/ip/lecture-notes/uni10/node2.html

كلمات هذه الصفحة

semantics — دلالات المعنى · المعنى المقصود من الكود

boolean_expression — تعبير منطقي · تعبير نتيجته صح أو خطأ

non-initialized — غير مُهيّأ · متغير لم يتم إعطاؤه قيمة أولية

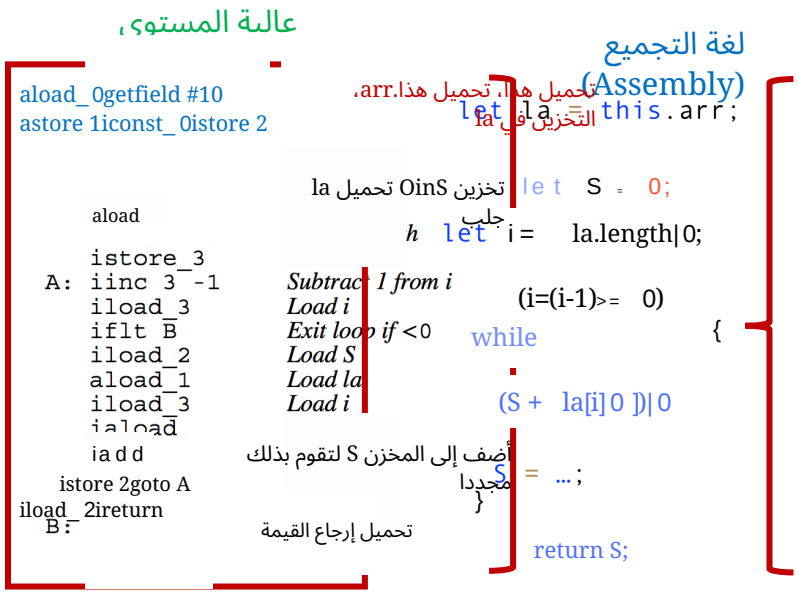
compiler — مترجم (كومبايلر) · برنامج يحول الكود إلى لغة الآلة

legal — مسموح / قانوني · في البرمجة: مسموح به حسب القواعد

لغات البرمجة (تتمة)

لغات البرمجة يمكن تصنيفها إلى:

- لغات المستوى المنخفض (لغة الآلة): هي لغات قريبة من لغة الآلة، مثل لغة التجميع (assembly language)
- لغات البرمجة عالية المستوى: هي لغات أقرب إلى لغة المبرمجين (أكثر سهولة!) مثل Java



كلمات هذه الصفحة

categorized — مصنفة · مقسّمة
إلى فئات

low-level — منخفضة المستوى ·
لغات قريبة من لغة الآلة

high-level — عالية المستوى ·
لغات قريبة من لغة البشر

مستويات لغات البرمجة

- لغات البرمجة عالية المستوى تعد سهلة الاستخدام نسبيا (أي أنها توفر مرونة أكبر، ويسهل تنفيذها وصيانتها)

- لسوء الحظ، فإن عتاد الحاسوب (computer hardware) لا يفهم اللغات عالية المستوى!
  - لذلك، يجب أن يتم تحويل برنامج اللغة عالية المستوى إلى لغة منخفضة المستوى.

كلمات هذه الصفحة

flexibility — مرونة · سهولة التعديل أو الاستخدام

implemented — يتم تنفيذه / تطبيقه · تحويل الفكرة إلى كود عملي

maintained — مُصان · تحديث البرنامج وإصلاح أخطائه

اختيار لغة البرمجة

- يعتمد على مهامك:
  - اتصالات الأقمار الصناعية (السرعة) → Assembly
  - التعليم → Basic, Pascal
  - الأعمال → Cobol
  - تطوير الويب → JavaScript, JSP, .NET
  - الحسابات العلمية → Fortran, Matlab
  - الأنظمة المدمجة وأنظمة التشغيل → C
  - تطبيقات المؤسسات (البنوك وغيرها) إلى Java

