CPE3243 วิศวกรรมซอฟต์แวร์
(Software Engineering)

Piyavit Laung-Aram

Major of Computer Engineering

Faculty of Engineering

Ramkhamhaeng University, Thailand

การเขียนโปรแกรมด้วยภาษา Java

(Java Programming)

Java คืออะไร?

Java ได้รับการพัฒนาโดย Sun Microsystems (ซึ่งปัจจุบันเป็นบริษัทย่อยของ Oracle) ในปี 1995 James Gosling เป็นที่รู้จักในนามบิดาแห่ง Java ก่อน Java ชื่อของมันคือ โอ๊ค เนื่องจาก Oak เป็นบริษัทจดทะเบียนอยู่แล้ว ดังนั้น James Gosling และทีมของเขาจึงเปลี่ยนชื่อจาก Oak เป็น Java

แหล่งที่มา https://javacodeplayground.wordpress.com/

Java คืออะไร?

- Java เป็นภาษาโปรแกรมและแพลตฟอร์ม
- Java เป็นภาษาโปรแกรมระดับสูง มีความแข็งแกร่ง เป็นภาษา OOP และมีความปลอดภัยสูง

แพลตฟอร์ม คือ สภาพแวดล้อมของฮาร์ดแวร์หรือซอฟต์แวร์ใดๆ ที่โปรแกรมทำงานหรือรันอยู่

เนื่องจาก Java มีสภาพแวดล้อมในการทำงาน (JRE-Java Runtime Environment) และ API จึงเรียก Java ว่าแพลตฟอร์ม

ตัวอย่าง Java

มาดูตัวอย่างการเขียนโปรแกรม Java กัน คำอธิบายโดยละเอียดของตัวอย่าง Hello Java มีอยู่ใน slide ถัดไป (Java Online Compiler https://www.onlinegdb.com/)

```
1. //Main.java
2. class Main
3. {
4.     public static void main(String args[])
5.     {
6.      System.out.println("Hello Java");
7.     }
8. }
```

แอปพลิเคชันของ Java

จากข้อมูลของ Sun อุปกรณ์ 3 พันล้านเครื่องใช้งาน Java มีอุปกรณ์จำนวนมากที่ใช้ Java อยู่ในปัจจุบัน บางส่วนมีดังนี้:

- แอปพลิเคชันเดสก์ท็อป เช่น โปรแกรมอ่าน Acrobat โปรแกรมเล่นสื่อ โปรแกรมป้องกันไวรัส ฯลฯ
- เว็บแอปพลิเคชัน เช่น irctc.co.in, เป็นต้น
- แอปพลิเคชันระดับองค์กร เช่น แอปพลิเคชันธนาคาร
- มือถือ
- ระบบสมองกลฝังตัว
- สมาร์ทการ์ด
- วิทยาการหุ่นยนต์
- เกมส์ ฯลฯ.

ประเภทของแอปพลิเคชัน *Java*

แอปพลิเคชันส่วนใหญ่มี 4 ประเภท

1) แอปพลิเคชันแบบสแตนด์อโลน (Stand Alone)

แอปพลิเคชันแบบสแตนด์อโลนเรียกอีกอย่างว่าแอปพลิเคชันเดสก์ท็อปหรือแอปพลิเคชันแบบหน้าต่าง เหล่านี้เป็นซอฟต์แวร์ดั้งเดิมที่เราจำเป็นต้องติดตั้งในทุกเครื่อง ตัวอย่างของแอปพลิเคชันแบบสแตนด์อโลน ได้แก่ เครื่องเล่นสื่อ โปรแกรมป้องกันไวรัส ฯลฯ AWT และ Swing ใช้ใน Java สำหรับการสร้างแอปพลิเคชันแบบสแตนด์อโลน

2) เว็บแอปพลิเคชัน (Web Application)

แอปพลิเคชันที่ทำงานบนฝั่งเซิร์ฟเวอร์และสร้างเพจแบบไดนามิกเรียกว่าเว็บแอปพลิเคชัน ปัจจุบันมีการใช้เทคโนโลยี Servlet, JSP, Struts, Spring, Hibernate, JSF และอื่น ๆ เพื่อสร้างเว็บแอปพลิเคชันใน Java

ประเภทของแอปพลิเคชัน Java (ต่อ)

3) แอปพลิเคชันระดับองค์กร (Enterprise)

แอปพลิเคชันที่เผยแพร่ในลักษณะเช่นแอปพลิเคชันธนาคาร ฯลฯ เรียกว่าแอปพลิเคชันระดับองค์กร มีข้อดีเช่นการรักษาความปลอดภัยระดับสูง การทำโหลดบาลานซ์ และการจัดกลุ่ม ใน Java EJB ใช้สำหรับสร้างแอปพลิเคชันระดับองค์กร

4) แอปพลิเคชั่นสำหรับมือถือ (Mobile)

แอปพลิเคชันที่สร้างขึ้นสำหรับอุปกรณ์มือถือเรียกว่าแอปพลิเคชันมือถือ ปัจจุบันใช้ Android และ Java ME เพื่อสร้างแอปพลิเคชันมือถือ

แพลตฟอร์ม Java / รุ่นต่างๆ

Java มี 4 แพลตฟอร์มหรือรุ่น

1) Java SE (Java Standard Edition)

เป็นแพลตฟอร์มการเขียนโปรแกรม Java ประกอบด้วย Java Programming API เช่น java.lang, java.io, java.net, java.util, java.sql, java.math เป็นต้น รวมถึงหัวข้อหลักเช่น OOPs, String, Regex, Exception, Inner class, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection ฯลฯ

2) Java EE (Java Enterprise Edition)

เป็นแพลตฟอร์มระดับองค์กรที่ใช้เป็นหลักในการพัฒนาเว็บและแอปพลิเคชันระดับองค์กร มันถูกสร้างขึ้นบนแพลตฟอร์ม Java SE ประกอบด้วยหัวข้อเช่น Servlet, JSP, Web Services, EJB, JPA เป็นต้น

แพลตฟอร์ม Java / รุ่นต่างๆ (ต่อ)

3) Java ME (Java Micro Edition)

เป็นแพลตฟอร์มไมโครที่ทุ่มเทให้กับแอปพลิเคชันมือถือ

4) JavaFX

ใช้ในการพัฒนาอินเทอร์เฟซผู้ใช้ขนาดย่อมสำหรับแอปพลิเคชันอินเทอร์เน็ตที่มีฟังก์ชันครบครัน มันถูกใช้เพื่อพัฒนาอินเทอร์เน็ตแอปพลิเคชั่นที่หลากหลาย โดยใช้ API ส่วนต่อประสานผู้ใช้ที่มีขนาดเล็ก ใช้ในการสร้างแอปพลิเคชั่นให้แอปพลิเคชั่นปกติที่เราเขียนโค้ดด้วยภาษา Java ปกติ สามารถทำงานบนเครือข่ายอินเทอร์เน็ตได้ ตอนนี้ JavaFX เป็นโอเพ่นซอร์ส แพลตฟอร์มแอปพลิเคชันไคลเอนต์รุ่นใหม่สำหรับเดสก์ท็อป มือถือ และระบบฝังตัวที่สร้างขึ้นบน Java เป็นความพยายามร่วมกันของบุคคลและบริษัทจำนวนมาก โดยมีเป้าหมายในการผลิตชุดเครื่องมือที่ทันสมัย มีประสิทธิภาพ และมีคุณสมบัติครบถ้วนสำหรับการพัฒนาไคลเอนต์แอปพลิเคชัน-https://openjfx.io/

ประวัติของ *Java*

ประวัติของ Java นั้นน่าสนใจมาก Java ได้รับการออกแบบมาสำหรับโทรทัศน์แบบอินเทอร์แอคทีฟ แต่เดิม Java เป็นเทคโนโลยีที่ล้ำหน้าเกินไปสำหรับอุตสาหกรรมเคเบิลทีวีแบบดิจิทัลในขณะนั้น ประวัติของ Java เริ่มต้นด้วย Green Team สมาชิกในทีม Java (หรือที่รู้จักในชื่อ Green Team) ได้ริเริ่มโครงการนี้เพื่อพัฒนาภาษาสำหรับอุปกรณ์ดิจิทัล เช่น กล่องรับสัญญาณ โทรทัศน์ ฯลฯ อย่างไรก็ตาม เหมาะที่สุดสำหรับการเขียนโปรแกรมทางอินเทอร์เน็ต ต่อมา Netscape ได้รวมเทคโนโลยี Java เข้าไปในโปรแกรม

หลักการสำหรับการสร้างการเขียนโปรแกรม Java คือ "ง่าย แข็งแกร่ง พกพา ไม่ขึ้นกับแพลตฟอร์ม ปลอดภัย ประสิทธิภาพสูง มัลติเธรด สถาปัตยกรรมเป็นกลาง เน้นวัตถุ ตีความ และไดนามิก" Java ได้รับการพัฒนาโดย James Gosling ซึ่งเป็นที่รู้จักในนามบิดาของ Java ในปี 1995 James Gosling และสมาชิกในทีมของเขาเริ่มโครงการในช่วงต้นทศวรรษ 90

ประวัติของ Java (ต่อ)

เจมส์ กอสลิ่ง - ผู้ก่อตั้ง java

ปัจจุบัน Java ถูกใช้ในการเขียนโปรแกรมทางอินเทอร์เน็ต อุปกรณ์พกพา เกม โซลูชันธุรกิจอิเล็กทรอนิกส์ ฯลฯ ต่อไปนี้คือจุดสำคัญที่อธิบายประวัติของ Java

1) James Gosling, Mike Sheridan และ Patrick Naughton ริเริ่มโครงการภาษา Java ในเดือนมิถุนายน 1991 ทีมวิศวกร ของ Sun กลุ่มเล็กๆ ชื่อ Green Team

2) ในขั้นต้น มันถูกออกแบบมาสำหรับระบบฝังตัวขนาดเล็กในเครื่องใช้ไฟฟ้าเช่น set-top box

3) ประการแรก มันถูกเรียกว่า "Greentalk" โดย James Gosling และนามสกุลไฟล์คือ .gt

4) หลังจากนั้นเรียกว่าโอ๊คและได้รับการพัฒนาเป็นส่วนหนึ่งของ Green Project

ประวัติของ Java (ต่อ)

ทำไม Java ถูกตั้งชื่อว่า "Oak"?

