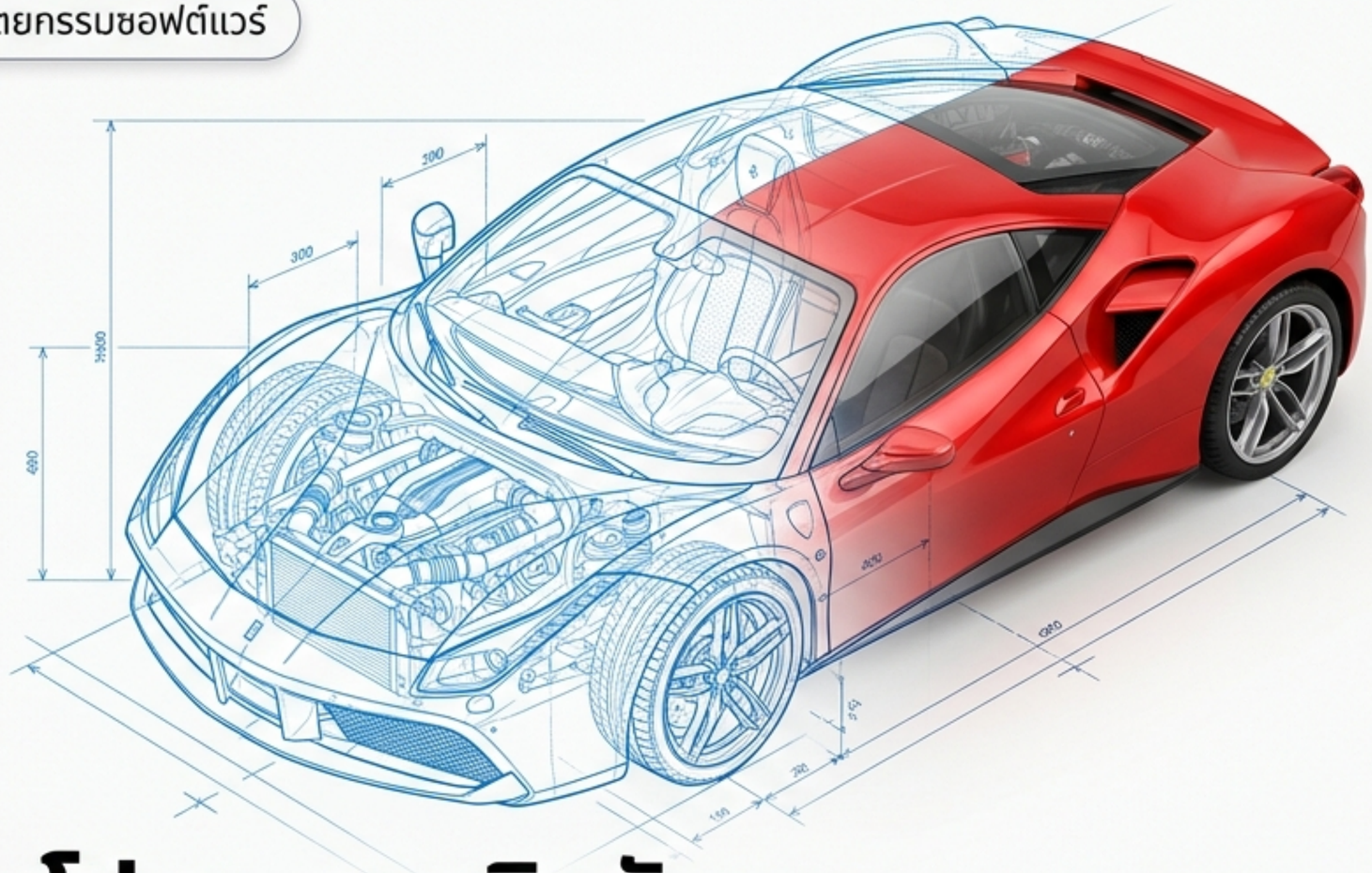
บทสรุปแนวคิดเชิงสถาปัตยกรรมซอฟต์แวร์

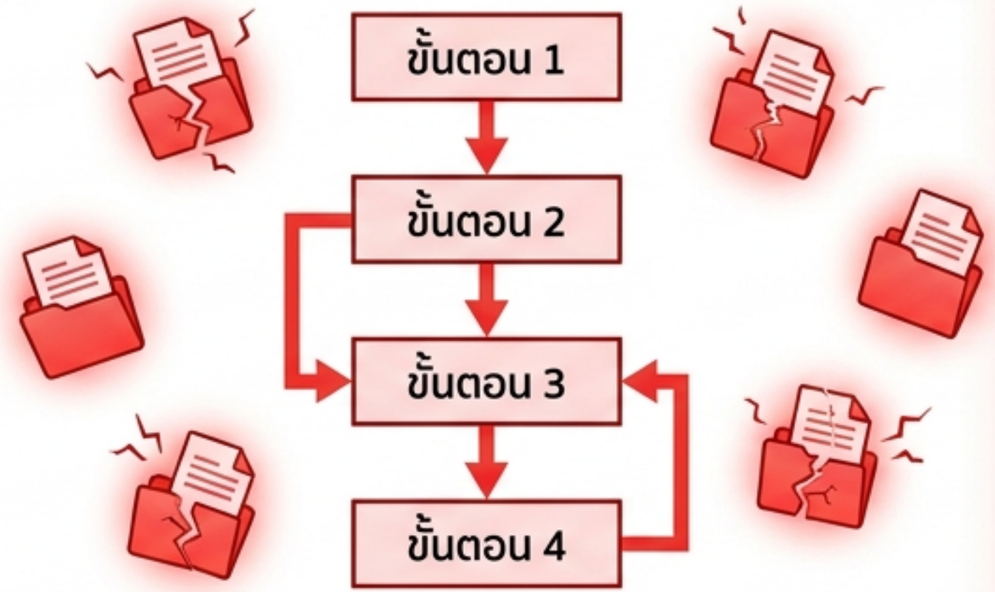


การเขียนโปรแกรมเชิงวัตถุและพื้นฐานภาษาจาวา

จากพิมพ์เขียวทางความคิด สู่การเขียนโค้ดในโลกจริง

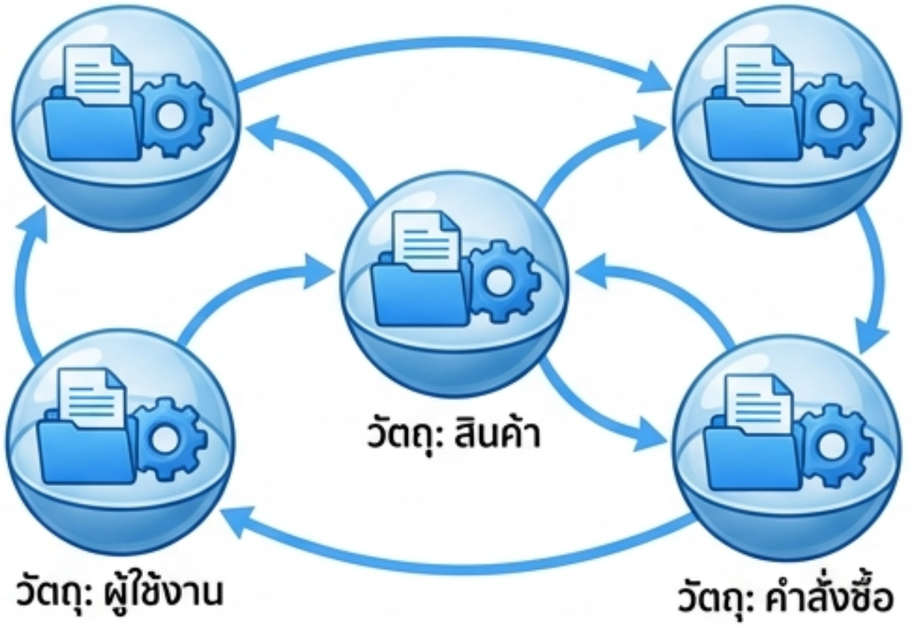
การเปลี่ยนผ่านแนวคิด: กระบวนการ สู่ วัตถุ

การเขียนโปรแกรมเชิงกระบวนการ (Procedural)



- แบ่งโปรแกรมตาม ลำดับขั้นตอนการทำงาน
- แยกข้อมูล (Data) ออกจากคำสั่ง (Function)
