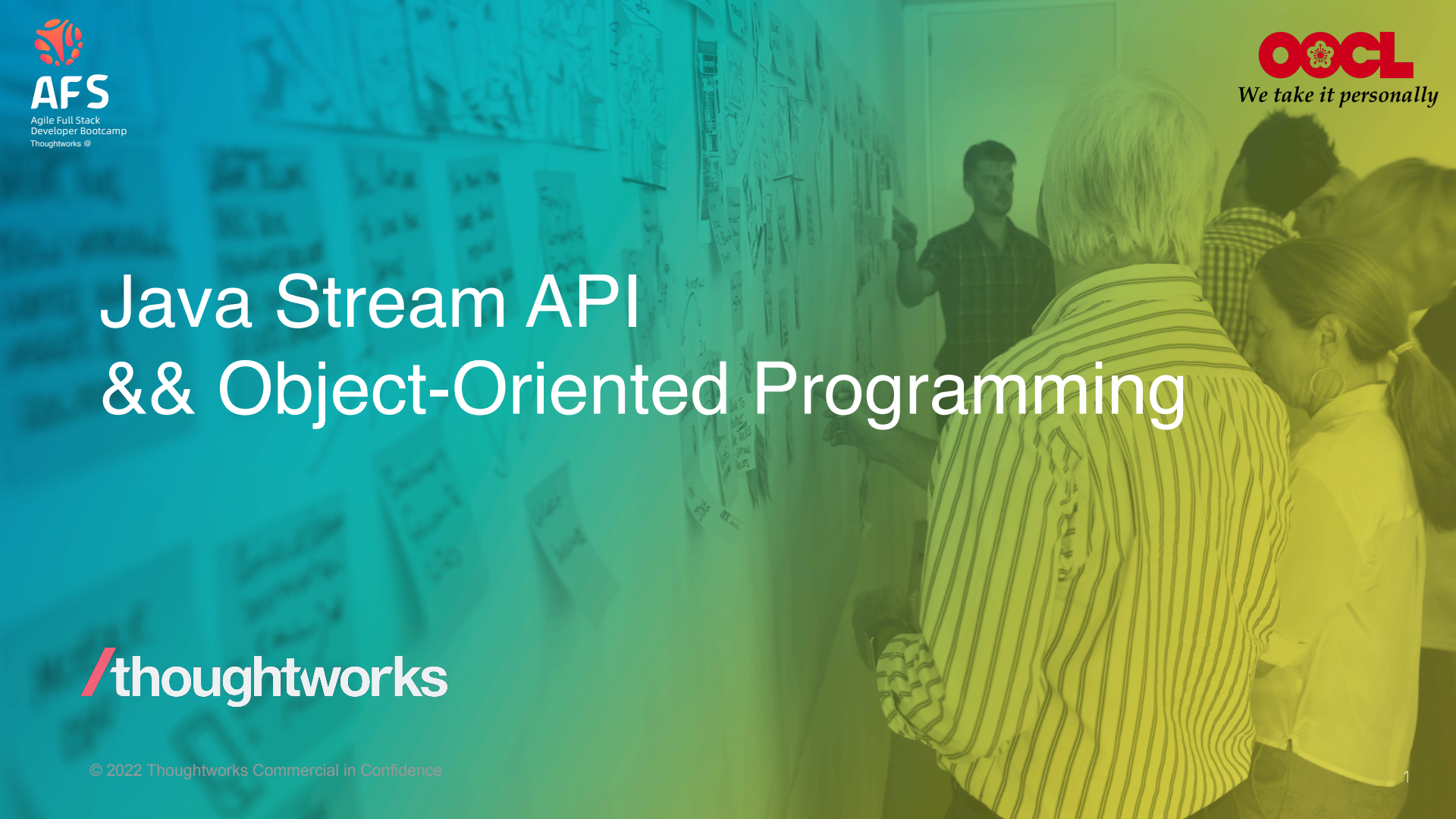
AFS
Agile Full Stack
Developer Bootcamp
Thoughtworks @