كلمات هذه الصفحة

satellite — أقمار صناعية

embedded — أنظمة مدمجة
برمجيات داخل أجهزة خاصة

enterprise — مؤسساتي
تطبيقات ضخمة للشركات

ترجمة / تحويل اللغات عالية المستوى إلى لغات منخفضة المستوى

- يجب ترجمة البرنامج المصدري (source program) إلى كود آلة (machine code).
- برنامج تطبيقي يسمى مترجم **(translator)** يقوم بتحويل الكود من لغة عالية المستوى إلى لغة منخفضة المستوى، مما يجعله كودا مفهوما للآلة.
- المترجمات **(Compilers)** والمفسرات (interpreters) هي ببساطة مترجمات لغات في برمجة الحاسوب.
- يتم تشغيل البرنامج عن طريق تحميل كود الآلة (machine code) إلى الذاكرة الرئيسية. يقوم المعالج بتنفيذ تعليمات لغة الآلة هذه مباشرة.

كلمات هذه الصفحة

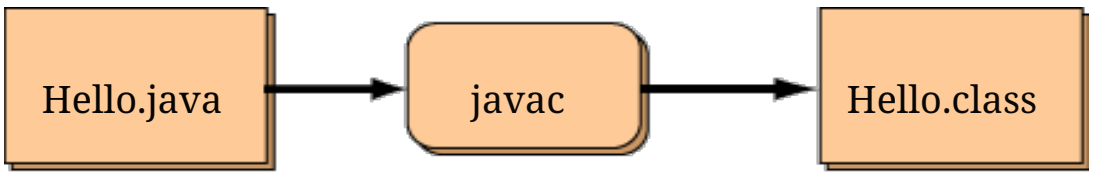
source program — الكود المصدري · الكود الذي يكتبه المبرمج

transforms — يحوّل

translators — مترجمات · برامج تحويل لغات البرمجة

المترجمات

- الـ مترجم **(compiler)** هو برنامج يأخذ برنامجا مكتوبا بلغة عالية المستوى ويقوم بترجمة البرنامج بالكامل إلى كود الآلة دفعة واحدة
- تترجم جميع التعليمات برمجيا قبل أن يقوم الـ CPU بتنفيذ أي منها.
- يقوم مترجم **Java (javac)** بترجمة الكود المصدري لـ **Hello.java** (Java) إلى تمثيل خاص يسمى بايت كود **(bytecode)** (برنامج كائني)
- عند ترجمة برنامج Java إلى بايت كود، تكون ملفات البايت كود متطابقة تماما بغض النظر عن نظام الحاسوب المستخدم.
- تتم عملية التجميع (compilation) مرة واحدة فقط، ويمكن تشغيل البرنامج الكائني (object program) الناتج بالقدر المطلوب من المرات دون الحاجة إلى إعادة تجميع البرنامج المصدري



كلمات هذه الصفحة

entire — كامل · البرنامج ككل

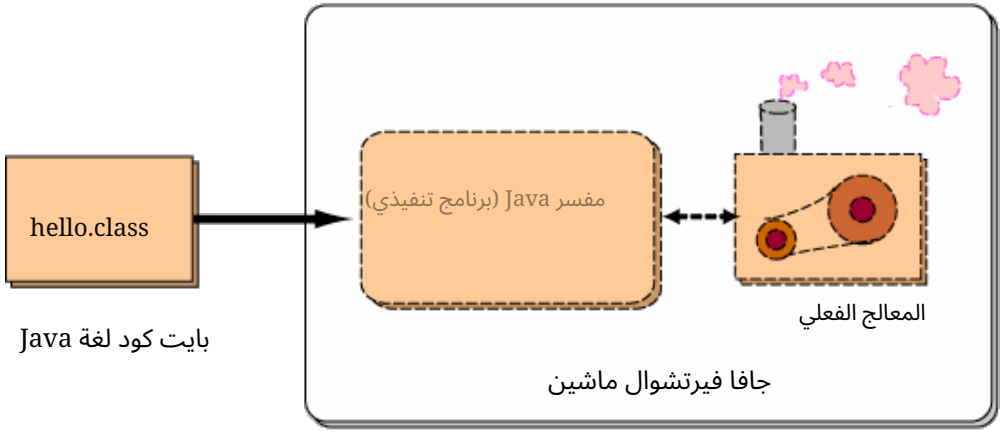
bytecode — بايت كود · كود وسيط خاص بجافا