- **ปัญหา:** เมื่อสเกลใหญ่ขึ้น ข้อมูลถูกเรียกใช้กระจัดกระจาย ตรวจสอบยากว่าถูกแก้ไขจากจุดใด

การเขียนโปรแกรมเชิงวัตถุ (OOP)



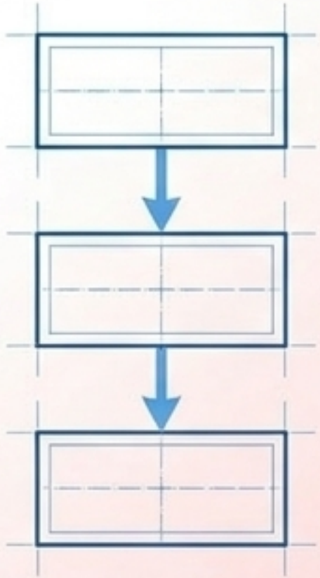
- แบ่งโปรแกรมตาม วัตถุ (จำลองโลกจริง)
- รวมข้อมูลและพฤติกรรมเข้าไว้ด้วยกันภายในวัตถุ
- **ข้อดี:** พัฒนาเร็ว, นำกลับมาใช้ใหม่ได้ง่าย (Reusable), ลดข้อผิดพลาด

โครงสร้างพื้นฐานของการเขียนโปรแกรม

หลักการเชิงกระบวนการ 3 รูปแบบที่เป็นรากฐานของลอจิก:

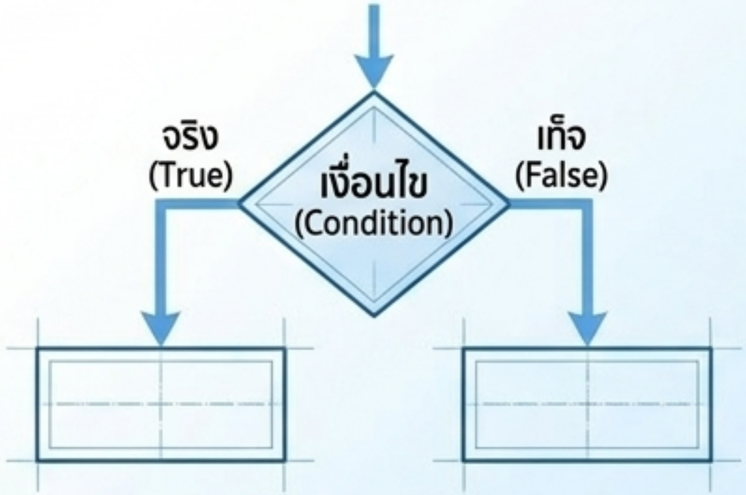
1. Sequence

การทำงานแบบตามลำดับ:
ทำงานจากบนลงล่าง
มีจุดเริ่มต้นและจุดสิ้นสุดเดียว



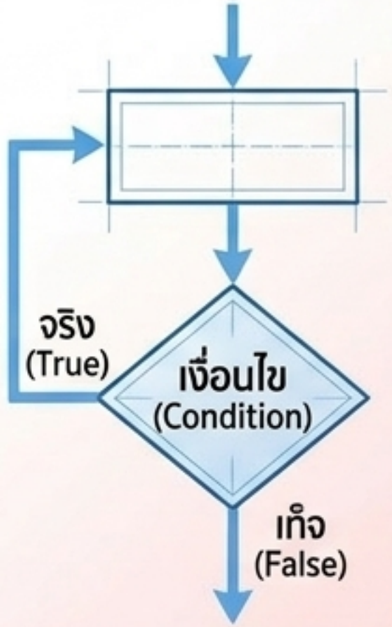
2. Decision

การเลือกกระทำตามเงื่อนไข:
ตัดสินใจจากเงื่อนไข
(เช่น ถ้าคะแนน >= 80 ให้เกรด A)