OOCL
We take it personally

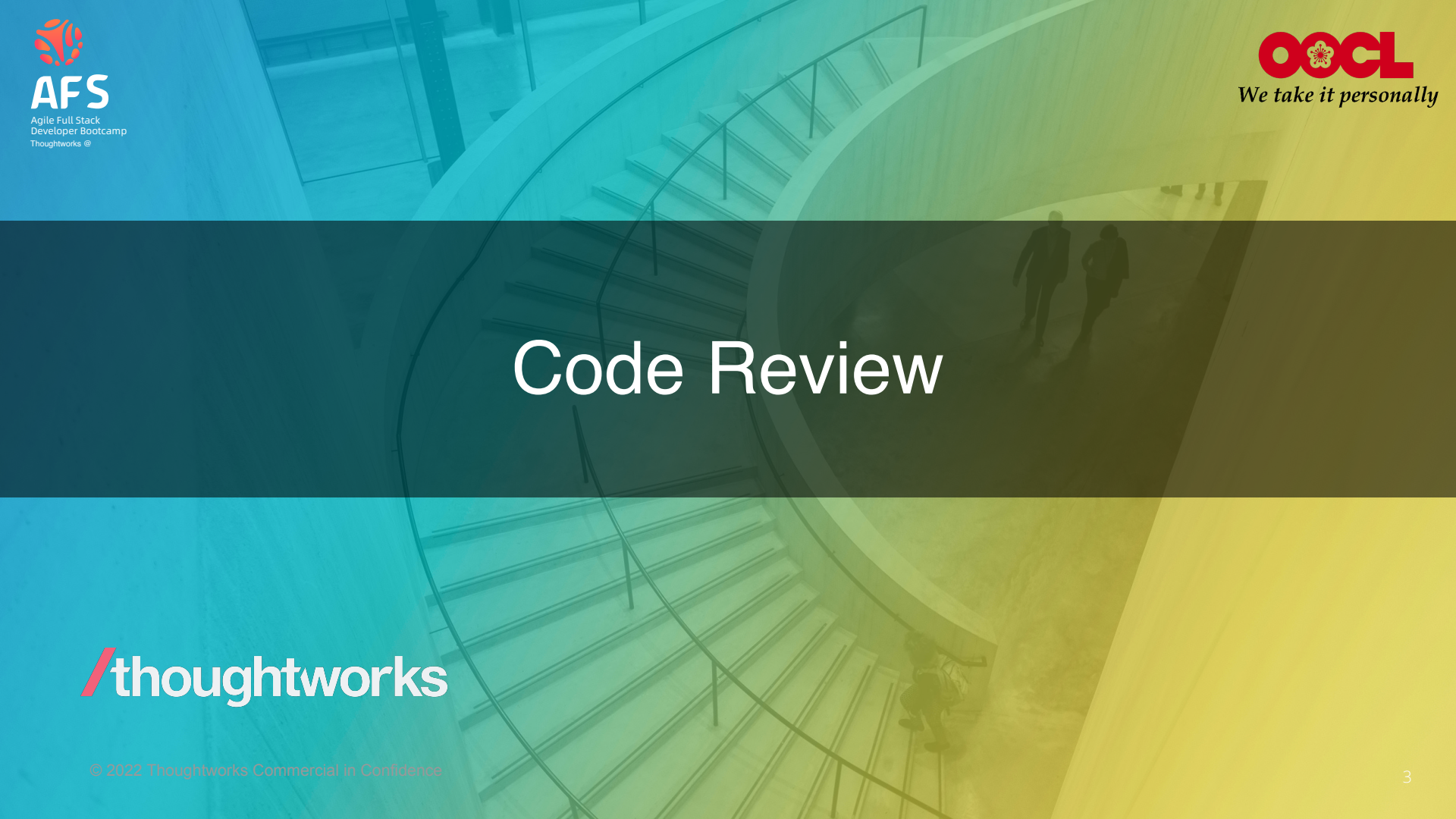
Java Stream API
&& Object-Oriented Programming



thoughtworks

Today's Topics

- **Code review**
- **Java Stream API**
- **Object-Oriented Programming**



What is Code Review

- Code review is a software quality assurance activity.
- This activity need person to check a program by viewing and reading source code.
- At least one reviewer who is not the code author.