object program — برنامج الكائن · الكود الناتج بعد الترجمة

re-compiling — إعادة ترجمة · تحويل الكود المصدري مجدداً

- يقوم المفسر **(interpreter)** بقراءة البرنامج المصدري المكتوب بلغة عالية المستوى وتنفيذ التعليمات التي يطلبها البرنامج واحدة تلو الأخرى

- بمجرد ترجمة وتنفيذ تعليمة معينة، تتبعها التعليمة التالية، وهكذا دواليك
- تستخدم Java Virtual Machine (JVM) مفسرا، لـ ترجمة الـ bytecode إلى لغة الآلة وتنفيذها



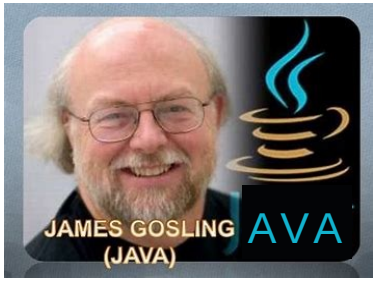
كلمات هذه الصفحة

interpreter — مفسّر · برنامج يترجم وينفذ الكود سطراً بسطر

one by one — سطراً بسطر · تدريجياً

لغة البرمجة Java

تاريخ لغة Java



- **Java** هي لغة برمجة عالية المستوى تعتمد على الكائنات (object-oriented) طورها جيمس غوسلينغ في شركة **Sun Microsystems** في أوائل التسعينيات.
- لم يكن راضيا عن استخدام C++ كلغة برمجة، لذا قام بتطوير Java.
  - كان الاسم القديم للغة Java هو "Oak"
- **Oak**" 1995" تم تغيير اسمها إلى **Java** كأحد المكونات الأساسية لمنصة Java الخاصة بشركة Sun Microsystems
- لغة Java هي لغة توحيد ، صممت لتستوعب نقاط القوة في اللغات السابقة وتتجاهل نقاط ضعفها

كلمات هذه الصفحة

object-oriented — كائنية التوجه · نمط برمجي يعتمد على الكائنات

consolidation — دمج · لغة تدمج أفضل الميزات

absorb — تمتص / تستوعب · أخذ الميزات الجيدة

ميزات Java

- بسيطة
- مستقلة عن المنصة (قابلة للنقل)
  - كائنية التوجه (Object-Oriented)
- قوية وآمنة
- مترجمة ومفسرة
- موزعة
- متعدد الـ thread
- محايد للبنية
- عالي الأداء

كلمات هذه الصفحة

portable — قابل للنقل · يعمل على أي جهاز بسهولة

robust — قوي / متين · يتحمل الأخطاء ولا ينهار بسهولة

architecture-neutral — محايد معمارياً · لا يعتمد على نوع معين من المعالجات

ميزات Java (تتمة)

- البساطة
  - تعتمد قواعد لغة Java على لغة C++ (مع تبسيطها وتحسينها بشكل كبير).
  - قامت Java بـ إزالة العديد من الميزات المعقدة وقليلة الاستخدام، على سبيل المثال، المؤشرات الصريحة (explicit pointers)، وتحميل المعاملات الزائد (operator overloading)، إلخ.
  - لا توجد حاجة لحذف الكائنات غير المشار إليها (unreferenced objects) لوجود تجميع القمامة التلقائي (Automatic Garbage Collection) في Java.
- مستقلة عن المنصة (قابلة للنقل)
  - اكتب مرة واحدة وشغل في أي مكان
  - لغة Java قابلة للنقل بالكامل؛ حيث سيعمل كود Java نفسه بشكل متطابق على منصات مختلفة، بغض النظر عن توافق العتاد أو أنظمة التشغيل.
- كائنية التوجه
  - تعد البرمجة كائنية التوجه (OOP) نهجا برمجيا شائعا يحل محل البرمجة الإجرائية التقليدية، مما يوفر مرونة كبيرة، ونمطية، ووضوحا، وقابلية لإعادة الاستخدام.
  - تقع جميع الأكواد والبيانات ضمن الكلاسات (classes) والأوبجكتات (objects)