5) ทำไมต้องโอ๊ค? ต้นโอ๊กเป็นสัญลักษณ์ของความแข็งแกร่งและได้รับเลือกให้เป็นต้นไม้ประจำชาติของหลายประเทศ เช่น สหรัฐอเมริกา ฝรั่งเศส เยอรมนี โรมาเนีย เป็นต้น

6) ในปี 1995 Oak ถูกเปลี่ยนชื่อเป็น "Java" เพราะเป็นเครื่องหมายการค้าของ Oak Technologies แล้ว

ประวัติของ Java (ต่อ)

เหตุใด Java Programming จึงมีชื่อว่า "Java"

7) ทำไมพวกเขาถึงเลือกชื่อ Java สำหรับภาษา Java? ทีมงานรวมตัวกันเพื่อเลือกชื่อใหม่ คำที่แนะนำคือ " dynamic-ไดนามิก" " revolutionary-ปฏิวัติ" " Silk-ไหม" " jolt - กระตุก" "DNA-ดีเอ็นเอ" ฯลฯ พวกเขาต้องการบางสิ่งที่สะท้อนถึงแก่นแท้ของเทคโนโลยี: ปฏิวัติ ไดนามิก มีชีวิตชีวา เท่ ไม่ซ้ำใคร และง่ายต่อการ สะกดและสนุกกับการพูด

ตามที่ James Gosling กล่าวไว้ "Java เป็นหนึ่งในตัวเลือกอันดับต้น ๆ ร่วมกับ Silk" เนื่องจาก Java มีเอกลักษณ์เฉพาะตัว สมาชิกในทีมส่วนใหญ่จึงชอบ Java มากกว่าชื่ออื่นๆ

ประวัติของ Java (ต่อ)

8) ชวาเป็นเกาะในประเทศอินโดนีเซียที่ผลิตกาแฟเป็นครั้งแรก (เรียกว่ากาแฟชวา) เป็นเมล็ดกาแฟเอสเปรสโซชนิดหนึ่ง เจมส์ กอสลิงเลือกชื่อชวาขณะดื่มกาแฟใกล้ๆ ที่ทำงานของเขา

9) สังเกตว่า Java เป็นเพียงชื่อ ไม่ใช่ตัวย่อ

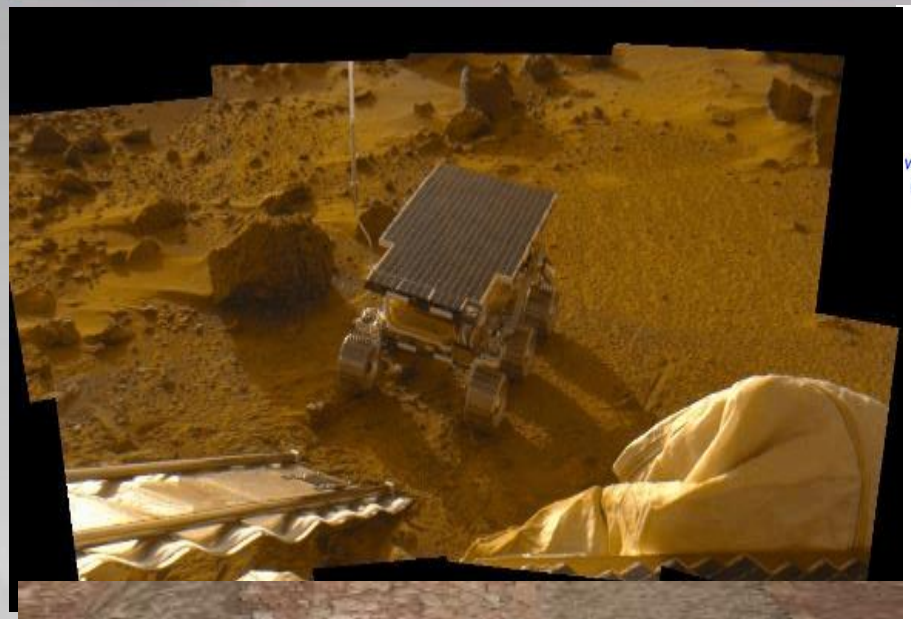
10) เริ่มแรกพัฒนาโดย James Gosling ที่ Sun Microsystems (ซึ่งปัจจุบันเป็นบริษัทย่อยของ Oracle Corporation) และเปิดตัวในปี 1995

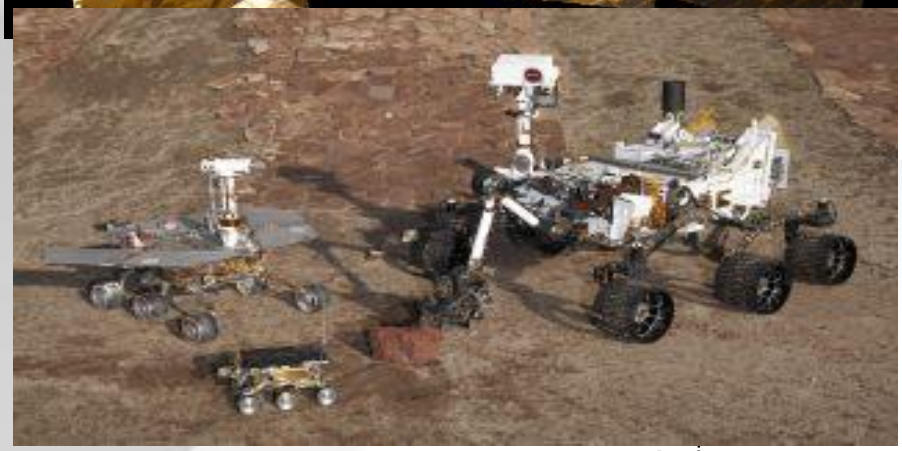
11) ในปี 1995 นิตยสาร Time ยกให้ Java เป็นหนึ่งในสิบผลิตภัณฑ์ที่ดีที่สุดของปี 1995

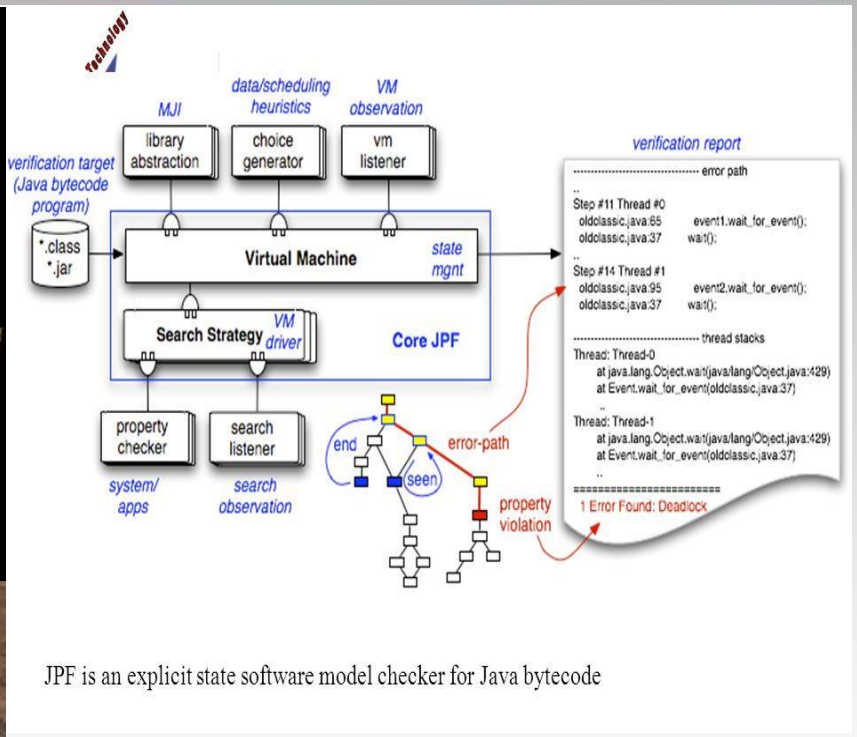
ประวัติของ Java (ต่อ)

12) JDK 1.0 เปิดตัวเมื่อวันที่ 23 มกราคม พ.ศ. 2539 หลังจาก Java รุ่นแรกมีคุณลักษณะเพิ่มเติมมากมายที่เพิ่มเข้ามาในภาษา ขณะนี้มีการใช้ Java ในแอปพลิเคชัน Windows, เว็บแอปพลิเคชัน, แอปพลิเคชันระดับองค์กร, แอปพลิเคชันมือถือ, การ์ด ฯลฯ เวอร์ชันใหม่แต่ละเวอร์ชันเพิ่มคุณลักษณะใหม่ใน Java

ประวัติของ Java (ต่อ)







JPF is an explicit state software model checker for Java bytecode

แหล่งที่มา https://slideplayer.com/ , https://www.space.com/ https://pvs-studio.com/

ประวัติของ Java (ต่อ)

แหล่งที่มา https://www.youtube.com/watch?v=6t3IARmIdOI

ประวัติเวอร์ชัน Java

Java หลายรุ่นได้รับการเผยแพร่จนถึงขณะนี้ Java รุ่นเสถียรในปัจจุบันคือ Java SE 10

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
7. J2SE 5.0 (30th Sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 (18th Mar 2014)
11. Java SE 9 (21st Sep 2017)
12. Java SE 10 (20th Mar 2018)
13. Java SE 11 (September 2018)
14. Java SE 12 (March 2019)
15. Java SE 13 (September 2019)
16. Java SE 14 (Mar 2020)
17. Java SE 15 (September 2020)
18. Java SE 16 (Mar 2021)
19. Java SE 17 (September 2021)
20. Java SE 18 (March 2022)
21. Java SE 19 (September 2022)
22. Java SE 20 (March 2023)
23. Java SE 21 (September 2023)
24. Java SE 22 (March 2024)
25. Java SE 23 ((September 2024)

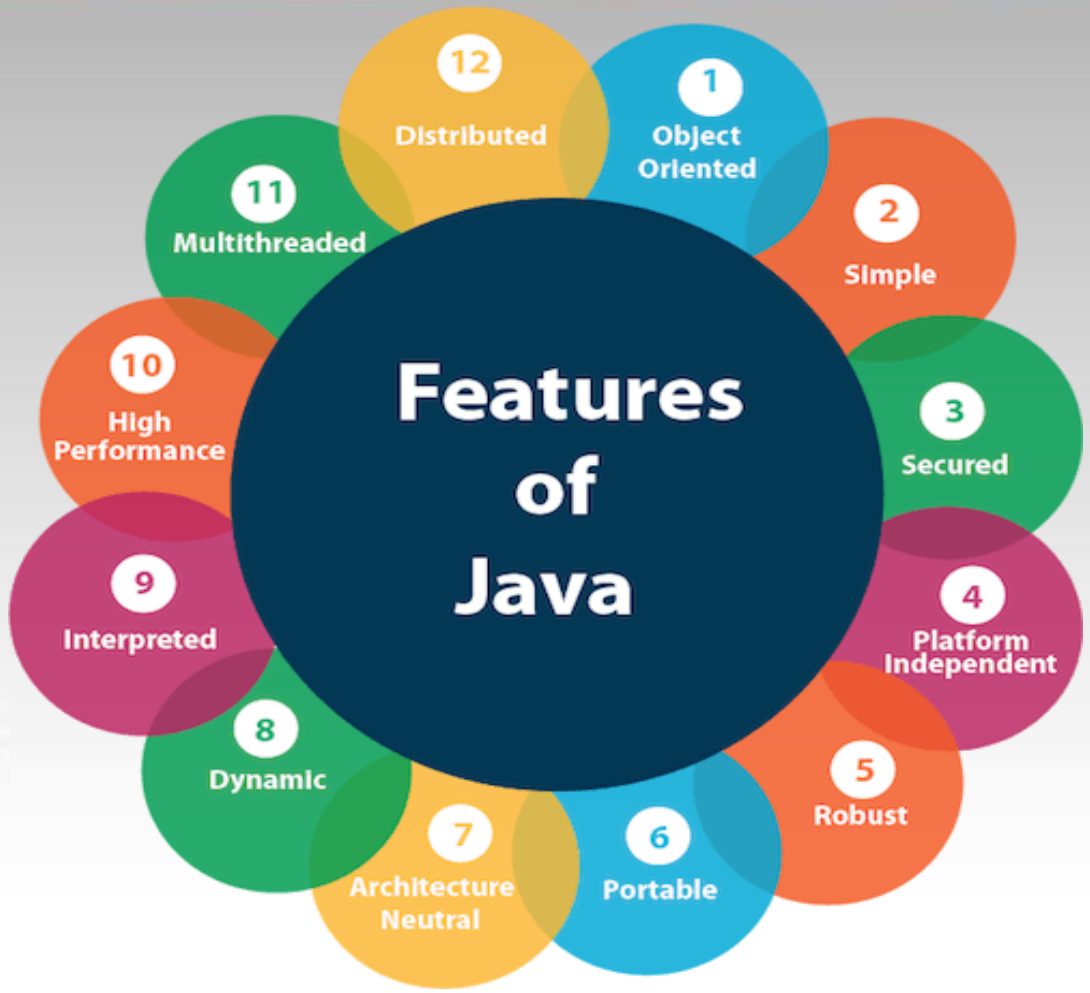
นับตั้งแต่เปิดตัว Java SE 8 -บริษัท Oracle จะดำเนินตามรูปแบบที่เวอร์ชันคู่ออกในเดือนมีนาคมและเวอร์ชันคี่ที่ออกในเดือนกันยายน