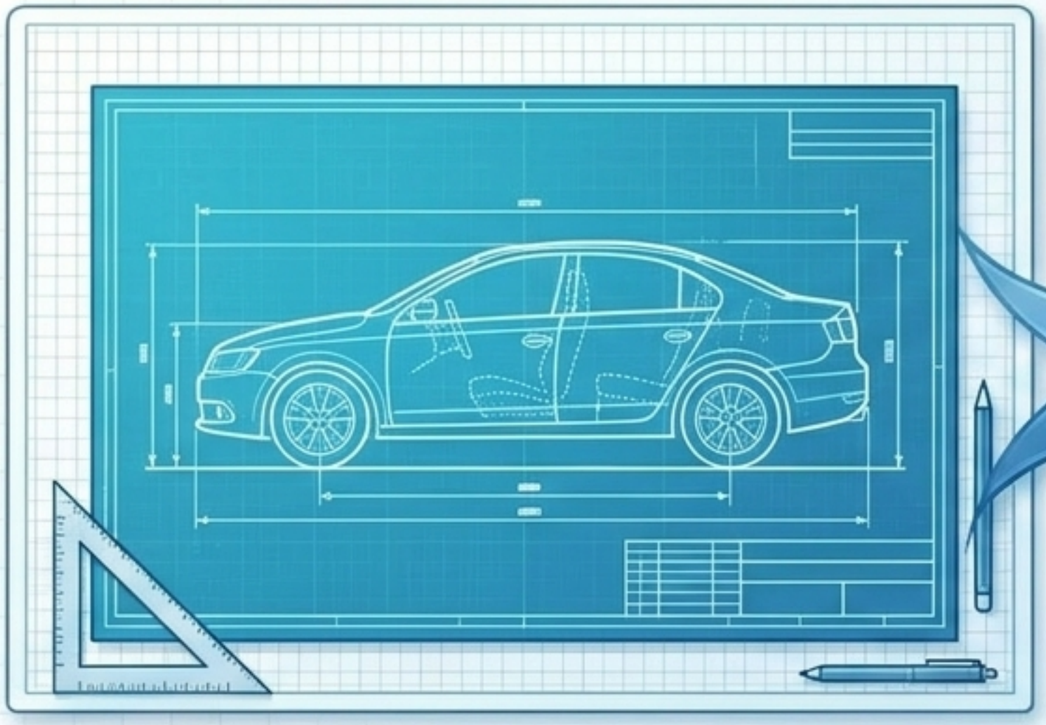


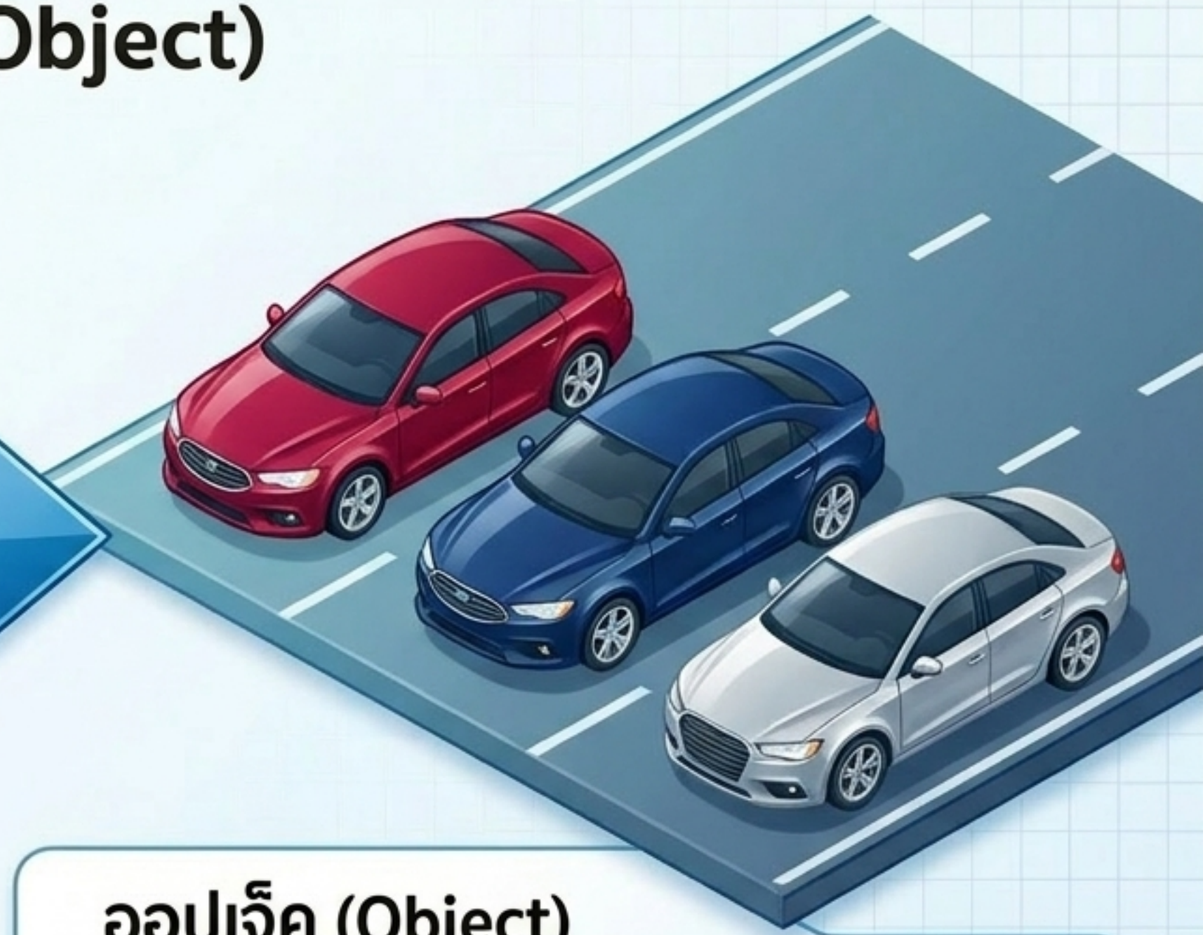
3. Loop

การทำซ้ำ: ทำกระบวนการเดิมซ้ำๆ
จนกว่าเงื่อนไขจะเป็นเท็จ



แม่พิมพ์ และ ความจริง (Class vs. Object)





คลาส (Class)

- แม่พิมพ์ หรือ เค้าโครง (Blueprint)
- กำหนดลักษณะและพฤติกรรมพื้นฐาน
- ตัวอย่าง: คลาส รถยนต์

ออปเจ็ค (Object)

- ตัวตนที่ถูกสร้างขึ้นจริงจากแม่พิมพ์
- 1 คลาส สามารถสร้างออปเจ็คได้ไม่จำกัด
- ตัวอย่าง: รถยนต์สีแดงของนาย ก., รถยนต์สีน้ำเงินของนาย ข.

โครงสร้างภายในของวัตถุ (Anatomy of an Object)

นักศึกษา (Student)

แอททริบิวท์
(Attribute / ข้อมูล)

รหัส, ชื่อ, นามสกุล, เกรดเฉลี่ย

เมธอด
(Method / พฤติกรรม)

ลงทะเบียนเรียน, สอบ, เดิน, ดีใจ

สุนัข (Dog)

แอททริบิวท์
(Attribute / ข้อมูล)