Benefit of Code Review

- Share context with your mates
- Learn from others
- Find the potential issue from source code

Code Review Process

- Prepare
- Share code background and requirements
- Run test
- Explain
- Reviewers review and ask questions

Code Review Rule

- Learning by Challenge
- Learning by Sharing
- Focus on the code and commits
- Set time box per repo
- Summary and Record the Good Points and Questions

How to do Code Review

- Code review file by file

- Code review commit by commit

Demo: Code Review with Github

- Create Pull Request from your Repo to Organization Repo
- Visit Github Organization
  - https://github.com/AFS-Bootcamp-202211-Group-A
  - https://github.com/AFS-Bootcamp-202211-Group-B
- Find the Pull Request you will review
- Start code review
  - Give Comprehensive Comment
  - Give Code Comment

Practice Time

POS machine
Finish all features and test can pass. At least 6 function box in context map. At least 6 Git commits. No obvious duplicate code. No long method(>15 lines). Use meaningful names for variables, methods, classes. Context map and implementation code are consistent.

Let’s start our code review

AFS
Agile Full Stack
Developer Bootcamp
Thoughtworks @

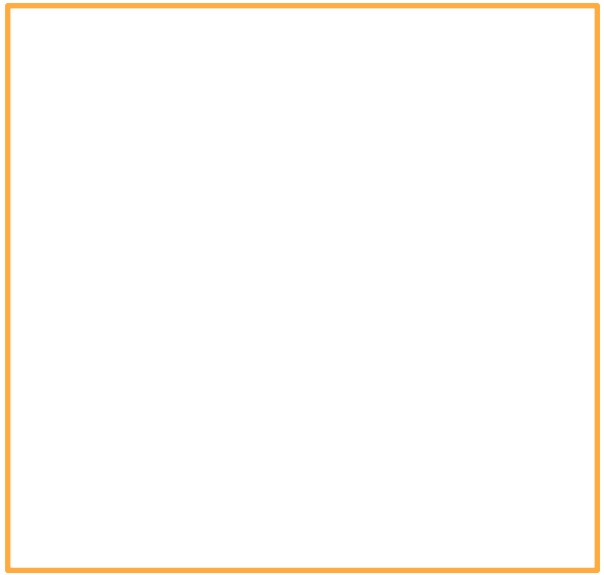
OOCL
We take it personally

Java Stream API

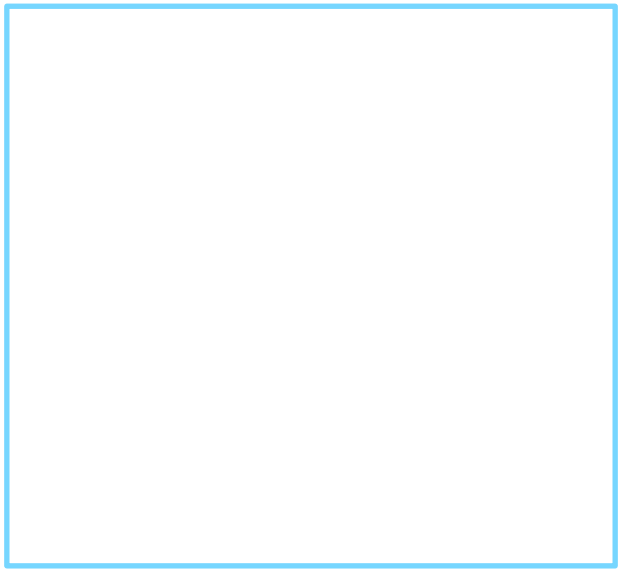
thoughtworks

Have you ever used Java Stream API?