คุณสมบัติของ Java

วัตถุประสงค์หลักของการสร้างภาษาโปรแกรม Java คือทำให้ภาษาโปรแกรมแบบพกพาใช้งานง่ายและปลอดภัย นอกเหนือจากนี้ ยังมีคุณสมบัติที่ยอดเยี่ยมบางอย่างที่มีบทบาทสำคัญในความนิยมของภาษานี้ คุณสมบัติของ Java เรียกอีกอย่างว่า คำศัพท์ Java (Java buzzwords)

รายการคุณสมบัติที่สำคัญที่สุดของภาษา Java แสดงไว้ด้านล่าง

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic



แหล่งที่มา https://www.javatpoint.com/

Simple-*เรียบง่าย*

Java นั้นเรียนรู้ได้ง่ายมาก และไวยากรณ์ของมันนั้นเรียบง่าย สะอาดตา และเข้าใจง่าย ตามที่ Sun Microsystem ออกแบบไว้ ภาษา Java เป็นภาษาการเขียนโปรแกรมอย่างง่าย เนื่องจาก :

- ไวยากรณ์ Java ขึ้นอยู่กับ C ++ (เพื่อให้โปรแกรมเมอร์เรียนรู้ได้ง่ายขึ้นหลังจาก C ++)
- Java ได้ลบคุณลักษณะที่ซับซ้อนและไม่ค่อยได้ใช้ออกไปมากมาย เช่น พอยน์เตอร์ที่ชัดเจน การโอเวอร์โหลดของโอเปอเรเตอร์ ฯลฯ
- ไม่จำเป็นต้องลบอ็อบเจ็กต์ที่ไม่ได้อ้างอิงเพราะมี Automatic Garbage Collection ใน Java

Object-oriented

Java เป็นภาษา OOP ทุกอย่างใน Java เป็นอ็อบเจ็กต์ Object-Oriented หมายถึงเราจัดระเบียบซอฟต์แวร์ของเราเป็นการรวมกันของอ็อบเจ็กต์ประเภทต่างๆ ที่รวมทั้งข้อมูลและพฤติกรรม การเขียนโปรแกรม Object Oriented Programming (OOP) เป็นวิธีการที่ลดความยุ่งยากในการพัฒนาและบำรุงรักษาซอฟต์แวร์โดยให้กฎเกณฑ์บางประการ

แนวคิดพื้นฐานของ OOP คือ:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Platform Independent -อิสระจากแพลตฟอร์ม

Java เป็นแพลตฟอร์มที่ไม่ขึ้นกับแพลตฟอร์มเพราะแตกต่างจากภาษาอื่นๆ เช่น C, C++ เป็นต้น ซึ่งรวบรวมไว้ในเครื่องเฉพาะของแพลตฟอร์ม ในขณะที่ Java เป็นการเขียนเพียงครั้งเดียว รันได้ทุกที่ แพลตฟอร์มคือสภาพแวดล้อมของฮาร์ดแวร์หรือซอฟต์แวร์ที่โปรแกรมทำงาน

แพลตฟอร์มมีสองประเภทคือ

- ซอฟต์แวร์แพลตฟอร์ม
- ฮาร์ดแวร์แพลตฟอร์ม

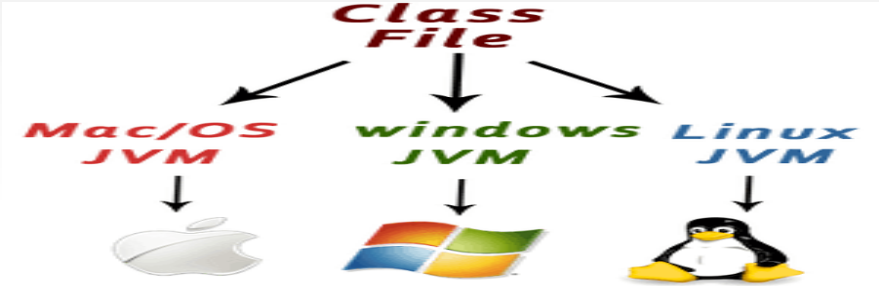
Java จัดเป็นซอฟต์แวร์แพลตฟอร์ม

Platform Independent –อิสระจากแพลตฟอร์ม(ต่อ)

แพลตฟอร์ม Java แตกต่างจากแพลตฟอร์มอื่น ๆ ส่วนใหญ่ในแง่ที่ว่าเป็นแพลตฟอร์มที่ใช้ซอฟต์แวร์ที่ทำงานบนแพลตฟอร์มที่ใช้ฮาร์ดแวร์อื่น ๆ มันมีสององค์ประกอบ:

1. สภาพแวดล้อมรันไทม์
2. API (อินเทอร์เฟซการเขียนโปรแกรมแอปพลิเคชัน)

โค้ด Java สามารถรันได้บนหลายแพลตฟอร์ม เช่น Windows, Linux, Sun Solaris, Mac/OS เป็นต้น คอมไพเลอร์โค้ด Java คอมไพล์แล้วแปลงเป็น bytecode bytecode นี้เป็นรหัสที่ไม่ขึ้นกับแพลตฟอร์ม เพราะสามารถทำงานบนหลายแพลตฟอร์ม เช่น เขียนครั้งเดียวและเรียกใช้ได้ทุกที่ (WORA)

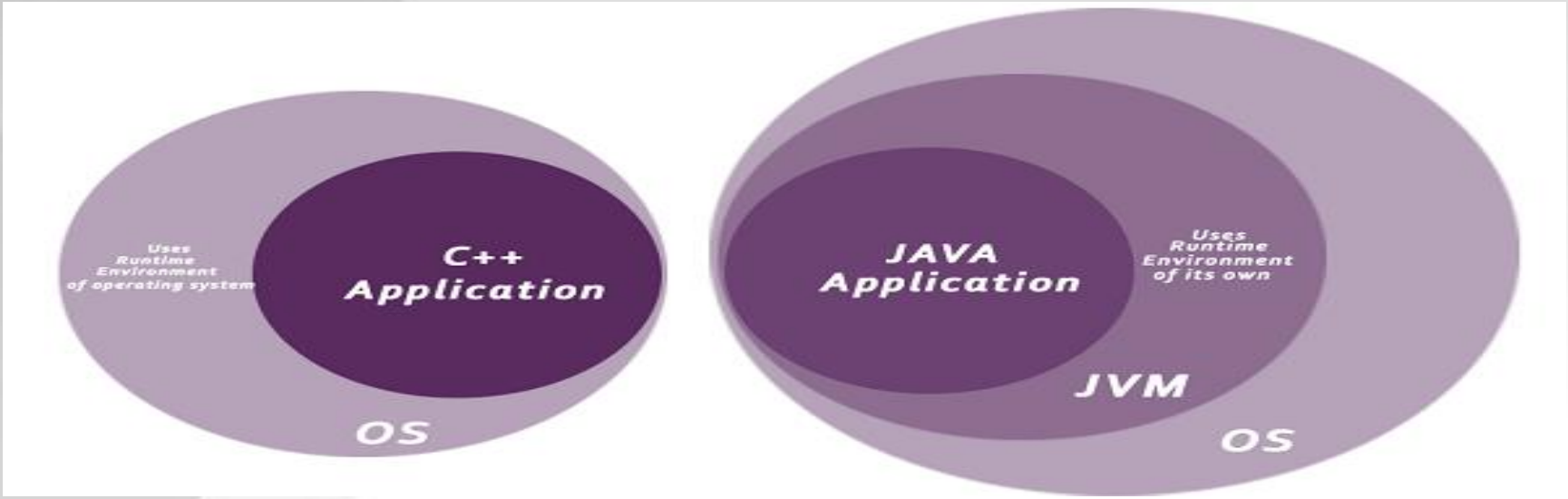


แหล่งที่มา https://www.javatpoint.com/

Secured

Java เป็นที่รู้จักกันดีในเรื่องความปลอดภัย ด้วย Java เราสามารถพัฒนาระบบที่ปราศจากไวรัส Java มีความปลอดภัยเนื่องจาก:

- ไม่มีตัวชี้ที่ชัดเจน-No explicit pointer
- โปรแกรม Java ทำงานภายใน virtual machine sandbox



แหล่งที่มา https://www.javatpoint.com/

Secured (ต่อ)

Classloader: Classloader ใน Java เป็นส่วนหนึ่งของ Java Runtime Environment (JRE) ซึ่งใช้ในการโหลดคลาส Java ลงใน Java Virtual Machine แบบไดนามิก เพิ่มความปลอดภัยโดยแยกแพ็กเกจสำหรับคลาสของระบบไฟล์ในเครื่อง ออกจากที่นำเข้าจากแหล่งที่มาของเครือข่าย

Bytecode Verifier: ตรวจสอบส่วนย่อยของโค้ดสำหรับโค้ดที่ผิดกฎหมายที่สามารถละเมิดสิทธิ์การเข้าถึงออปเจ็ก

Security Manager: กำหนดทรัพยากรที่คลาสสามารถเข้าถึงได้ เช่น การอ่านและการเขียนไปยังดิสก์ในเครื่อง