ชื่อ, พันธุ์, สี, สี, ส่วนสูง, น้ำหนัก

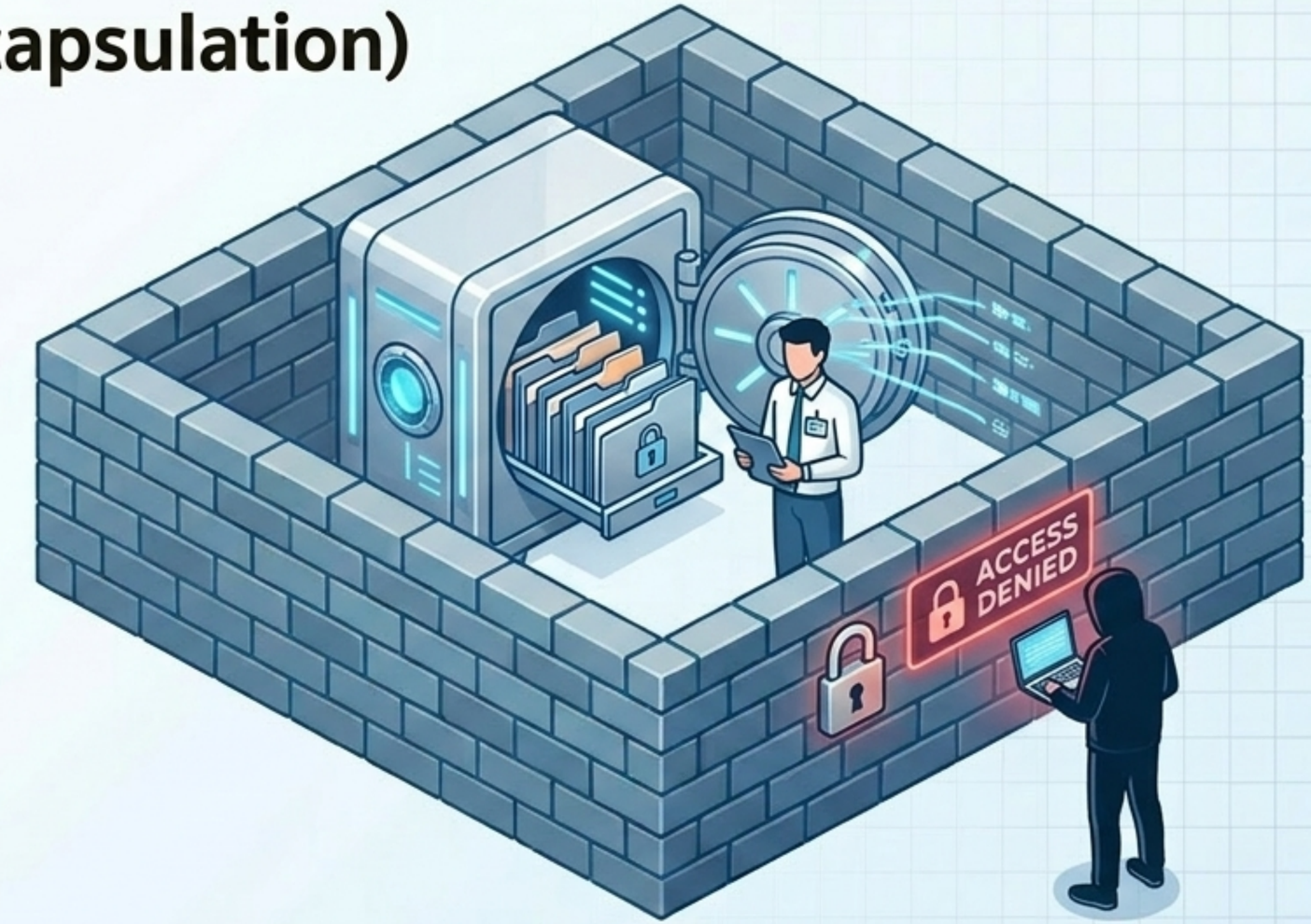
เมธอด
(Method / พฤติกรรม)

เห่า, คลาน, กระดิกหาง, วิ่ง, กิน

เสาหลักที่ 1: การห่อหุ้ม (Encapsulation)

Definition:

ซ่อนวิธีการทำงานและข้อมูลไว้ภายใน (Information Hiding) ป้องกันไม่ให้สิ่งภายนอกที่ไม่เกี่ยวข้องเข้ามาแทรกแซง



สร้าง “กำแพง” ป้องกันข้อมูล (Data) ไม่ให้ถูกแก้ไขโดยตรง

บุคคลภายนอกต้องติดต่อผ่านช่องทางที่อนุญาต (Method) เท่านั้น

เพิ่มความปลอดภัยและป้องกันระบบโดยรวมทำงานผิดเพี้ยน

เสาหลักที่ 2: การสืบทอด (Inheritance)

คลาสแม่: พนักงาน (Employee)

- ✓ - รหัสพนักงาน (ID)
- ✓ - ตำแหน่ง (Role)

คลาสลูก: ผู้จัดการ (Manager)

- ✓ - รหัสพนักงาน (ID)
- ✓ - ตำแหน่ง (Role)
- \+ รถประจำตำแหน่ง (Company Car)

Definition:

การสร้างคลาสใหม่โดยรับเอาคุณสมบัติจากคลาสเดิมที่มีอยู่แล้ว และเพิ่มความสามารถใหม่เข้าไปได้

- สืบทอดคุณสมบัติพื้นฐาน ไม่ต้องเขียนโค้ดซ้ำ
- นำสิ่งที่เคยสร้างแล้วกลับมาใช้ใหม่ (Re-use)
- ประหยัดเวลาในการพัฒนาซอฟต์แวร์อย่างมหาศาล

เสาหลักที่ 3: การพ้องรูป (Polymorphism)

เมธอดหลัก:
Calculate()


```
public void calculate(int x)
{
}
```

สำหรับลูกค้าประจำ ปัดเศษทิ้ง

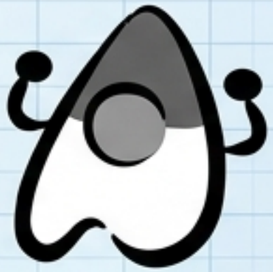
```
public void calculate(double x)
{
}
```

สำหรับลูกค้าขาจร คิดเงินละเอียดทุกสลึง

Definition: หนึ่งรูปแบบการติดต่อ แต่ตอบสนองได้หลายรูปแบบ (Overloading / Overriding)

