Jin L.C. Guo

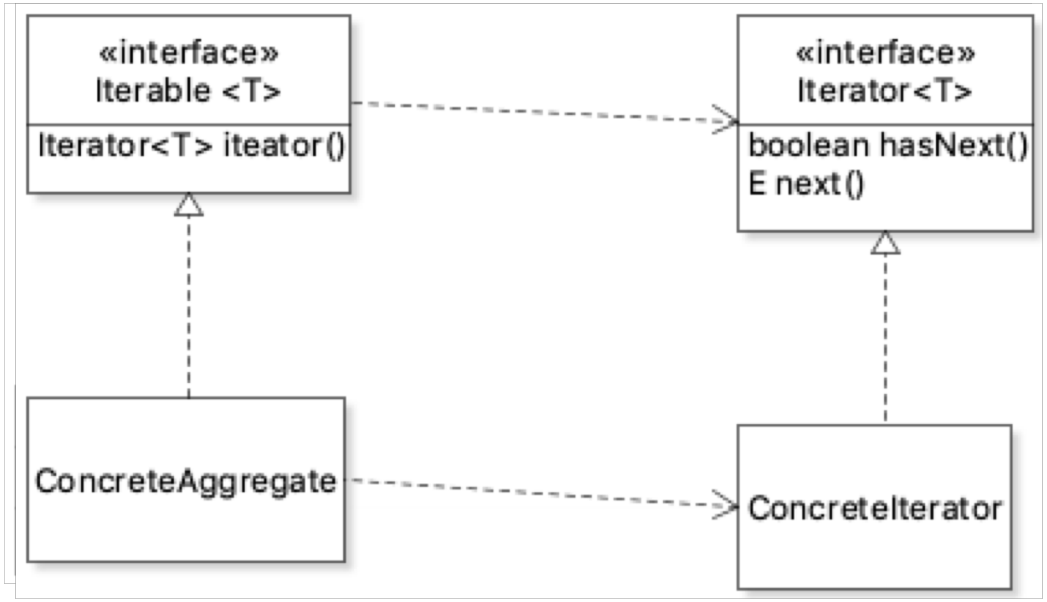
M2 (c) - Types and Polymorphism

Image Source: https://upload.wikimedia.org/wikipedia/commons/2/2b/Cepaea_nemoralis_active_pair_on_tree_trunk.jpg

Objective

- Iterator Design Pattern
- Strategy Design Pattern
- Interface Segregation Principle;

Iterator Design Pattern



Iterator Design Pattern

- Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation

Who controls the iteration?

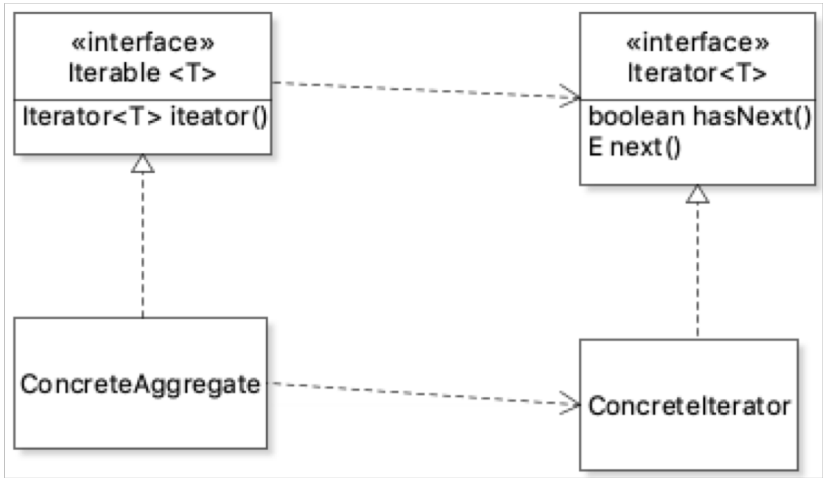
Client or iterator?

Who defines the algorithm?

Aggregate or iterator?

Is the iterator robust?

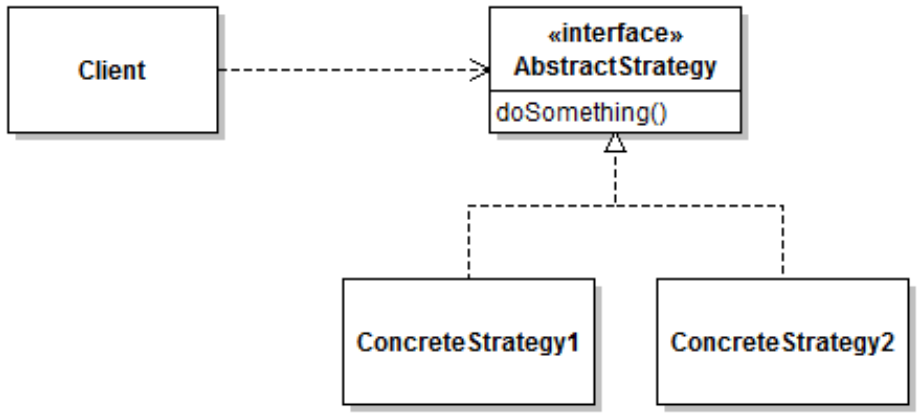
Handle Insertion or removal during iteration?