ภาษา Java เตรียมความปลอดภัยให้เป็นค่าเริ่มต้นในเบื้องต้น ความปลอดภัยบางอย่างนักพัฒนาแอปพลิเคชันสามารถกำหนดได้ ผ่าน SSL, JAAS, Cryptography ฯลฯ

Robust

Robust นั้นแข็งแกร่ง Java เป็น Robust เพราะ:

- ใช้การจัดการหน่วยความจำที่แข็งแกร่ง
- ไม่มีพอยน์เตอร์ ทำให้ช่วยหลีกเลี่ยงปัญหาด้านความปลอดภัย
- Java จัดเตรียมการรวบรวมขยะอัตโนมัติ(garbage collection) ซึ่งทำงานบน Java Virtual Machine เพื่อกำจัดอ็อบเจ็กต์ที่ไม่ได้ใช้โดยแอปพลิเคชัน Java อีกต่อไป
- มีการจัดการข้อยกเว้นและกลไกการตรวจสอบประเภทใน Java จุดเหล่านี้ทำให้ Java แข็งแกร่ง

Architecture-neutral-สถาปัตยกรรม*เป็นกลาง*

Java เป็นสถาปัตยกรรมที่เป็นกลางเนื่องจากไม่มีคุณลักษณะที่ขึ้นกับการใช้งาน เช่น ขนาดของ primitive types สามารถถูกแก้ไขได้

ในการเขียนโปรแกรมภาษา C ชนิดข้อมูล int ใช้หน่วยความจำ 2 ไบต์สำหรับสถาปัตยกรรม 32 บิตและหน่วยความจำ 4 ไบต์สำหรับสถาปัตยกรรม 64 บิต

อย่างไรก็ตามในภาษา Java มันใช้หน่วยความจำ 4 ไบต์สำหรับทั้งสถาปัตยกรรม 32 และ 64 บิต

Portable

Java เป็นแบบพกพาเพราะช่วยให้คุณพกพา Java bytecode ไปยังแพลตฟอร์มใดก็ได้ ไม่ต้องการการ implementation ใด ๆ

High-performance-ประสิทธิภาพสูง

Java เร็วกว่าภาษาการเขียนโปรแกรมตีความแบบดั้งเดิมอื่น ๆ เนื่องจาก Java bytecode นั้น "ใกล้" กับเนทีฟโค้ด ยังช้ากว่าภาษาที่คอมไพล์(compiled languages) เล็กน้อย (เช่น C++)

Java เป็น interpreted language จึงช้ากว่าภาษาที่เป็น compiled languages เช่น C, C++ เป็นต้น

Distributed

Java เป็น distributed เนื่องจากอำนวยความสะดวกให้ผู้ใช้สร้างแอปพลิเคชันแบบ distributed ใน Java RMI และ EJB ใช้สำหรับสร้างแอปพลิเคชันแบบกระจาย (distributed applications) ฟีเจอร์ของ Java นี้ทำให้เราสามารถเข้าถึงไฟล์ได้โดยการเรียกใช้เมธอดจากเครื่องใดก็ได้บนอินเทอร์เน็ต

Multi-threaded

เธรดเป็นเหมือนโปรแกรมย่อยที่ทำงานพร้อมกัน เราสามารถเขียนโปรแกรม Java ที่จัดการกับงานหลายอย่างพร้อมกัน โดยกำหนดหลายเธรด ข้อได้เปรียบหลักของมัลติเธรดคือ ไม่ใช้หน่วยความจำสำหรับแต่ละเธรด มันแบ่งปันพื้นที่หน่วยความจำร่วมกัน เธรดมีความสำคัญสำหรับมัลติมีเดีย เว็บแอปพลิเคชัน ฯลฯ

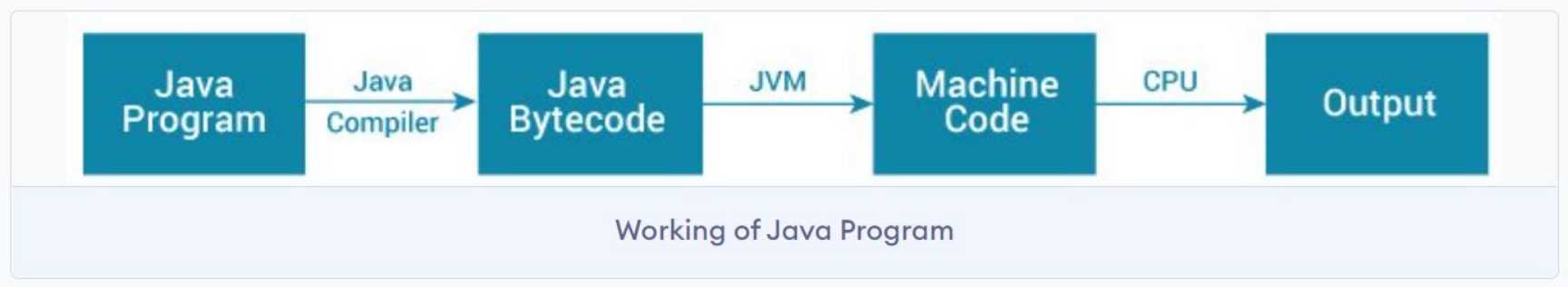
Dynamic

Java เป็นภาษาไดนามิก รองรับการโหลดคลาสแบบไดนามิก หมายความว่าคลาสโหลดได้ตามต้องการ นอกจากนี้ยังรองรับฟังก์ชันจากภาษาท้องถิ่น เช่น C และ C++

Java รองรับการคอมไพล์แบบไดนามิกและการจัดการหน่วยความจำอัตโนมัติ (การรวบรวมขยะ-garbage collection)

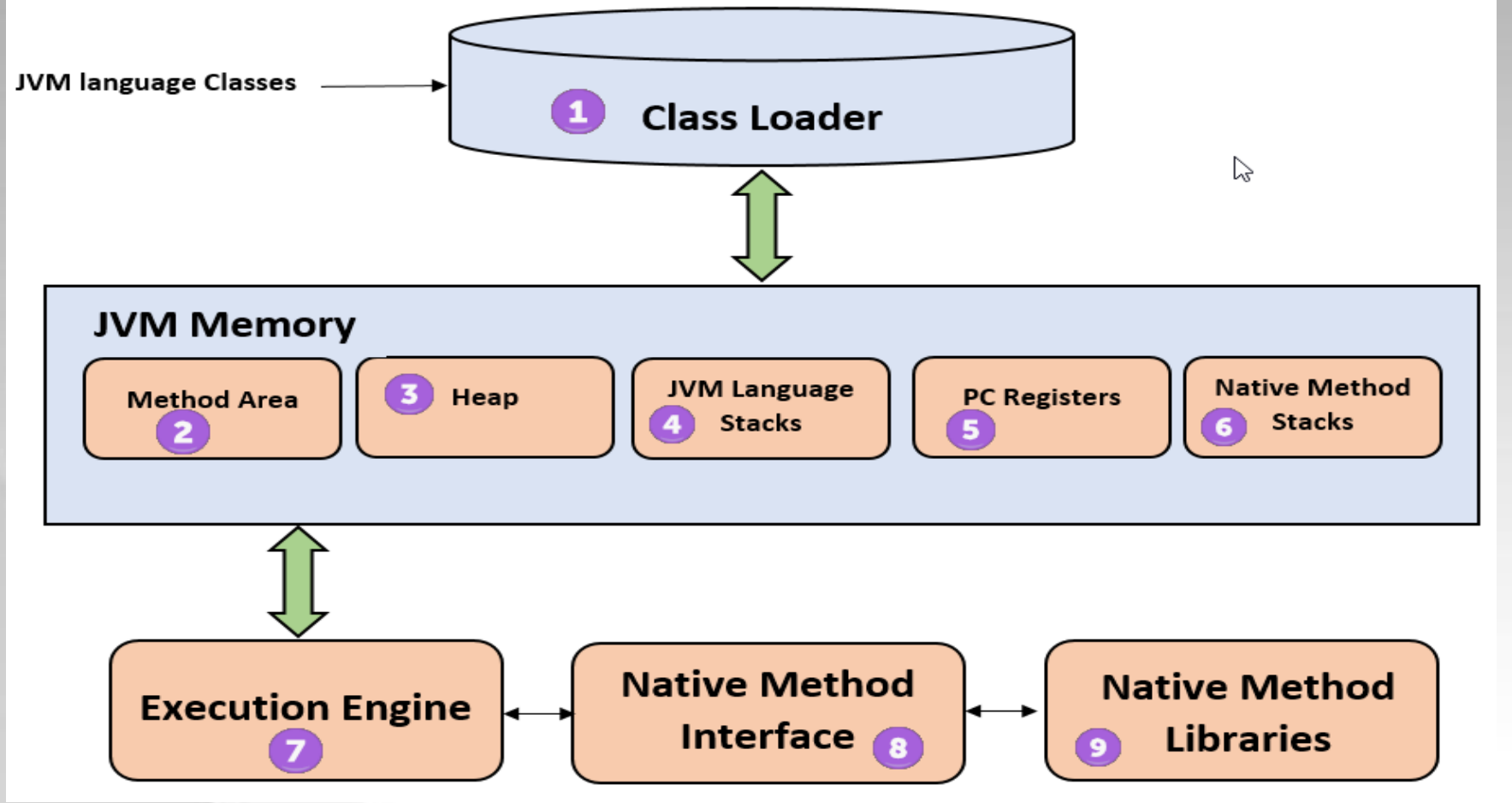
Java Virtual Machine คืออะไร

JVM (Java Virtual Machine) เป็นเครื่องมือที่ช่วยให้คอมพิวเตอร์ของคุณสามารถเรียกใช้โปรแกรม Java ได้ เมื่อคุณรันโปรแกรม Java คอมไพเลอร์ Java จะคอมไพล์โค้ด Java เป็น bytecode ก่อน จากนั้น JVM จะแปล bytecode เป็นรหัสเครื่องดั้งเดิม (ชุดคำสั่งที่ CPU ของคอมพิวเตอร์ควบคุมการดำเนินการได้โดยตรง) Java เป็นภาษาที่ไม่ขึ้นกับแพลตฟอร์ม เมื่อคุณเขียนโค้ด Java ท้ายที่สุดแล้ว โค้ดนี้จะถูกเขียนขึ้นสำหรับ JVM แต่ไม่ใช่เครื่องจริงของคุณ (คอมพิวเตอร์) เนื่องจาก JVM เรียกใช้ Java bytecode ซึ่งไม่ขึ้นกับแพลตฟอร์ม Java จึงไม่ขึ้นกับแพลตฟอร์ม