كلمات هذه الصفحة

explicit — صريح / مباشر · محدد بوضوح

pointers — مؤشرات · مفهوم برمجي يشير لعناوين الذاكرة

garbage collection — تنظيف الذاكرة التلقائي · حذف البيانات غير المستخدمة

procedural — إجرائي · نمط برمجي يعتمد على الدوال فقط

modularity — نمطية · تقسيم البرنامج لأجزاء مستقلة

reusability — قابلية إعادة الاستخدام

reside — يقطن / يتواجد · يوجد داخل

ميزات Java (تتمة)

- قوية وآمنة
  - معالجة الاستثناءات (exception handling) مدمجة مع التحقق الصارم من الأنواع (type checking)
  - جمع القمامة (garbage collection) تلقائي مما يجعل تلف الذاكرة أو الوصول غير المصرح به إليها أمرا مستحيلا
  - إدارة قوية للذاكرة (لا توجد مؤشرات pointers).
  - تعمل البرامج داخل بيئة معزولة (sandbox) للجهاز الافتراضي
- مترجمة ومفسرة
  - يتم تحويل الكود إلى بايت كود (bytecode) والذي يتم تفسيره بواسطة JVM
- الأنظمة الموزعة (Distributed)
  - إنشاء تطبيقات على الشبكات.
  - يمكن لعدة مبرمجين العمل معا على مشروع واحد من مواقع بعيدة متعددة ومشاركة البيانات والبرامج.

كلمات هذه الصفحة

exception handling — معالجة الاستثناءات · التعامل مع الأخطاء أثناء التشغيل

corruptions — تلف · فساد البيانات في الذاكرة

sandbox — بيئة معزولة · بيئة تنفيذ آمنة ومقيدة

ميزات Java (تتمة)

- متعدد الـ **thread**
  - يسمح بمعالجة مهام متعددة في آن واحد (تشغيل عدة thread)
- محايد معماريا **(Architecture-Neutral)**
  - لا توجد ميزات تعتمد على التنفيذ، على سبيل المثال حجم الأنواع الأولية (primitive types) ثابت.
- أداء عال
  - تستخدم الـ JVM الجديدة تقنية تعرف باسم الترجمة الفورية (JIT) compilation.
  - مع مترجم JIT، يقدم الكود المفسر سرعة تقارب سرعة الكود الأصلي (native code).

كلمات هذه الصفحة

simultaneously — في آن واحد

primitive types — أنواع البيانات الأولية · أنواع أساسية مثل int

native code — كود أصلي · كود يعمل مباشرة على المعالج

تطبيقات Java

- وفقا لشركة **Sun**، تعمل لغة **Java** على **3** مليارات جهاز.
- هناك أربعة أنواع رئيسية من التطبيقات:

1. التطبيقات المستقلة **(Stand alone applications)**

- تعرف أيضا باسم تطبيقات سطح المكتب (desktop applications)
- وهي برمجيات تقليدية، أي أننا نحتاج إلى تثبيتها على كل جهاز.أمثلة: Acrobat reader، وMedia player، ومضاد الفيروسات (Antivirus)، إلخ.

2. تطبيقات الويب **(Web applications)**

- هي التطبيقات التي تعمل على جانب الخادم (server side) وتنشئ صفحات ديناميكية.
- تستخدم تقنيات مثل JSP و Servlet و spring وغيرها لإنشاء تطبيقات الويب باستخدام لغة Java.مثال: google.com و uqu.edu.sa (نظام معلومات الطالب (SIS))

3. تطبيقات المؤسسات

- تطبيق ذو طبيعة موزعة، مثل التطبيقات المصرفية.
- تتمتع بمزايا الأمان العالي، وموازنة الأحمال، والتجميع (clustering).