Demo on iterator pattern

Strategy Design Pattern

- Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

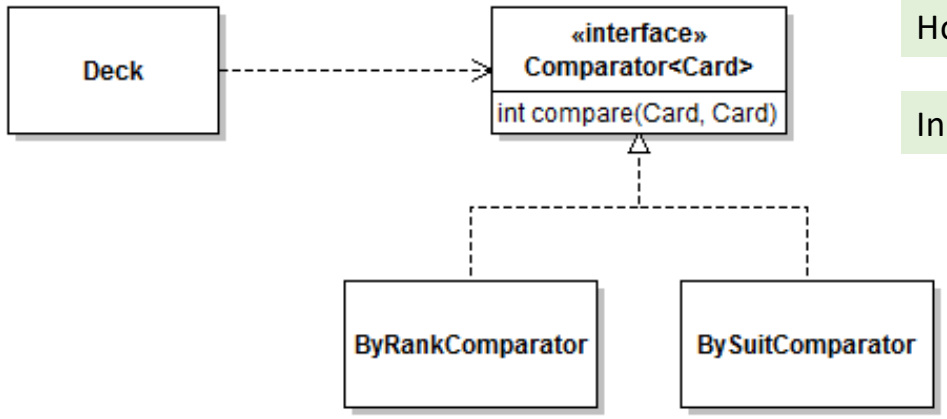


Algorithms are appropriate at different times

New Algorithms need to be introduced when necessary

Strategy Design Pattern

- Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.



How to provide the Strategy Context?

Input and output

Interface Segregation Principle

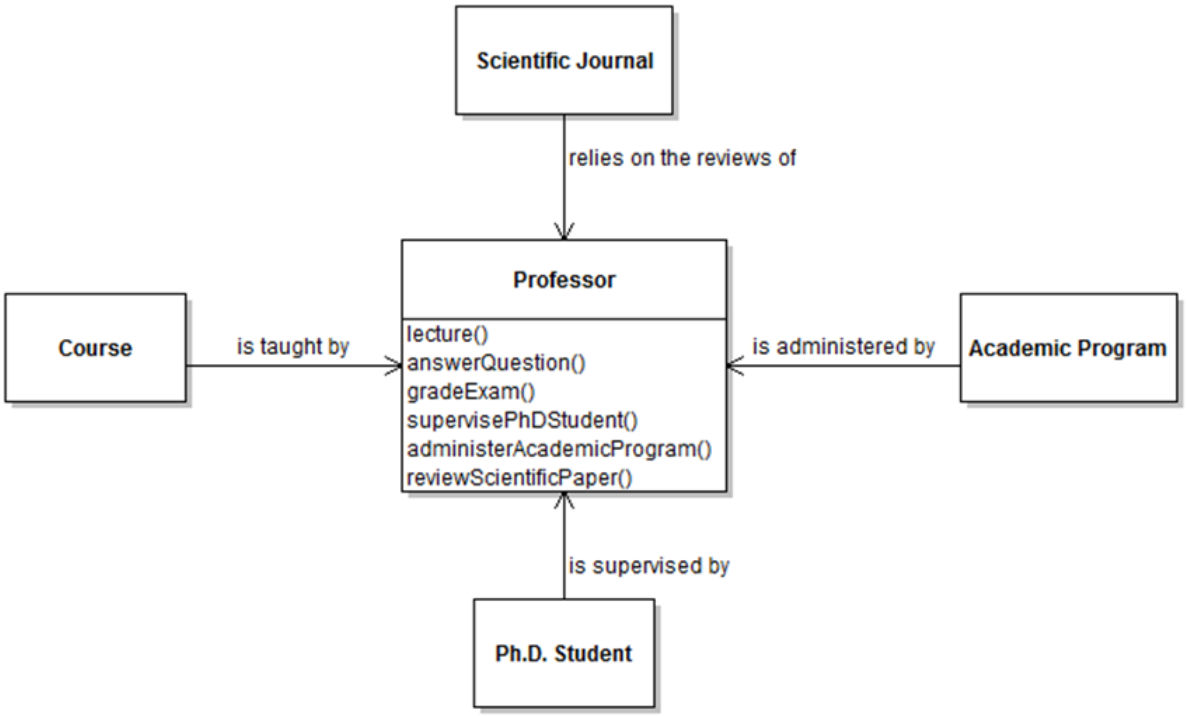
Clients should not be forced to depend on interfaces they do not need.

