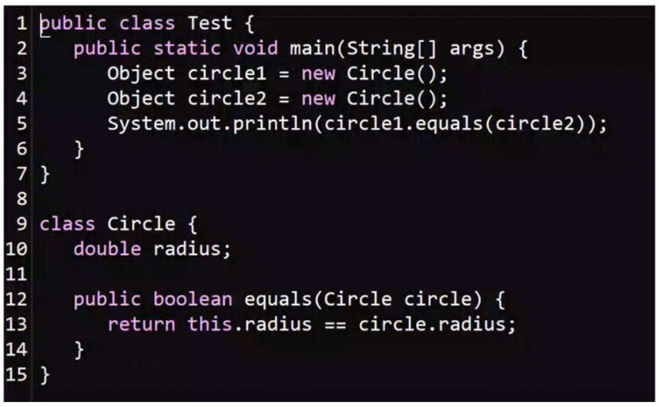
6419050024l Passakorn Puttawa



```
public class Test {
    public static void main(String[] args) {
        Object circle1 = new Circle();
        Object circle2 = new Circle();
        System.out.println(circle1.equals(circle2));
    }
}

class Circle {
    double radius;

    public boolean equals(Circle circle) {
        return this.radius == circle.radius;
    }
}
```

false beause Object didn't have method named radius

Connecting the data

- Coupling, cohesion → 2 widely-used quanlity metrics (among others)
- Abstraction, encapsulation, inheritance, polymorphism
  4 pillars of OOP → being used (observed)
- OO design principles → principles for designing better software
  often 4 pillars of OOP and OO quality to justify principles
- Design patterns → problem/solution pairs for recurring OO problems
  - Conceptual frame of approach to solve recurring problems
  - Language-independent
  - Favor delegation over inheritance
  - Often favor some desig quality over other

SOLID

S - Single Responsiblbility Principle (SRP)

→ A class should have one, and only one, reason to change.

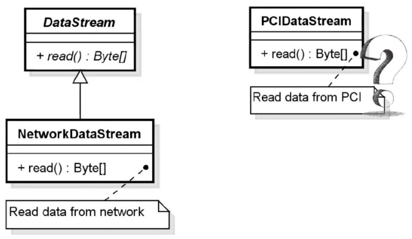
- SRP maks software easier to implement and prevents unexpected side-effects of future changes

O - Open-Closed Principle (OCP)

→ Software entilies (classes, modules, functions, etc.) should be open for extension, but closed for modification.

- You can't update the code the you have alreay written but you can add new code

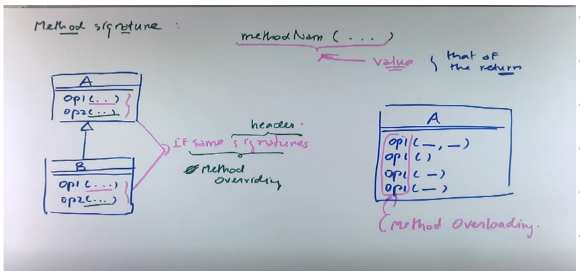
OCP through Inheritance



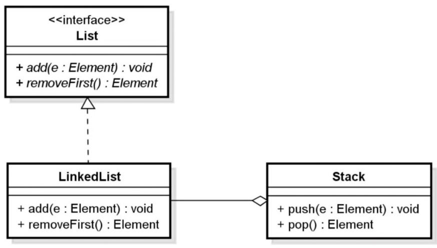
L - Liskov Substitution Principle (LSP)

→ Derived classes must be substitutable for their base classes

- An overridden method of a subclass must accept the same input parameter values as the method of the superclass



How to apply LSP if we have this relationship?

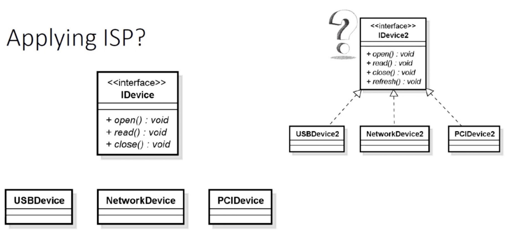

<<interface>>
List
+ add(e : Element) : void
+ removeFirst() : Element
LinkedList
+ add(e : Element) : void
+ removeFirst() : Element
Stack
+ push(e : Element) : void
+ pop() : Element

I - Interface Segregation Principle (ISP)

→ Clients should not be forced to depend upon interfaces that they do not use

- Similar to SRP, the goal ISP is to reduce the side effects and frequency of required changes
- Thogh straight forward, it is pretty easy to violate the ISP

Applying ISP?



What if all 3 devices are happy with the IDevice interface, except for the USB device that need another method *refresh*() ..

D - Dependency Inversion Principle (DIP)

→ High-level modules should not depend upon low-level modules. Both should depend upon abstractions.

- both should depend upon abstractions"
  - Each module should program to an interface!