4. تطبيقات الهواتف المحمولة

- تطبيق تم إنشاؤه من أجل الأجهزة المحمولة
- حاليا، Android و Java ME يستخدمان لإنشاء تطبيقات الهواتف المحمولة.
- تطبيقات أخرى مثل الروبوتات، والألعاب، وما إلى ذلك.

https://www.slideshare.net/MadishettyPrathibha/introduction-java-237623974?from_search=16

كلمات هذه الصفحة

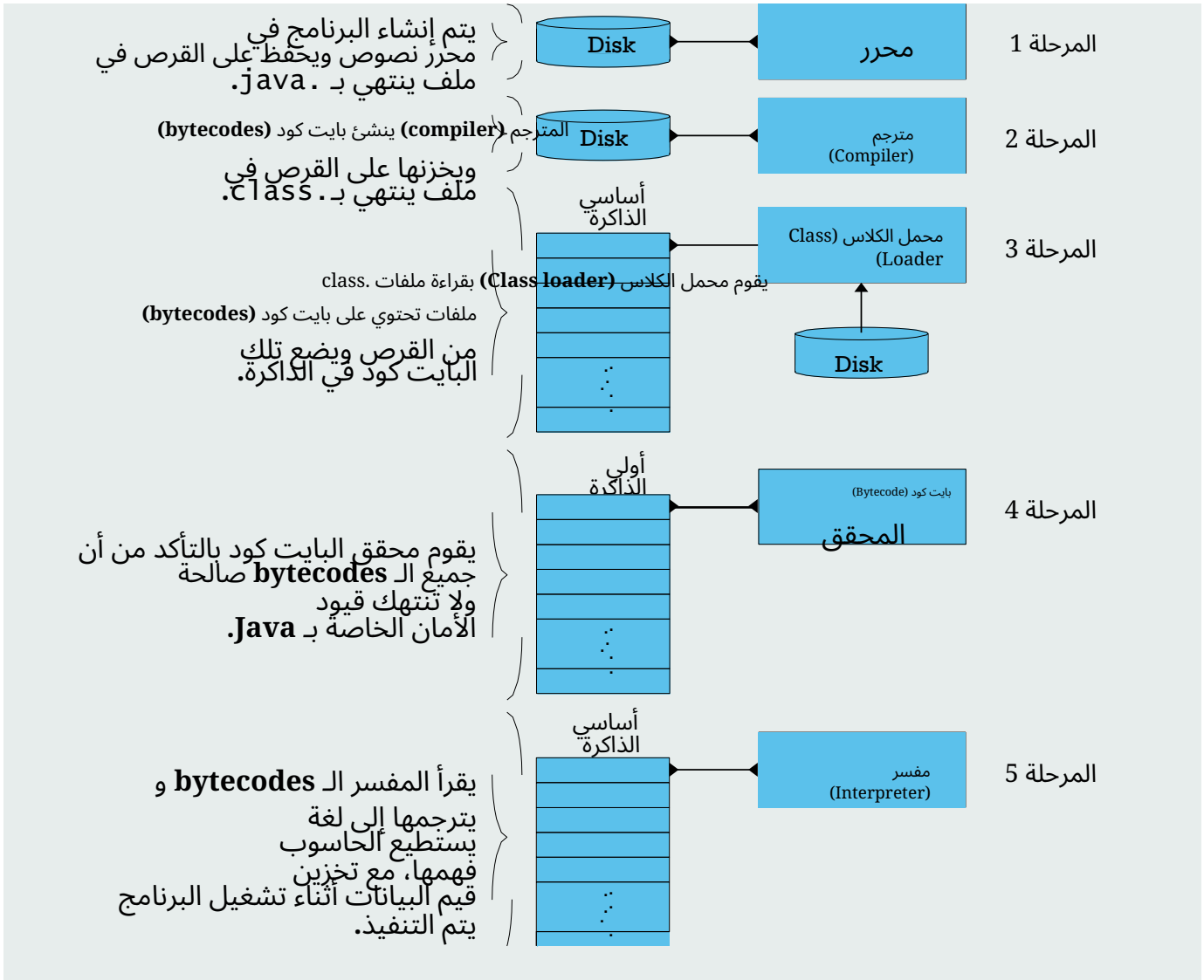
devices — أجهزة

stand alone — مستقل · يعمل بمفرده دون إنترنت

dynamic — ديناميكي · متغير ومستجيب

بيئة تطوير Java

- تمر برامج Java عادة عبر خمس مراحل



كلمات هذه الصفحة

phases — مراحل

editor — محرر نصوص · برنامج لكتابة الكود

verifier — مُحقق · برنامج يتأكد من صحة الكود

الخطوة 1: إنشاء وتعديل البرامج

- تتضمن هذه الخطوة استخدام محرر **(editor)** لكتابة برنامج بلغة Java (الكود المصدري)، ثم حفظه على جهاز تخزين ثانوي، مثل القرص الصلب.
  - محررات معروفة: برنامج Notepad على نظام Windows وبرنامج TextEdit على نظام macOS

- بيئات التطوير المتكاملة **(IDEs)** توفر أدوات تدعم عملية تطوير البرمجيات، مثل المحررات، وأدوات تصحيح الأخطاء (debuggers) لتحديد الأخطاء المنطقية (الأخطاء التي تتسبب في تنفيذ البرامج بشكل غير صحيح) وغيرها.
  - بيئات تطوير **Java** الشهيرة:
    - Eclipse (www.eclipse.org)
    - NetBeans (www.netbeans.org)
    - IntelliJ IDEA (www.jetbrains.com)
    - VSCode (code.visualstudio.com -- ليس للجافا فقط!)