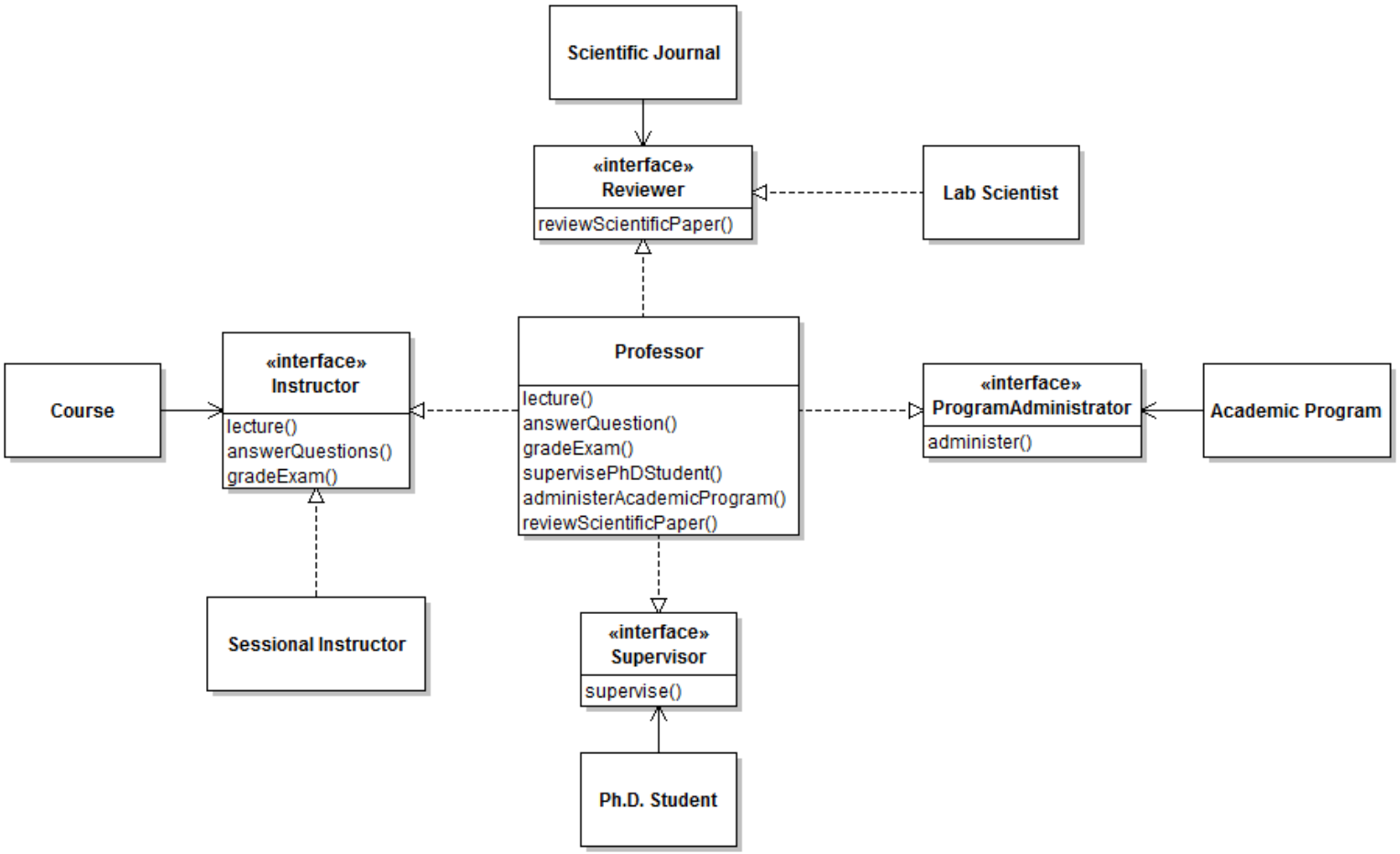
```
public final class Deck
{
  private  Stack<Card> aCards = new Stack<>();
  ...

  public Deck( Deck pDeck ) {...}

  public void shuffle() {...]
  public Card draw() {...}
  public boolean isEmpty() {...}
}
```

Interface Segregation Principle

```
public interface CardSource {
    /*
    * Remove a card from the source and return it.
    *
    * @return The card that was removed from the source
    * @pre !isEmpty()
    */
    Card draw();
    /*
    * @return True if there is not card in the source.
    */
    boolean isEmpty();
}
```

Interface Segregation Principle




Scientific Journal
«interface»
Reviewer
reviewScientificPaper()
Lab Scientist
Course
«interface»
Instructor
lecture()
answerQuestions()
gradeExam()
Professor
lecture()
answerQuestion()
gradeExam()
supervisePhDStudent()
administerAcademicProgram()
reviewScientificPaper()
«interface»
ProgramAdministrator
administer()
Academic Program
Sessional Instructor
«interface»
Supervisor
supervise()
Ph.D. Student

Acknowledgement

- Some examples are from the following resources:
  - *COMP 303 Lecture note* by Martin Robillard.
  - *The Pragmatic Programmer* by Andrew Hunt and David Thomas, 2000.
  - *Effective Java* by Joshua Bloch, 3rd ed., 2018.