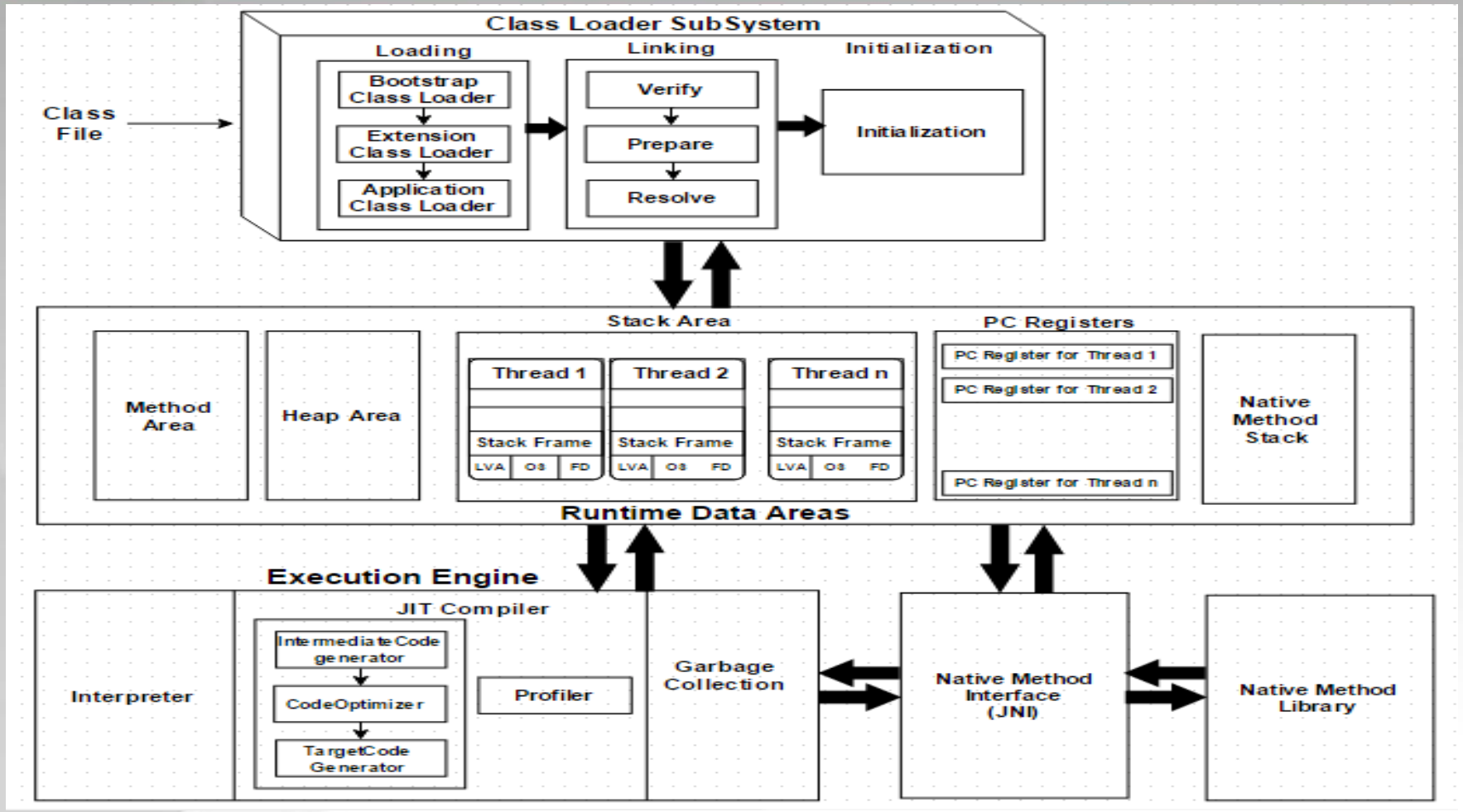
แหล่งที่มา https://www.programiz.com/

สถาปัตยกรรม JVM



แหล่งที่มา https://www.guru99.com/

สถาปัตยกรรม JVM (ต่อ)



แหล่งที่มา https://dzone.com/

สถาปัตยกรรม JVM (ต่อ)

1) ClassLoader : ตัวโหลดคลาสเป็นระบบย่อยที่ใช้สำหรับการโหลดไฟล์คลาส ทำหน้าที่สำคัญ 3 ประการ ได้แก่ กำลังโหลด การเชื่อมโยง และการเริ่มต้น

2) Method Area : JVM Method Area จัดเก็บโครงสร้างคลาส เช่น ข้อมูลเมตาเดต้า constant runtime pool และโค้ดสำหรับเมธอด

3) Heap : ออบเจ็กต์ทั้งหมด ตัวแปรอินสแตนซ์ที่เกี่ยวข้อง และอาร์เรย์จะถูกเก็บไว้ในฮีป หน่วยความจำนี้ใช้ร่วมกันและใช้ร่วมกันในหลายเธรด

4) JVM Stacks : Java language Stack จัดเก็บตัวแปร local variables และผลลัพธ์บางส่วน แต่ละเธรดมีสแต็ก JVM ของตัวเอง ซึ่งสร้างขึ้นพร้อมกันเมื่อสร้างเธรด เฟรมใหม่จะถูกสร้างขึ้นทุกครั้งที่มีการเรียกใช้เมธอด และจะถูกลบออกเมื่อกระบวนการเรียกใช้เมธอดเสร็จสิ้น

สถาปัตยกรรม JVM (ต่อ)

5) PC Registers : PC register เก็บที่อยู่ของคำสั่ง Java virtual machine ซึ่งกำลังดำเนินการอยู่ ใน Java แต่ละเธรดมีการลงทะเบียนพีซีแยกต่างหาก

6) Native Method Stacks : Native Method Stacks ถือคำสั่งของโค้ดเนทีฟขึ้นอยู่กับไลบรารีเนทีฟ มันเขียนเป็นภาษาอื่นแทนจาวา

7) Execution Engine : เป็นซอฟต์แวร์ประเภทหนึ่งที่ใช้ทดสอบฮาร์ดแวร์ ซอฟต์แวร์ หรือระบบทั้งหมด เอ็นจิ้นการดำเนินการทดสอบไม่มีข้อมูลใดๆ เกี่ยวกับผลิตภัณฑ์ที่ทดสอบ

8) Native Method interface : Native Method Interface เป็นเฟรมเวิร์กการเขียนโปรแกรมอนุญาตให้โค้ด Java ที่ทำงานอยู่ใน JVM เรียกโดยไลบรารีและแอปพลิเคชันดั้งเดิม

9) Native Method Libraries : Native Libraries คือคอลเล็กชันของ Native Libraries (C, C++) ซึ่ง Execution Engine ต้องการ

JRE คือ อะไร ?

- JRE (Java Runtime Environment) เป็นแพ็คเกจซอฟต์แวร์ที่มีไลบรารีคลาส Java, Java Virtual Machine (JVM) และส่วนประกอบอื่นๆ ที่จำเป็นสำหรับการรันแอปพลิเคชัน Java
- JRE เป็น superset ของ JVM.

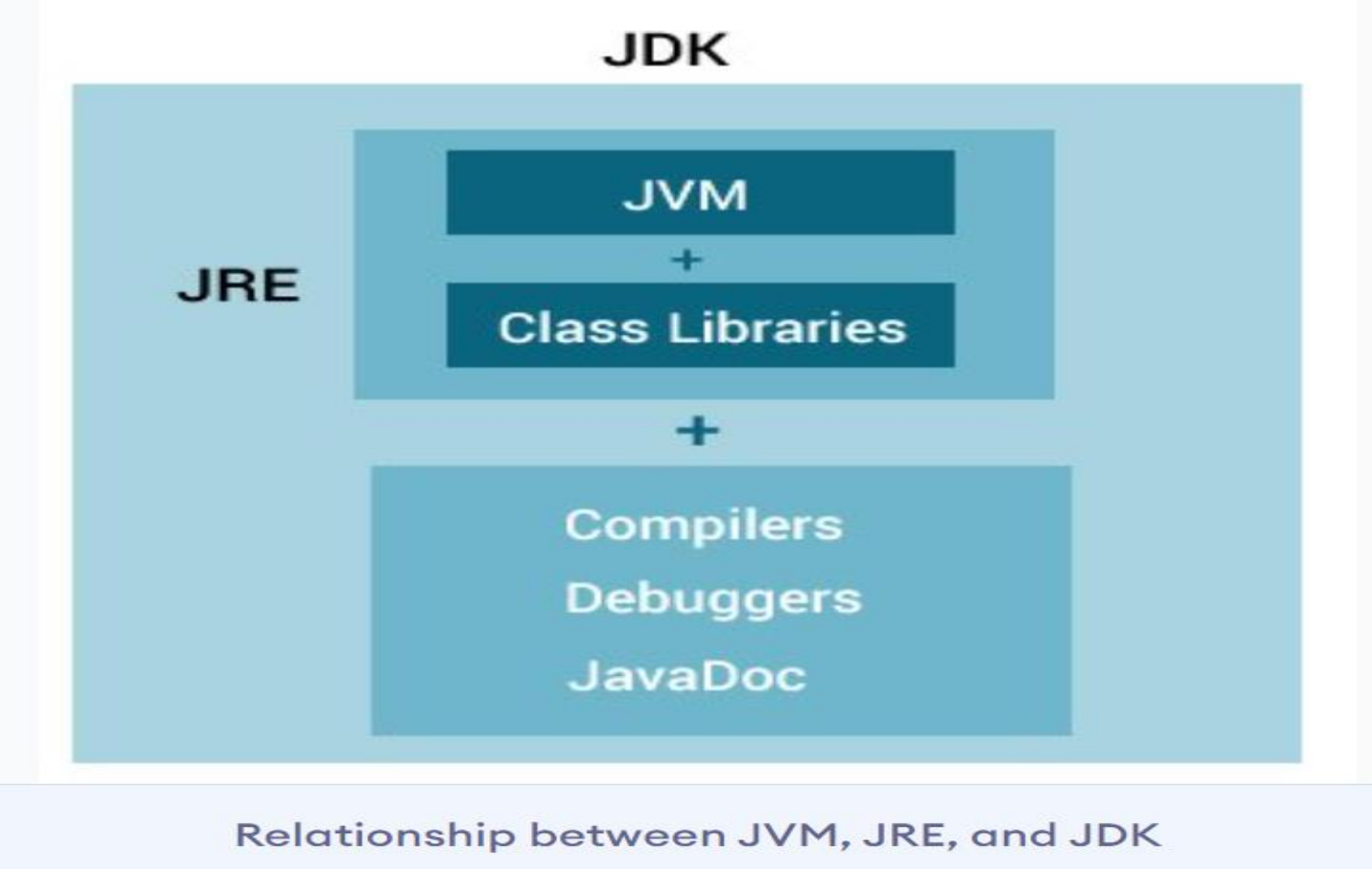
แหล่งที่มา https://www.programiz.com/

JDK คือ อะไร ?