- تنتهي جميع ملفات الكود المصدري بلغة **Java** بالامتداد **.java**.

كلمات هذه الصفحة

secondary storage — تخزين ثانوي · مثل القرص الصلب

debuggers — أدوات تنقيح · أدوات لإصلاح أخطاء الكود

extension — امتداد · لاحقة الملف مثل .java

الخطوة 2: تجميع برنامج Java إلى بايت كود (byte-code)

- لا يقوم مترجم ***(compiler)*** لغة **Java** بترجمة برنامج Java إلى لغة الآلة لحاسوب معين.

- بدلا من ذلك، فإنه يقوم بترجمة برنامج **Java** إلى بايت كود.
  - البايت كود هو لغة الآلة الخاصة بحاسوب افتراضي (أو مفسر - ***interpreter***) يسمى **Java Virtual Machine**.
  - ملف البايت كود يأتي بامتداد .class
  - كود البايت **(byte-code)** يمثل المهام التي سيتم تنفيذها في مرحلة التنفيذ

كلمات هذه الصفحة

hypothetical — افتراضي

الخطوة 2: تجميع برنامج Java إلى Bytecodes - لماذا الـ Bytecode؟

- أكواد البايت **(bytecodes)** يتم تنفيذها بواسطة جافا فيرتشوال ماشين **(JVM)**— وهي جزء من الـ JDK وأساس منصة جافا

- الـ **Bytecodes** مستقلة عن المنصة
  - فهي لا تعتمد على منصة عتاد (hardware) معينة

- الـ **Bytecodes** قابلة للنقل
  - يمكن تنفيذ نفس الـ bytecodes على أي منصة تحتوي على JVM تفهم إصدار Java الذي تم تجميع (compile) الـ bytecodes باستخدامه

- كود البايت **(Bytecode)** يمكن إرساله عبر الإنترنت واستخدامه في أي مكان في العالم. وهذا يجعل لغة **Java** مناسبة لتطبيقات الإنترنت.