Key Takeaway: สิ่งหนึ่งสามารถทำงานได้หลากหลายรูปแบบตามบริบท (Context) ที่เกิดขึ้นในขณะนั้น

กำเนิดภาษาจาวา (The Genesis of Java)



1990
The Green Project

เริ่มต้นโครงการโดย James Gosling ณ บริษัท Sun Microsystems

1992
Oak & *7 Device

สร้างภาษาขนาดเล็กเพื่อควบคุมอุปกรณ์อิเล็กทรอนิกส์ เช่น รีโมต Star Seven

1995
Java & The Web

เปลี่ยนชื่อเป็น JAVA ได้รับความนิยมสูงสุดจากการรองรับโปรแกรมเทียร์บน Web Browser

เป้าหมายสูงสุด: ต้องเป็นภาษาที่ไม่ยึดติดกับฮาร์ดแวร์ (Platform-Independent)

อาณาจักรของจาวาแพลตฟอร์ม (The Java Ecosystem)

J2SE (Standard Edition)

แพลตฟอร์มมาตรฐานสำหรับแอปพลิเคชันทั่วไป
มีคลาสและอินเทอร์เฟซให้ใช้กว่า 3,279 แบบ

J2EE (Enterprise Edition)

สำหรับการพัฒนาระบบองค์กรขนาดใหญ่
(Multitiered Programs)

J2ME (Micro Edition)

สำหรับอุปกรณ์ขนาดเล็ก เช่น โทรศัพท์มือถือ,
PDA, และกล่องทีวี (Set-top box)

ปัญหาการแปลภาษา: Compiler vs. Interpreter

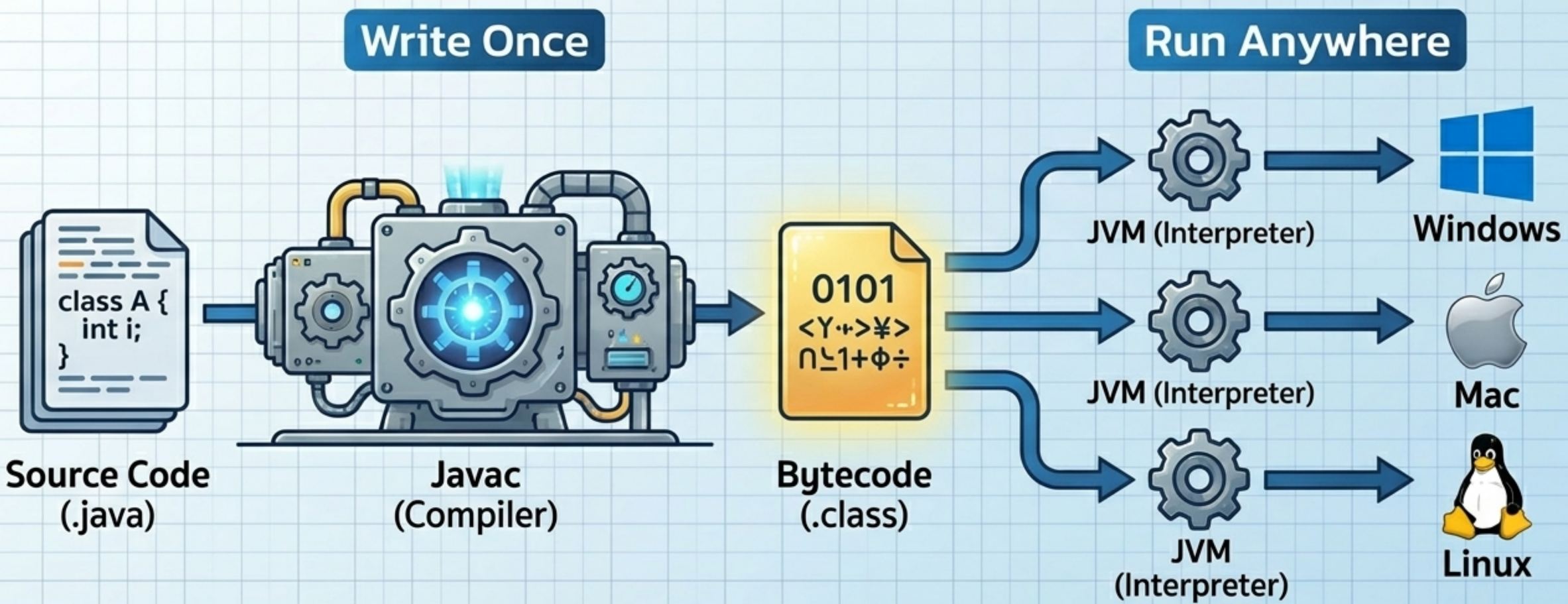
คอมไพเลอร์ – Compiler

- แปล Source Code ทั้งโปรแกรม เป็น Executable Code ทีเดียว
- **ข้อดี**: ทำงานเร็วมาก
- **ข้อเสีย**: ยึดติดกับระบบปฏิบัติการ (Platform Specific) ต้องคอมไพล์ใหม่เมื่อย้ายเครื่อง