- JDK (Java Development Kit) เป็นชุดพัฒนาซอฟต์แวร์ที่จำเป็นสำหรับการพัฒนาแอปพลิเคชันใน Java เมื่อคุณดาวน์โหลด JDK JRE จะถูกดาวน์โหลดด้วยนอกจาก JRE แล้ว JDK ยังมีเครื่องมือในการพัฒนาอีกจำนวนหนึ่ง (คอมไพเลอร์, JavaDoc, Java Debugger เป็นต้น)

แหล่งที่มา https://www.programiz.com/

ความสัมพันธ์ระหว่าง JVM, JRE และ JDK



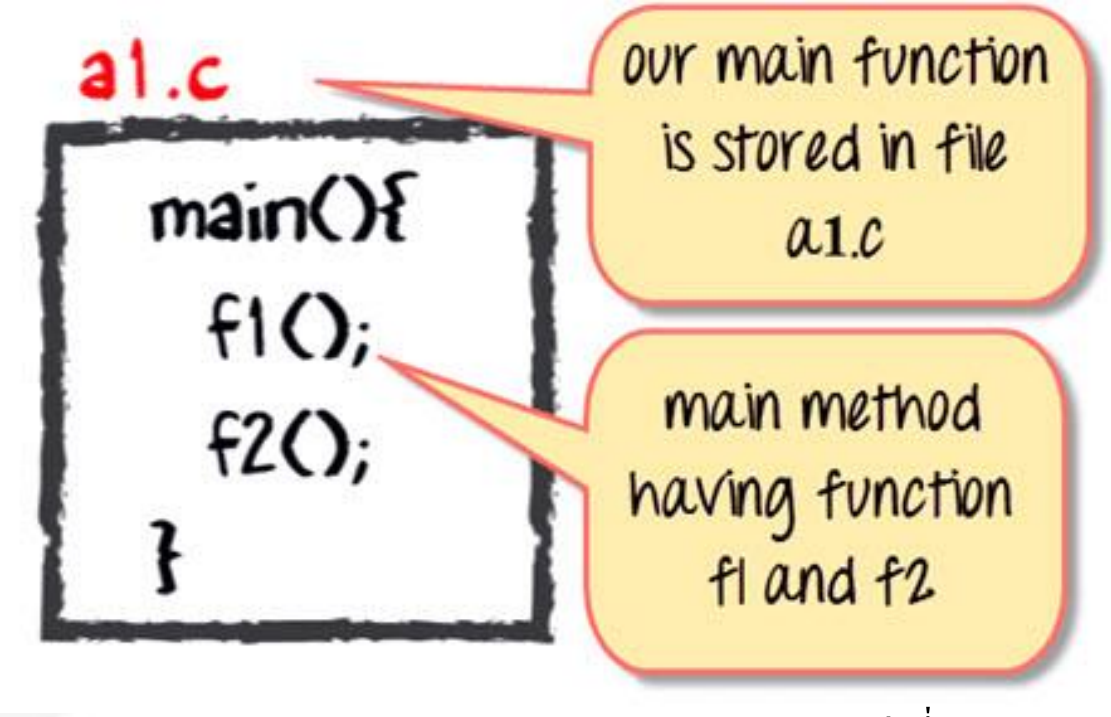
Relationship between JVM, JRE, and JDK

แหล่งที่มา https://www.programiz.com/

การดำเนินการของโค้ดภาษา Java

กระบวนการรวบรวมและดำเนินการโค้ดภาษา C

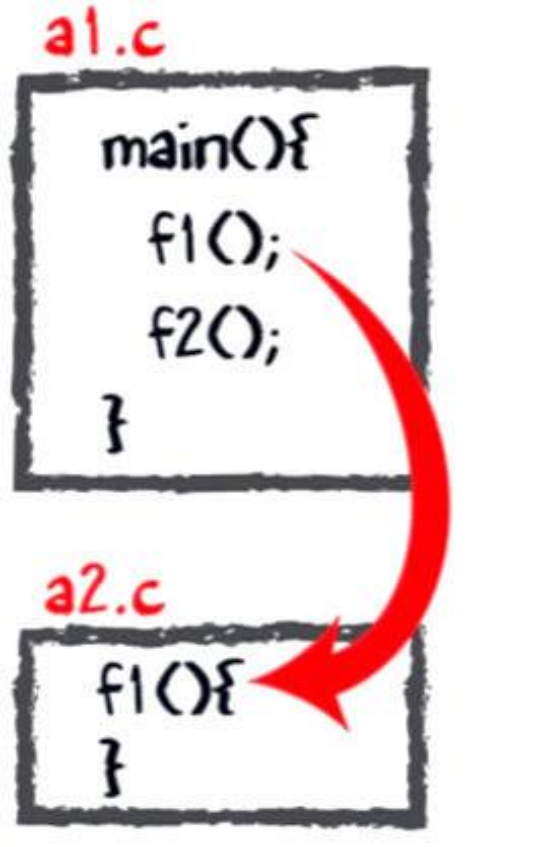
เพื่อทำความเข้าใจกระบวนการคอมไพล์ Java ใน Java ก่อนอื่น มาดูขั้นตอนการคอมไพล์และเชื่อมโยงใน C กันก่อนสมมติว่าใน main คุณเรียกฟังก์ชันสองฟังก์ชัน f1 และ f2 ฟังก์ชั่นหลักถูกเก็บไว้ในไฟล์ a1.c.



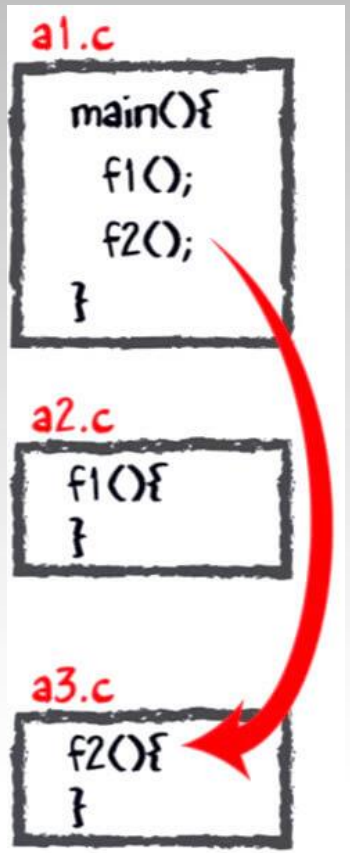
แหล่งที่มา https://www.guru99.com/

กระบวนการรวบรวมและดำเนินการโค้ดภาษา C (ต่อ)

ฟังก์ชั่น f1 ถูกเก็บไว้ในไฟล์ a2.c



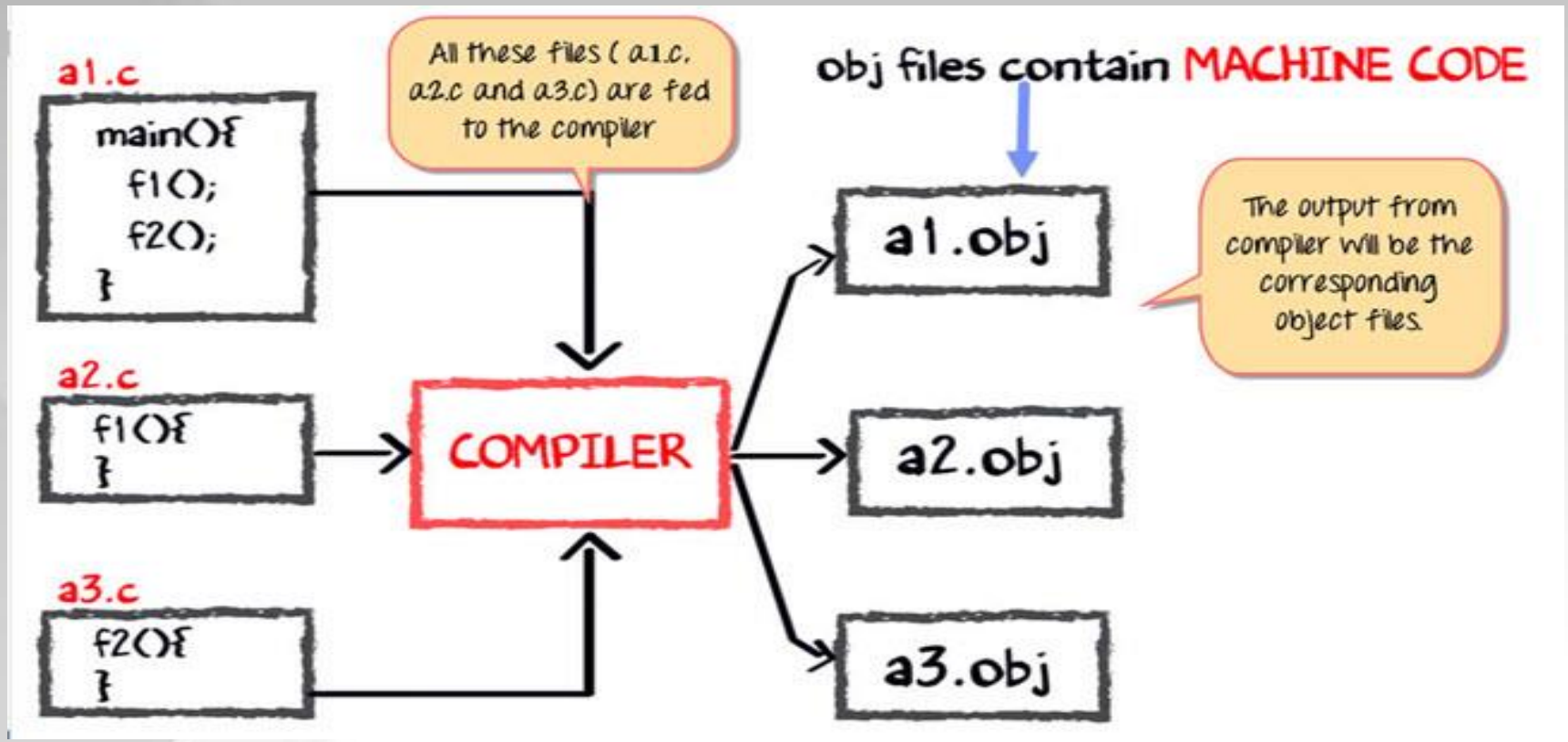
ฟังก์ชั่น f2 ถูกเก็บไว้ในไฟล์ a3.c



แหล่งที่มา https://www.guru99.com/

กระบวนการรวบรวมและดำเนินการโค้ดภาษา C (ต่อ)