الخطوة 3: تحميل البرنامج إلى الذاكرة

- تقوم آلة جافا الافتراضية **(JVM)** بوضع البرنامج في الذاكرة لتنفيذه، وتعرف هذه العملية باسم التحميل (loading)

- يقوم محمل الكلاسات **(class loader)** في الـ **JVM** (والذي يسمى أيضا الرابط ***(linker)***) بأخذ ملفات **class.** التي تحتوي على أكواد البايت (byte-codes) الخاصة بالبرنامج ونقلها إلى الذاكرة الرئيسية.

- يقوم محمل الكلاسات ***(class loader)*** تلقائيا بربط الكلاسات ببعضها البعض.

كلمات هذه الصفحة

linker — رابط · برنامج لربط أجزاء الكود

transfers — ينقل

الخطوة 4: التحقق من الـ Bytecode

- أثناء تحميل الكلاسات، يقوم محقق البايت كود (bytecode verifier) بفحص البايت كود الخاص بها لضمان أنها سليمة ولا تنتهك قيود الأمان الخاصة بـ Java

- تفرض لغة Java إجراءات أمنية صارمة لضمان أن برامج Java التي تصل عبر الشبكة لا تلحق الضرر بملفاتك أو بنظامك (كما قد تفعل الفيروسات والديدان الحاسوبية)

كلمات هذه الصفحة

enforces — يفرض · تطبيق القوانين بصرامة

worms — ديدان حاسوبية · نوع من الفيروسات

الخطوة 5: التفسير والتنفيذ

• تقوم بيئة تشغيل **Java**، المعروفة بـ **(Java Virtual Machine (JVM**،بترجمة كل تعليمة من نوع **byte-code** وتنفيذ التعليمات الناتجة بلغة الآلة **(machine-language)** قبل الانتقال لترجمة التعليمة التالية من نوع **byte-code**.

• تنفذ معظم برامج Java اليوم باستخدام مترجم ***(Just-In-Time (JIT*** . تتفاعل مترجمات **JIT** مع بيئة تشغيل **(Java (JVM** أثناء وقت التشغيل وتحول تسلسلات الـ bytecode المناسبة إلى كود آلة (native machine code) يمكن إرساله مباشرة إلى معالج الحاسوب (CPU).

كلمات هذه الصفحة

sequences — سلاسل / تتابعات

مراجعة!

- عادة ما يتم تجميع كود Java إلى bytecode مستقل عن المنصة (ملفات class).
- تستطيع الـ JVM تحميل ملفات الـ class وتنفيذ الـ bytecode الخاص بـ Java عبر مفسر Java.
- على الرغم من أن هذا الـ bytecode يفسر عادة، إلا أنه قد يترجم أيضا إلى كود آلة (native machine code) باستخدام تقنية الـ JIT (Just-In-Time) في الـ JVM.
- على عكس المترجم العادي، يقوم مترجم الـ JIT بترجمة الكود (bytecode) فقط عند الحاجة إليه.
- لذا، توفر الـ JVM طريقة مستقلة عن المنصة لتنفيذ الـ bytecode الخاص بـ Java. وهي حجر الأساس لمبدأ "اكتب مرة واحدة، وشغل في أي مكان" (write-once run anywhere) الذي تتميز به برامج Java.