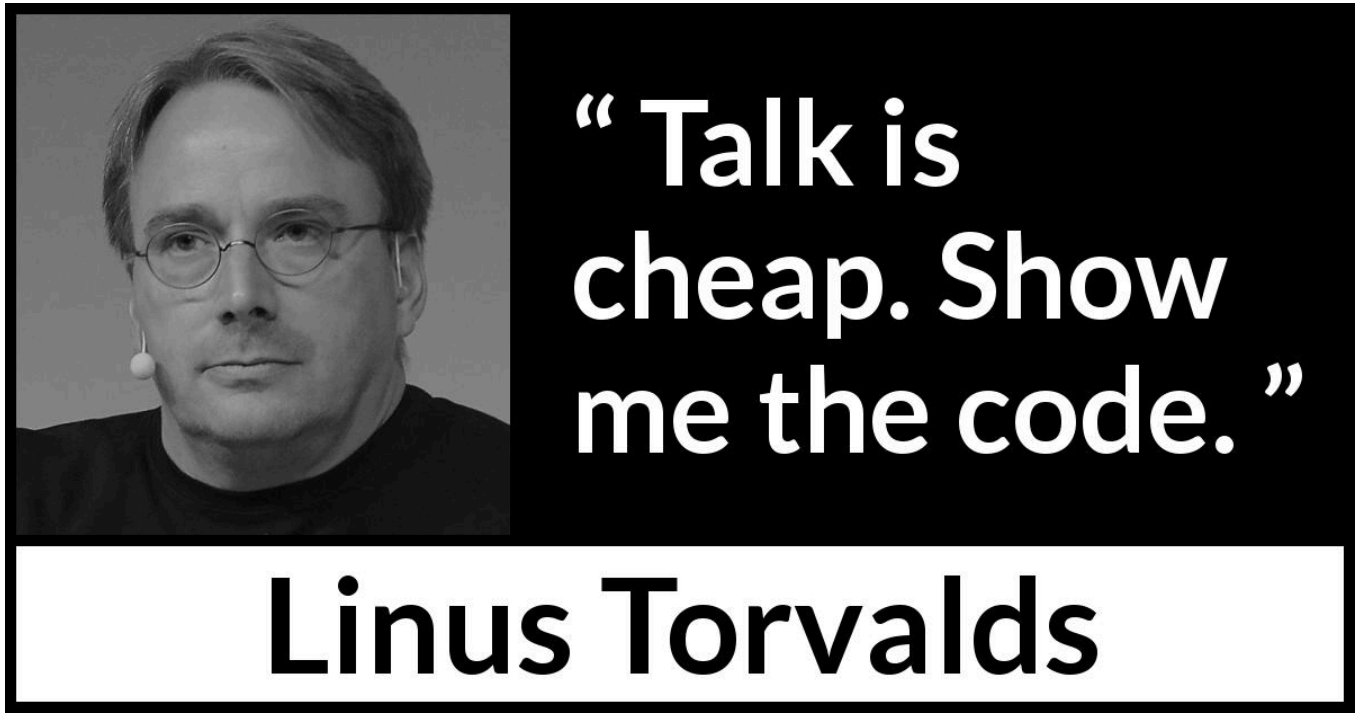
Yes



No

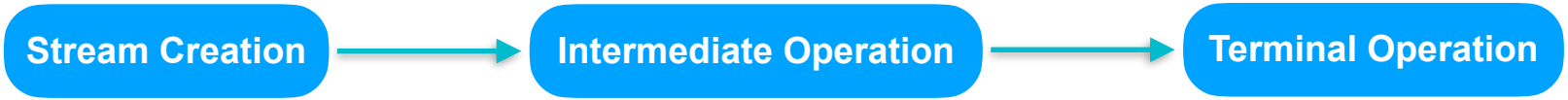


Java Stream API - Start from code



Java Stream API Syntax

- Stream Creation
  - Wrap your original data structure such as list, set
- Intermediate Operation (Optional)
  - Transform Stream to another Stream
- Terminal Operation
  - Convert Stream to another Type (List, Integer …)