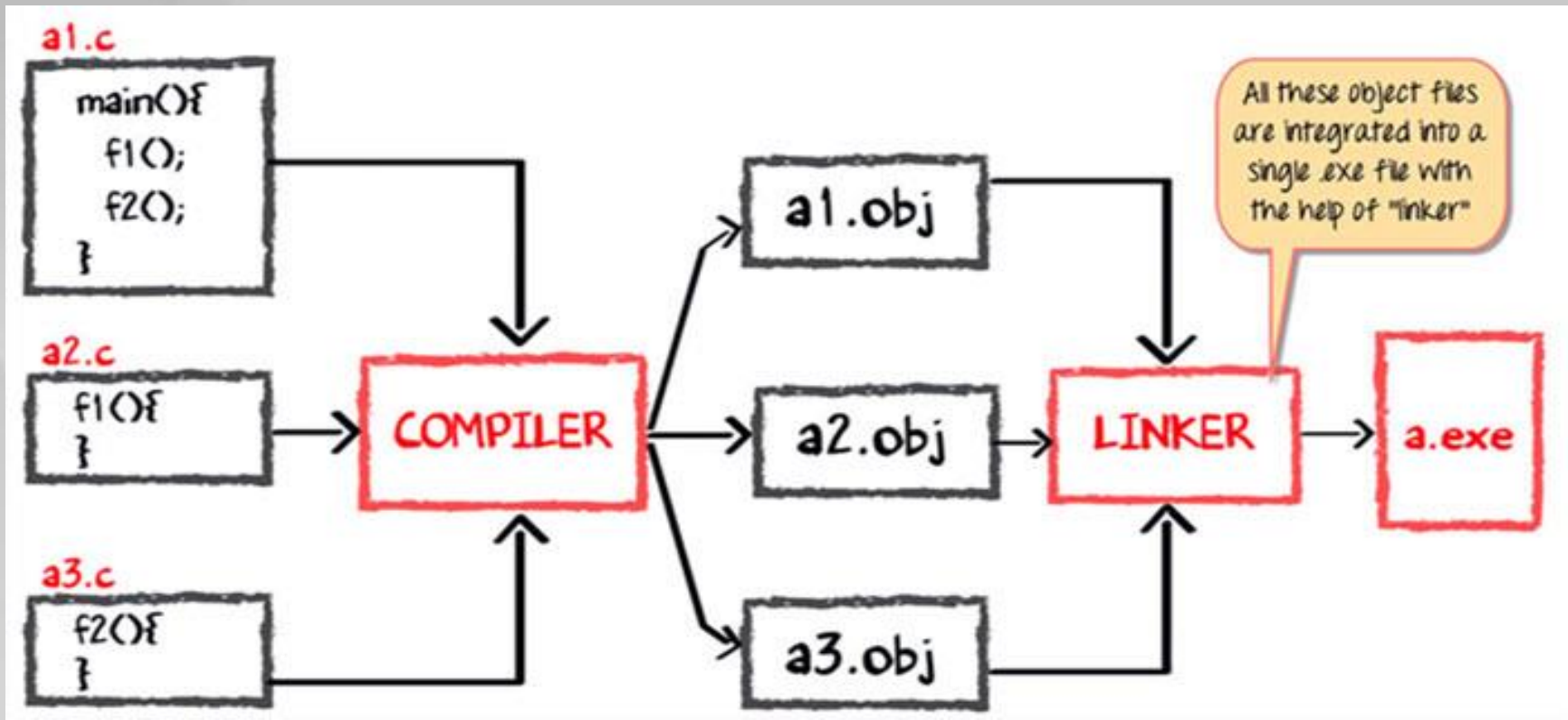
ไฟล์ทั้งหมดเหล่านี้ เช่น a1.c, a2.c และ a3.c ถูกป้อนเข้าสู่คอมไพเลอร์ ผลลัพธ์ที่ได้คือไฟล์อ็อบเจ็กต์ที่เกี่ยวข้องซึ่งเป็นรหัสเครื่อง



แหล่งที่มา https://www.guru99.com/

กระบวนการรวบรวมและดำเนินการโค้ดภาษา C (ต่อ)

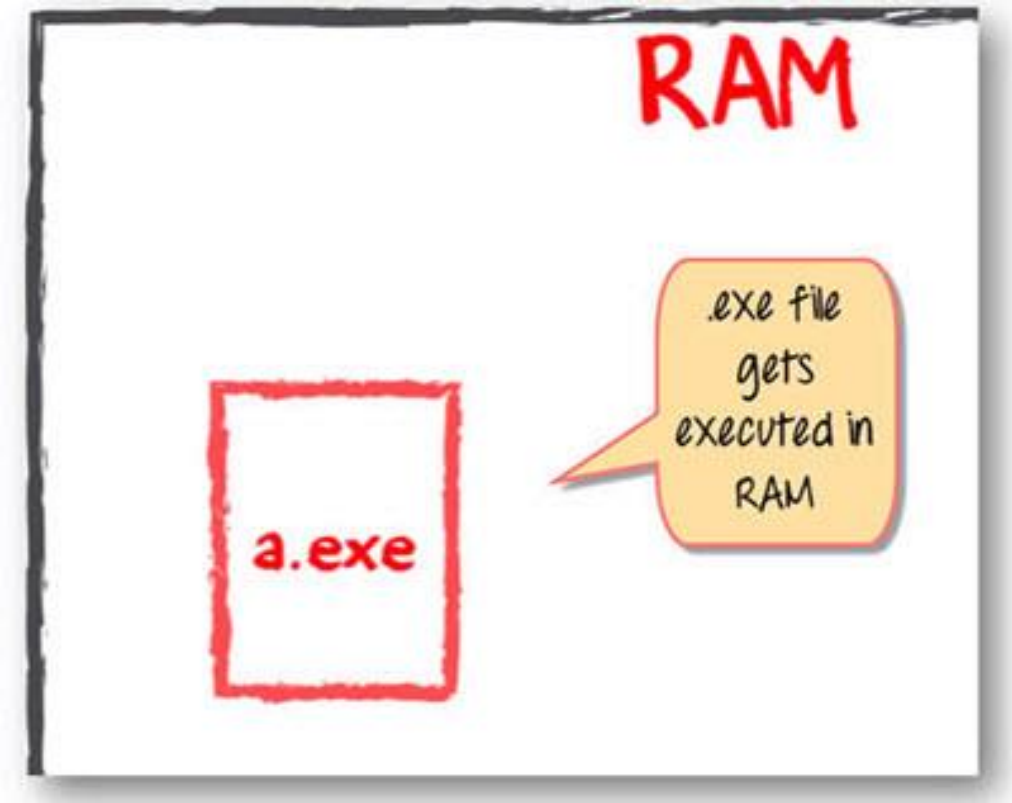
ขั้นตอนต่อไปคือการรวมไฟล์อ็อบเจ็กต์เหล่านี้ทั้งหมดไว้ในไฟล์ .exe ไฟล์เดียวโดยใช้ตัวเชื่อมโยง ตัวเชื่อมโยงจะรวมไฟล์ทั้งหมดเหล่านี้เข้าด้วยกันและสร้างไฟล์ .exe



แหล่งที่มา https://www.guru99.com/

กระบวนการรวบรวมและดำเนินการโค้ดภาษา C (ต่อ)

ในระหว่างการรันโปรแกรม โปรแกรมโหลดเดอร์จะโหลด a.exe ลงใน RAM เพื่อดำเนินการ

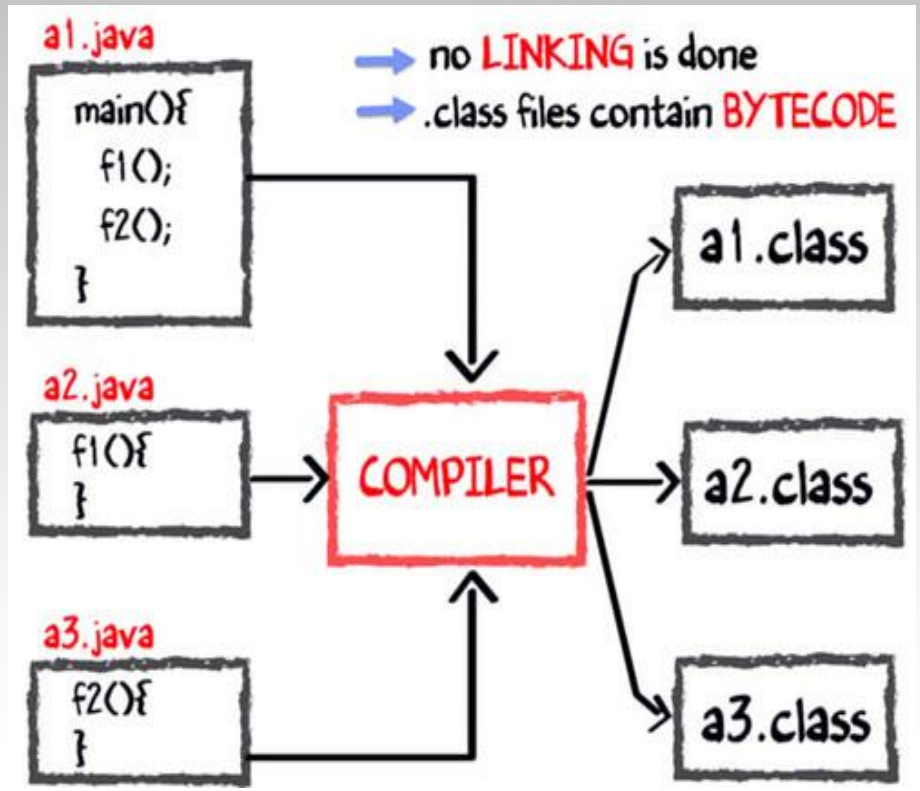


แหล่งที่มา https://www.guru99.com/

การคอมไพล์และประมวลผลโค้ด Java ใน Java VM

ในบทช่วยสอน JVM นี้ มาดูกระบวนการสำหรับ JAVA ในหลักของคุณ คุณมีสองวิธี คือ f1 และ f2

- วิธีการหลักถูกเก็บไว้ในไฟล์ a1.java
- f1 ถูกเก็บไว้ในไฟล์เป็น a2.java
- f2 ถูกเก็บไว้ในไฟล์เป็น a3.java

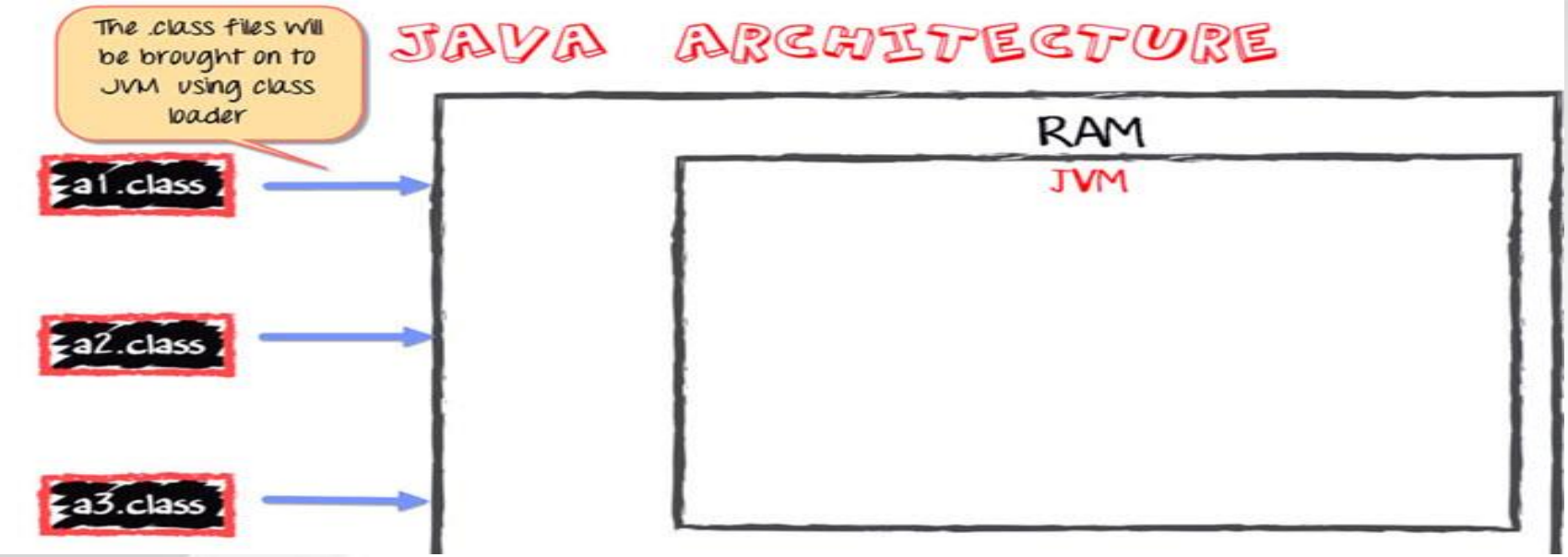


แหล่งที่มา https://www.guru99.com/

การคอมไพล์และประมวลผลโค้ด Java ใน Java VM(ต่อ)