https://www.slideshare.net/wso2.org/java-performance-and-profiling

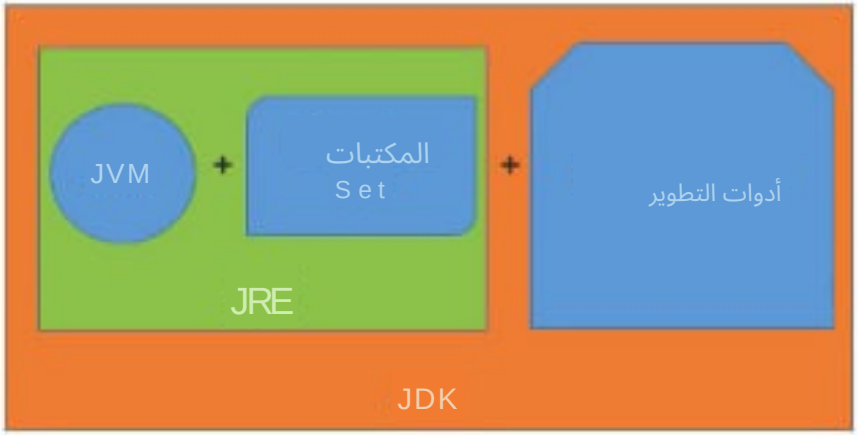
كلمات هذه الصفحة

keystone — حجر الأساس

المفاهيم الأساسية في Java

جافا فيرتشوال ماشين

بيئة تشغيل Java



طقم تطوير جافا

~~المصدر~~

بيئة تشغيل جافا (JRE)

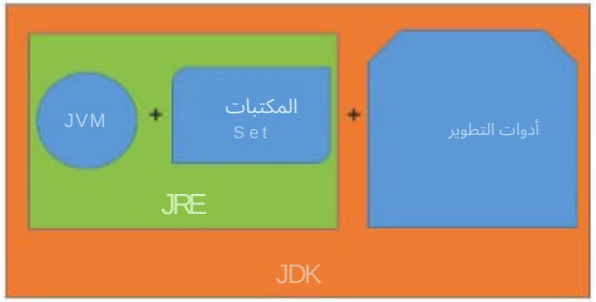
- تتضمن:
  - جافا فيرتشوال ماشين (JVM) مع مترجم JIT
  - مكتبات كلاسات جافا

بيئة تشغيل جافا

- تستخدم لـ:
  - قراءة البايت كود (bytecode)
  - تشغيل نفس الـ bytecode على أي جهاز يحتوي على JVM (تشغيل أي برنامج Java)

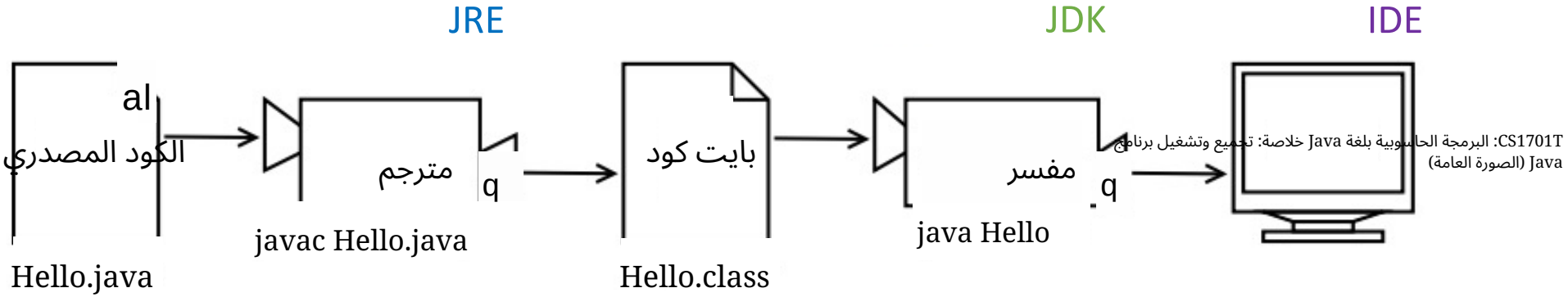
طقم تطوير جافا (JDK)

- يتضمن ما يلي:
  - بيئة تشغيل جافا (JRE)
  - مترجم جافا (Java Compiler)
  - أدوات أخرى (تتضمن مكتبات جافا، ومترجمات كود جافا المصدري، ومصححات أخطاء جافا، وأدوات التجميع والنشر)
- يستخدم ببساطة لترجمة الـ bytecode (من .ra



طقم تطوير Java

مراجعة: تجميع وتشغيل برنامج Java (نظرة عامة)



هل لديك أسئلة؟