Stream API example

```
return numbers
        .stream()
        .distinct()
        .collect(Collectors.toList());
```

Stream Creation **Intermediate Operation** **Terminal Operation**

Java Stream Filter

thoughtworks

Code Demo

Java Stream Filter

thoughtworks

Practice

Java Stream Filter — Summary

- Intermediate operation
- Find all the items that meet a certain condition.
- Filter needs a Lambda expression takes one parameter of T and return a boolean

Java Stream Map

thoughtworks

Code Demo

Java Stream Reduce(Optional)

Code Demo

Java Stream Map

thoughtworks

Practice

Java Stream Map — Summary

- Intermediate operation
- Map each element to its corresponding result
- Map needs a Lambda expression takes one parameter of T and return a result of R

Java Stream Reduce — Summary

- Terminal operation
- Produce one single result from a sequence of elements, by repeatedly applying a combining operation to the elements in the sequence.
- Reduce needs a Lambda expression takes two parameters

Java Stream Intermediate Operations

- concat()
- distinct()
- filter()
- flatMap()
- map()
- peek()
- skip()
- sorted()
- parallel()
- sequential()
- unordered()

Java Stream Terminal Operations

- allMatch()
- anyMatch()
- collect()
- count()
- findAny()
- findFirst()
- forEach()
- max()
- min()
- reduce()
- toArray()

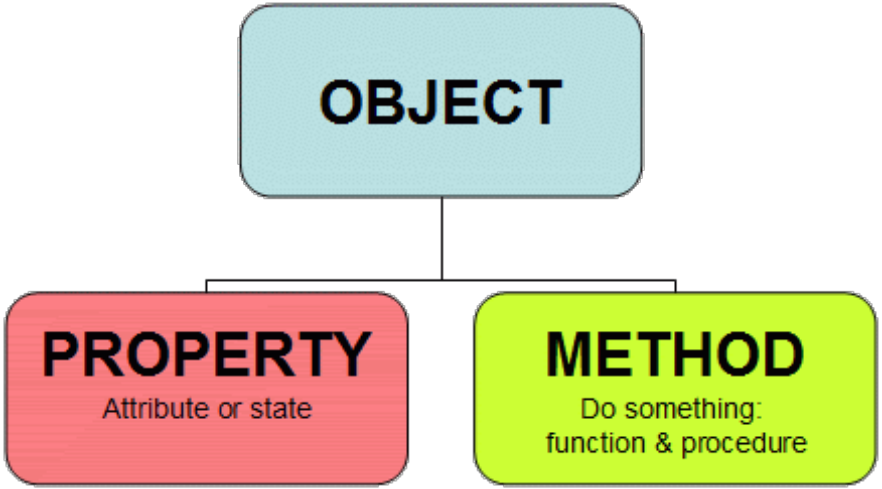
AFS
Agile Full Stack
Developer Bootcamp
Thoughtworks @