คอมไพเลอร์จะคอมไพล์ไฟล์ทั้งสามและสร้างไฟล์ .class ที่สอดคล้องกัน 3 ไฟล์ซึ่งประกอบด้วยโค้ด BYTE ต่างจาก C ตรงที่ไม่มีการลิงก์

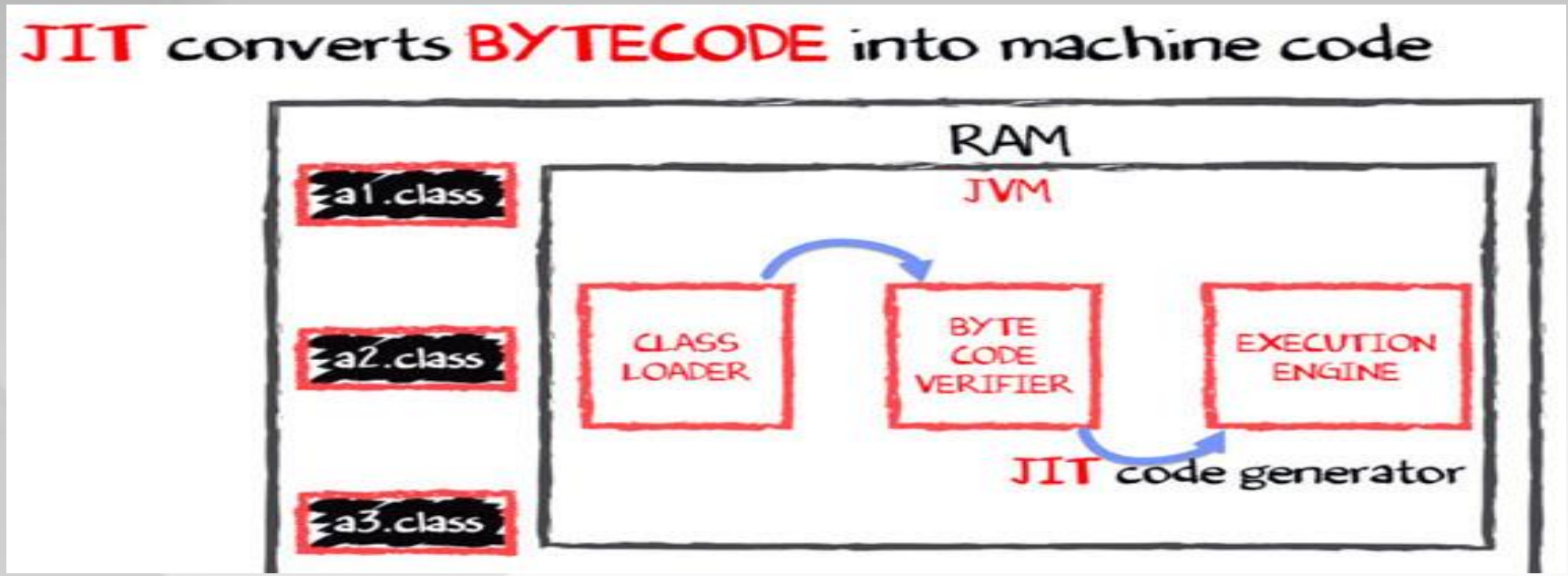
Java VM หรือ Java Virtual Machine อยู่บน RAM ในระหว่างการประมวลผล ไฟล์คลาสจะถูกนำไปยัง RAM โดยใช้ class loader BYTECODE จะถูกตรวจสอบสำหรับการละเมิดความปลอดภัย



แหล่งที่มา https://www.guru99.com/

การคอมไพล์และประมวลผลโค้ด Java ใน Java VM(ต่อ)

ถัดไป เอ็นจิ้นการดำเนินการจะแปลง Bytecode เป็นรหัสเครื่องดั้งเดิม(Native machine code) นี่เป็นหนึ่งในการทำงานในช่วงเวลาคอมไพล์โปรแกรม เป็นหนึ่งในสาเหตุหลักที่ทำให้ Java ค่อนข้างช้า



หมายเหตุ: JIT or Just-in-time compiler เป็นส่วนหนึ่งของ Java Virtual Machine (JVM) มันตีความส่วนหนึ่งของ Bytecode ที่มีการทำงานที่คล้ายกันในเวลาเดียวกัน

แหล่งที่มา https://www.guru99.com/

เหตุใด Java จึงเป็นทั้งภาษาที่ตีความและคอมไพล์- Why is Java both Interpreted and Compiled Language?

ภาษาโปรแกรมจัดอยู่ในประเภท

- ภาษาระดับสูง- Higher Level Language เช่น C++, Java
- ภาษาระดับกลาง- Middle-Level Languages เช่น C
- ภาษาระดับต่ำ- Low-Level Language เช่น Assembly
- ระดับต่ำสุดเป็น Machine Language

เหตุใด Java จึงเป็นทั้งภาษาที่ตีความและคอมไพล์- Why is Java both Interpreted and Compiled Language?

คอมไพเลอร์ คือ โปรแกรมที่แปลงโปรแกรมจากระดับภาษาหนึ่งหรือ Level หนึ่งเป็นอีกระดับหนึ่งหรืออีก Level หนึ่ง ตัวอย่าง การแปลงโปรแกรม C++ เป็น Machine code คอมไพเลอร์ของจาวาแปลงโค้ดจาวาระดับสูงเป็น bytecode (ซึ่งจัดเป็น Machine code)

interpreter คือโปรแกรมที่แปลงโปรแกรมในระดับหนึ่งหรือ Level หนึ่งเป็นภาษาโปรแกรมอื่นในระดับเดียวกันหรือ Level เดียวกัน ตัวอย่างการแปลงโปรแกรม Java เป็น C++

ใน Java ตัวสร้าง Just In Time Code จะแปลง bytecode เป็น Machine code ดั้งเดิมซึ่งอยู่ในระดับการเขียนโปรแกรมเดียวกัน

ดังนั้น Java จึงเป็นทั้งภาษาที่ตีความและภาษาที่คอมไพล์

ทำไม Java ช้า?

สาเหตุหลักสองประการที่อยู่เบื้องหลังความช้าของ Java คือ

- การเชื่อมโยงแบบไดนามิก: ซึ่งจะต่างจากภาษา C ตรงที่ภาษา Java การลิงก์จะกระทำในเวลารันไทม์ ทุกครั้งที่รันโปรแกรมภาษา Java
- Run-time Interpreter: การแปลงรหัสไบต์-BYTECODE เป็นรหัสเครื่องดั้งเดิมจะทำที่รันไทม์ใน Java ซึ่งจะทำให้ความเร็วช้าลง

อย่างไรก็ตาม Java เวอร์ชันล่าสุดได้แก้ไขปัญหาคอขวดของประสิทธิภาพให้อยู่ในระดับที่ดีแล้ว

การคอมไพล์และประมวลผลซอฟต์แวร์

การคอมไพล์และประมวลผลซอฟต์แวร์

ในการเขียนและรันโปรแกรมซอฟต์แวร์ คุณต้องมีสิ่งต่อไปนี้

1) Editor-ตัวแก้ไข – ในการพิมพ์โปรแกรมของคุณ คุณสามารถบันทึกและแก้ไขโปรแกรม

2) Compiler-คอมไพเลอร์ – เพื่อแปลงโปรแกรมภาษาสูงของคุณเป็นรหัสเครื่องดั้งเดิม

3) Linker-ตัวเชื่อมโยง – เพื่อรวมไฟล์ที่อ้างอิงในโปรแกรมเข้าด้วยกัน

4) Loader-ตัวโหลดไฟล์ – เพื่อโหลดไฟล์จากอุปกรณ์จัดเก็บข้อมูล เช่น Hard Disk, Flash ลงใน RAM เพื่อดำเนินการ การโหลดจะทำโดยอัตโนมัติเมื่อคุณรันโค้ดของคุณ

5) Execution-ประมวลผล – การดำเนินการตามจริงของรหัสที่ระบบปฏิบัติการและตัวประมวลผลของคุณจัดการ

ตัวอย่างโปรแกรม Simple.java

```
1. class Simple
2. {
3.     public static void main(String args[])
4.     {
5.      System.out.println("Hello Java");
6.     }
7. }
```

ชื่อไฟล์กับชื่อคลาสต้องเหมือนกัน

compile program: javac Simple.java

Execute program: java Simple

Output:

```
Hello Java
```

พารามิเตอร์ที่ใช้ในโปรแกรม Java

- คีย์เวิร์ด class ใช้เพื่อประกาศคลาสใน Java
- คีย์เวิร์ด public เป็น access modifier ที่แสดงถึงการมองเห็น หมายความว่าทุกคนมองเห็นได้
- คีย์เวิร์ด static ถ้าเราประกาศวิธีการใด ๆ เป็นแบบ static จะเรียกว่าวิธีการแบบคงที่ ข้อได้เปรียบหลักของวิธีการแบบคงที่คือไม่จำเป็นต้องสร้าง object เพื่อเรียกใช้ วิธีการแบบคงที่ เมธอด main() ดำเนินการโดย JVM ดังนั้นจึงไม่จำเป็นต้องสร้าง object เพื่อเรียกใช้เมธอด main() ดังนั้นจึงช่วยประหยัดหน่วยความจำ
- void เป็นประเภทส่งคืนของเมธอด หมายความว่าไม่คืนค่าใด ๆ
- main หมายถึงจุดเริ่มต้นของโปรแกรม
- String[] args หรือ String args[] ใช้สำหรับอาร์กิวเมนต์บรรทัดคำสั่ง เราจะหารือกันในส่วนต่อๆ ไป
- System.out.println() ใช้สำหรับพิมพ์คำสั่งหรือข้อความ System เป็นคลาส out เป็น object ของคลาส PrintStream, println() เป็นวิธีการของคลาส PrintStream

Q & A