อินเตอร์พรีตเตอร์ – Interpreter

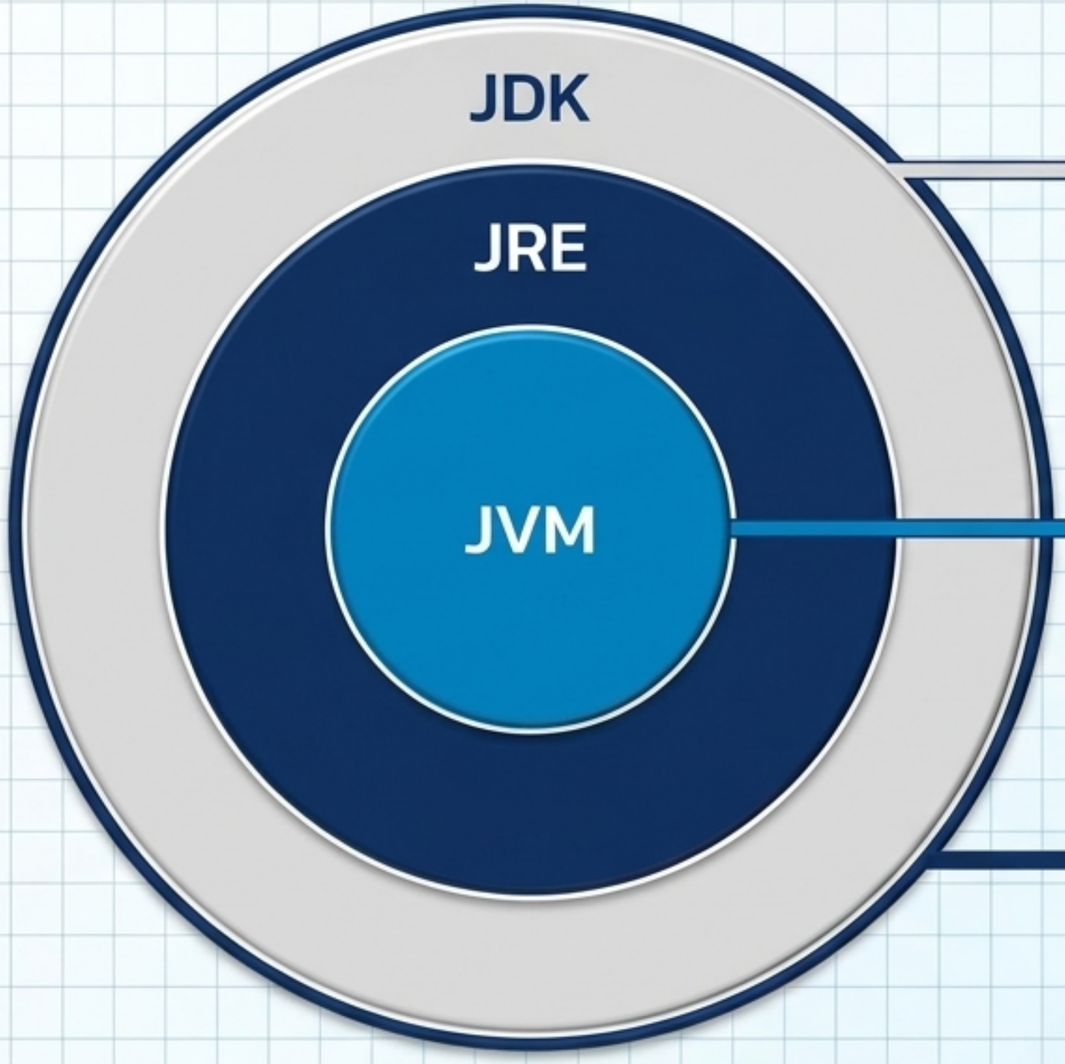
- แปล Source Code ทีละบรรทัด และทำงานทันที
- **ข้อดี**: ข้ามแพลตฟอร์มได้ง่าย
- **ข้อเสีย**: ทำงานช้ากว่าคอมไพเลอร์

เวทมนตร์ของจาวา (The Execution Pipeline)



จาวาคอมไพเลอร์ แปลโค้ดให้เป็น Bytecode ซึ่งเป็นภาษากลาง จากนั้น JVM บนเครื่องปลายทางจะทำหน้าที่เป็นอินเตอร์พรีตเตอร์ แปล Bytecode เป็นภาษาเครื่องที่เหมาะสมกับระบบปฏิบัติการนั้นๆ

สถาปัตยกรรมซ้อนทับ (Demystifying JDK, JRE, JVM)



