The two interfaces

In presence of delta programming

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone

http://www.pharo.org
Outline

- Reminder: the essence of OOP
- One question
- Classes have two different kind of clients!
Back to the roots: Inheritance

• Needs:
  ○ Usually we want small adaptations to existing classes
  ○ We want to **reuse** existing behavior (not reimplement)

• Solution: class inheritance
Inheritance: expressing deltas

Inheritance is a reuse mechanism.
A class:

- does not reimplement the code of its superclasses
- extends the definition of its superclasses
  - add state
  - extends/specializes behavior
- expresses a delta i.e. differences to its superclasses
What are the consequences of the idiom: “Fields should be private”?

class A {
    private x ;

    void foo(){ ... x ...}
}

Consequences

- Clients cannot access \( \times \)
  - sounds good
- But, subclasses cannot access \( \times \) too
  - not ok because how can we express a delta?
  - copying the body of \( \text{foo} \) in subclasses to extend it manually is also impossible!
What are the clients of a class?

- Its users (e.g., Person is a client of Address)
- But also its subclasses i.e. its extenders
Extensibility?

- Think about your extenders
  - When writing a class, you cannot predict how it MUST be extended in 5 years from now!
- `final` and `private` prevent expressing deltas
  - `better use` `protected`
So, the correct idiom is...

To support both encapsulation and *extension*:

- Fields should be private **AND** the class should provide **protected** accessors

Or

- Fields should be **protected**
Benefits

- Clients cannot access your state (encapsulation)
- Subclasses can **extend/refine** the behavior of superclasses (extensibility)
Conclusion

- OOP is about encapsulation AND extension
- A class has always two kinds of clients:
  - its users
  - its extenders