OOCL
We take it personally

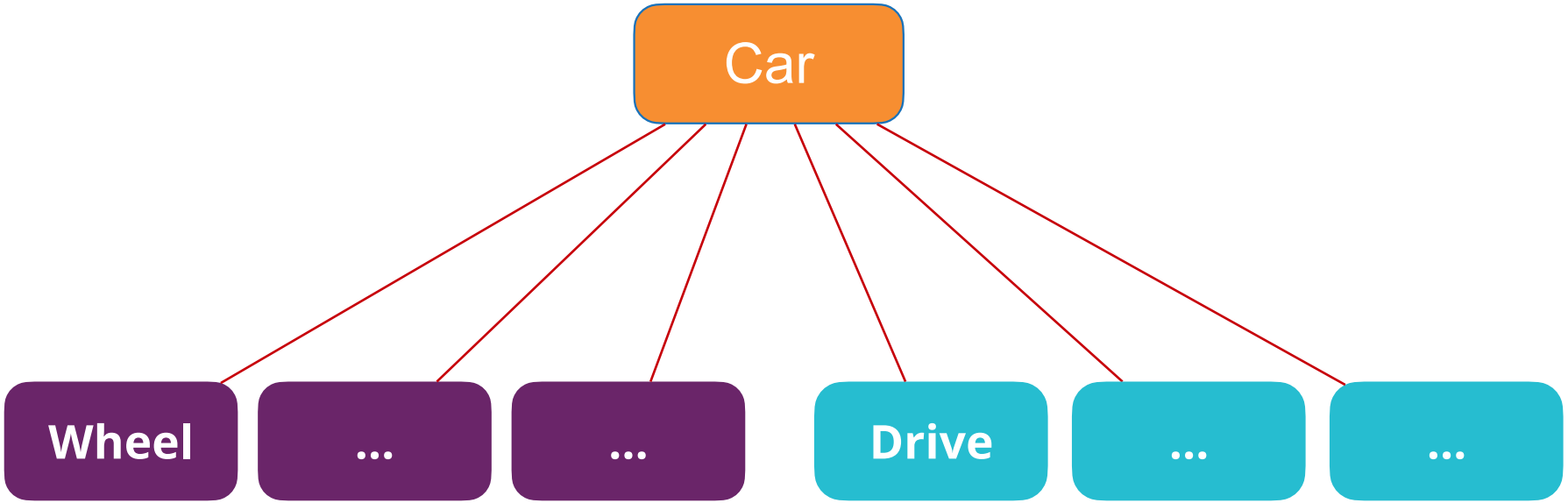
Java Object Oriented Programming

thoughtworks

What is Object



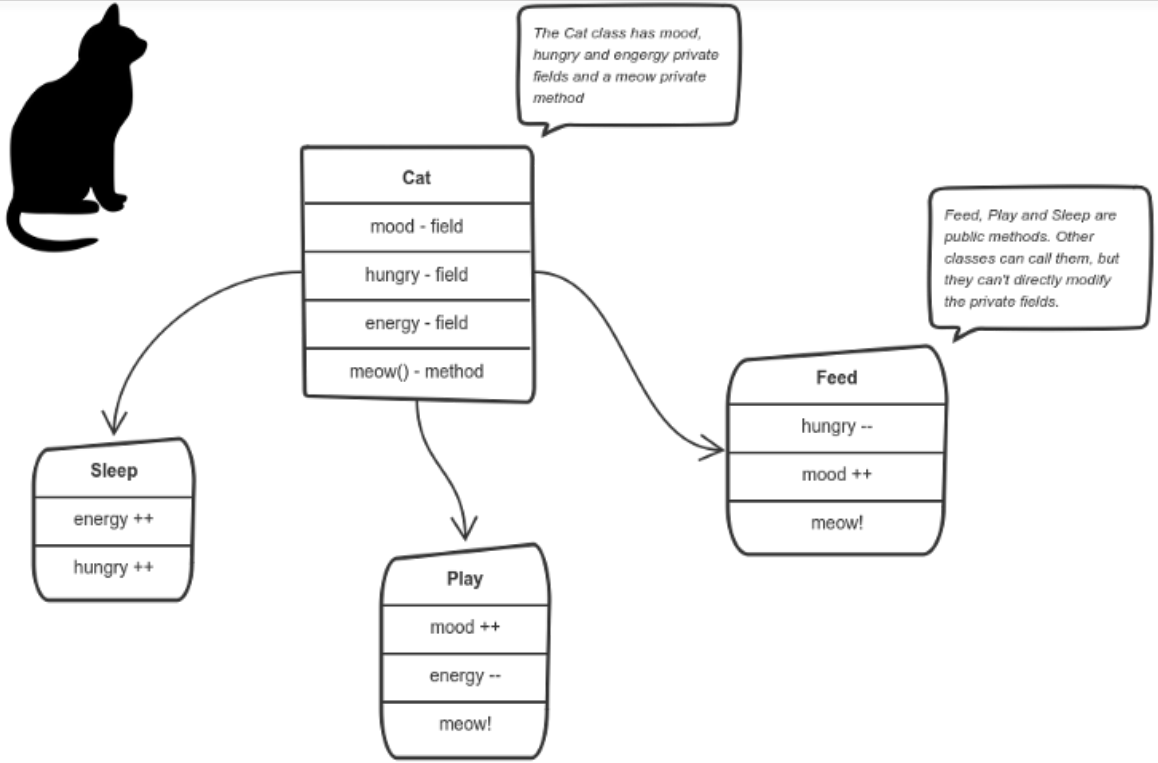
For Example



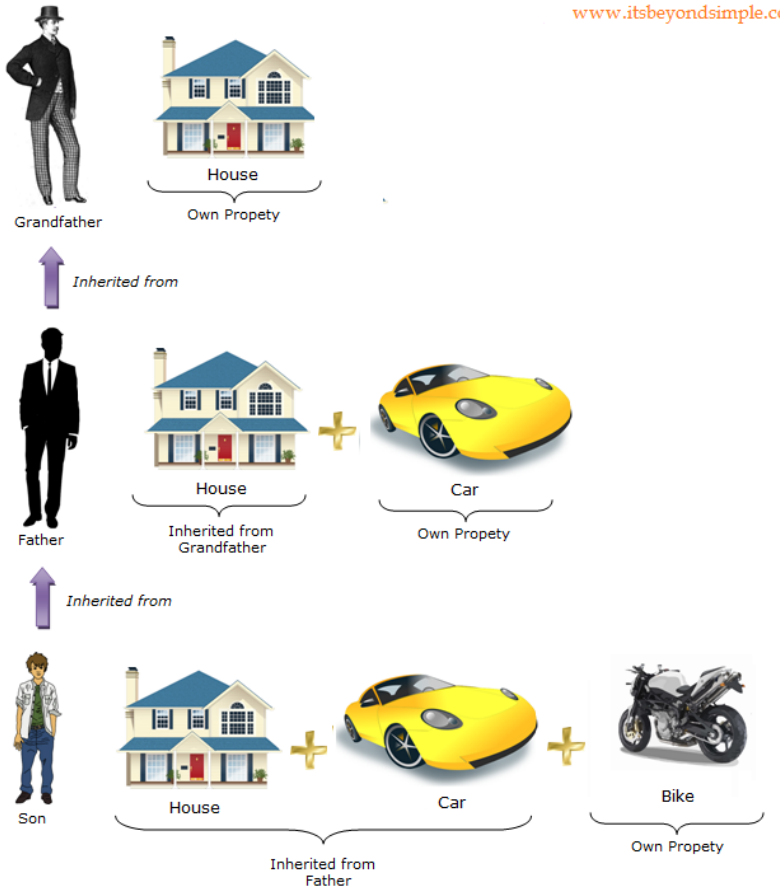
OO Features

- Encapsulation
- Inheritance
- Polymorphism

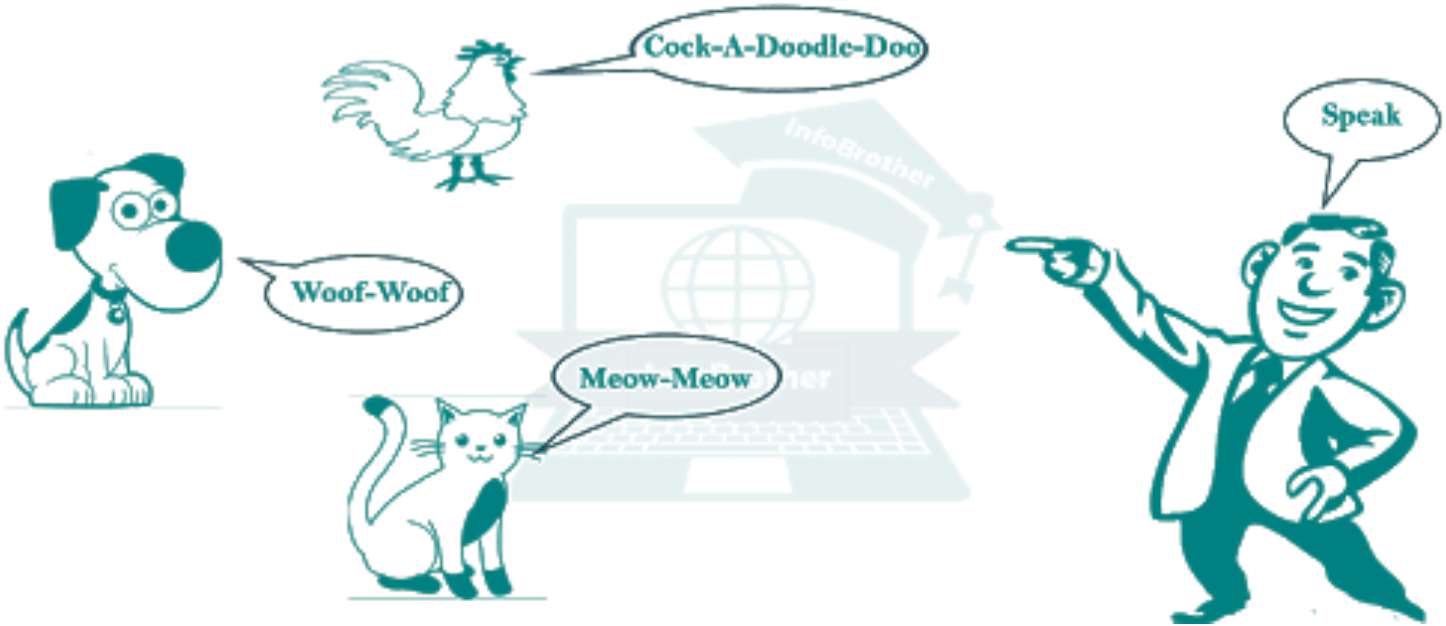
What's Encapsulation



What's Inheritance



What’s Polymorphism



Code Demo

Java OOP

- **AC1:** A car can show message with name and speed when speedup like: Cool Car: speed up 30 km/h
- **AC2:** A truck can show message with name and speed when speedup like: Big Truck: speed up 10 km/h
- **AC3:** A driver can drive multiple types of vehicles, and can show message with the name and speed when speedup:
  1. when drive a car the message is: Cool Car: speed up 30 km/h;
  2. when drive a truck the message is: Big Truck: speed up 10 km/h
- **AC4:** The speed of the car depends on the engine, with gasoline engine the speed is 30 km/h, and with an electric engine the speed is 25 km/h.

Practice

OOP Summary

OOP Feature

- Encapsulation
- Inheritance
- Polymorphism

OOP Advantage

- Encapsulate the complex logic inside the object, and avoid external modification of the private state of the object.
- Inheritance can maximize code reuse
- Polymorphism allows us to use the same method to deal with different objects.

OO Step by Step

Homework

Homework

1. **Daily Summary using ORID**
2. **Java OO Step by Step**

Submit Before 22:00

- **Complete step 1-6**
- Solving step 7 with Observer Pattern(Optional)
- Make sure **test cases can pass**, please don't change test cases
- Commit **at least once in one practice,** commit message format is **type: detail message**
- Submit your homework with your GitHub repository address

Extra points:
- No obvious access qualifier usage errors, e.g., no incorrect use of private, public
- No obvious duplicate code
- Use Java stream instead of for loop
- Use meaningful names for variables, methods