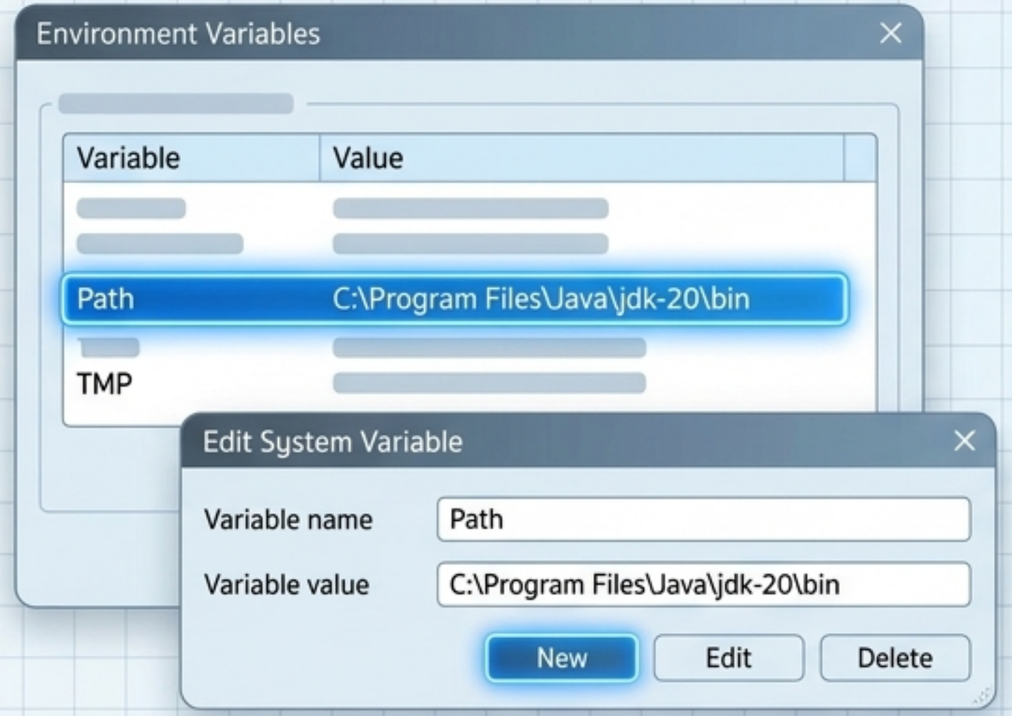
Java Development Kit:
ชุดเครื่องมือสำหรับผู้พัฒนา
ประกอบด้วย JRE + Javac
(Compiler) + เครื่องมือดีบัก

Java Runtime Environment:
สภาพแวดล้อมสำหรับรันโปรแกรม
ประกอบด้วย JVM + Libraries
ที่จำเป็นสำหรับผู้ใช้งาน

Java Development Kit: ชุดเครื่องมือ
สำหรับผู้พัฒนา ประกอบด้วย JRE +
Javac (Compiler) + เครื่องมือดีบัก

เตรียมความพร้อมสู่พื้นที่ปฏิบัติงาน (Workspace Setup)

1. การกำหนด PATH



หลังติดตั้ง JDK ต้องตั้งค่า Environment Variables -> Path
เพื่อให้ระบบปฏิบัติการรู้จักคำสั่งคอมไพล์จากทุกโฟลเดอร์

2. โปรแกรมเขียนโค้ด (IDE & Editors)

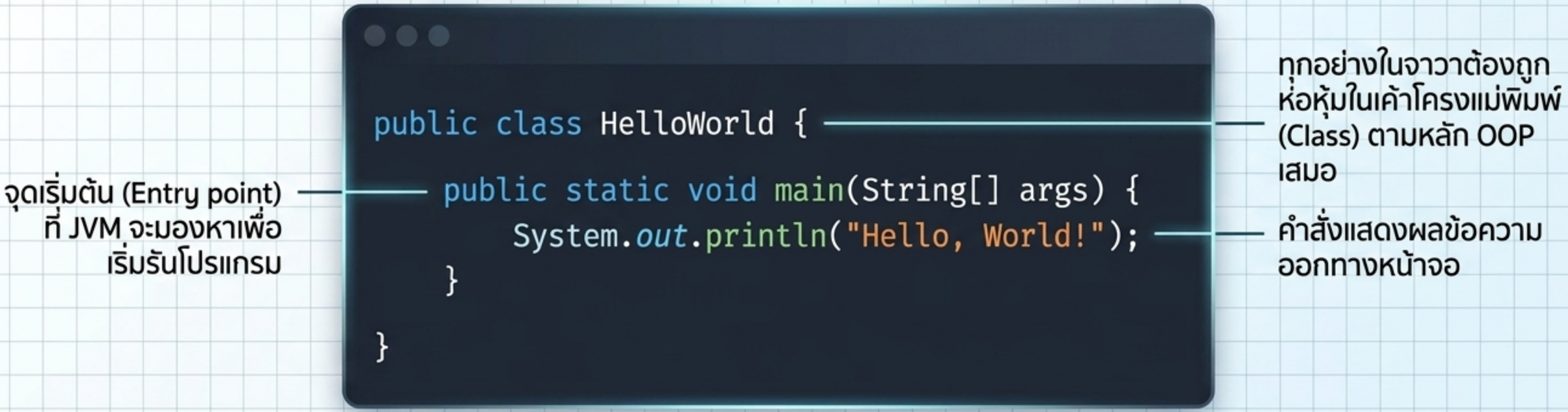
Basic: Notepad
(มาพร้อม Windows)

Professional:
Netbeans, Eclipse

Educational:
J-Lab (จุฬาฯ)

Professional:
Netbeans, Eclipse
(มีเครื่องมือช่วยครบวงจร)

โค้ดแรกในโลกจริง (The First Artifact: Hello World)



พิมพ์เขียวของคุณพร้อมแล้ว... เริ่มต้นสร้างโลกด้วยคีย์บอร์ดของคุณ