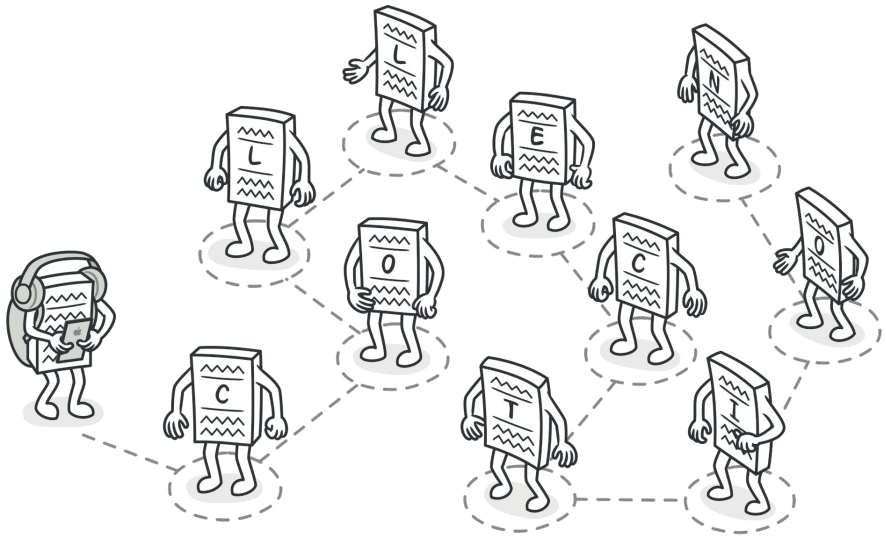
Iterator Design Pattern

Code Busters: Gavin D’mello, Kameron Jusseaume, Atefeh Rahmani

Iterator Background

- Object Behavioral Design Pattern
- Also known as the Cursor Design Pattern
- Allows for the traversal of elements of a collection without exposing the underlying representation
- Common in Object-Oriented systems

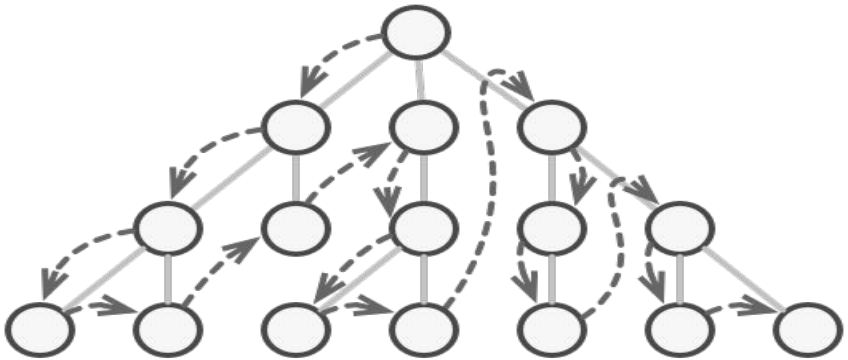


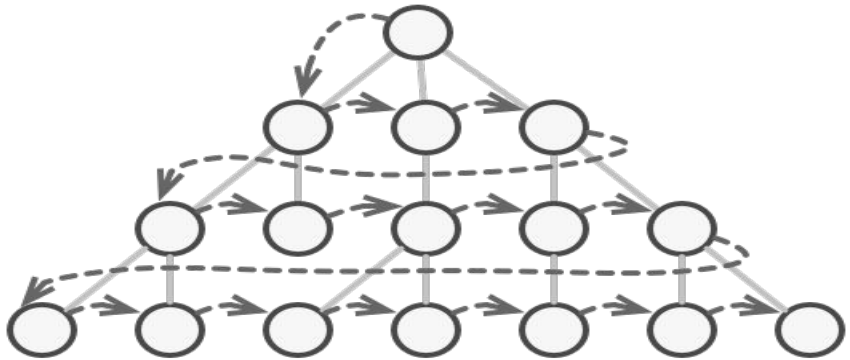
Related Design Patterns

- Composite
  - Iterators are often applied to recursive structures such as Composite
- Factory
  - Polymorphic iterators rely on factory methods to instantiate the appropriate Iterator subclass
- Memento
  - Iterator can use a Memento to capture the state of an iteration. The Iterator stores the Memento internally
- Visitor
  - Use with Iterator to traverse complex data structures and execute some operation over the elements even if they have different classes

Problem

- Collections must supply some way of accessing the elements so that the code can utilize them
- Adding more traversal algorithms to get the elements of a collection blurs its primary responsibility
- Some algorithms might be tailored for a specific application





Context

- An application that goes through each element of a collection without accessing the elements repeatedly
- Want to traverse the collection in different ways
- Should be a way to access the elements of a collection object in a sequential manner without needing to know the underlying representation
- A uniform interface for traversing many types of aggregate objects
- Reduce the duplication of the traversal code across a program

Consequences

Three important consequences:

1. Supports variations in the traversal of an aggregate, meaning complex aggregates may be traversed in many ways.
2. Iterators simplify the Aggregate interface,
3. More than one traversal can be pending on an aggregate

Iterator Design Pattern Pros and Cons

Pros

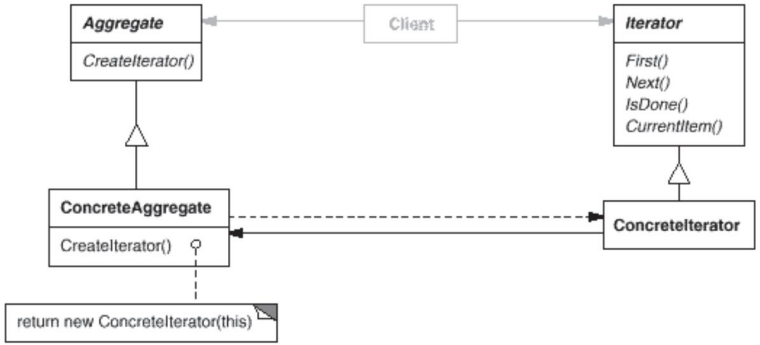
- Single Responsibility Principle
  - Allows for the cleanup of client code and aggregates/collections by extracting bulky traversal algorithms into separate classes
- Open/Closed Principle
  - Can implement new types of aggregates/ collections and iterators and pass them to existing code and not break anything
- Can iterate over the same collection in parallel because each object contains its own iteration state
- For the same reason, you can delay an iteration and continue when needed

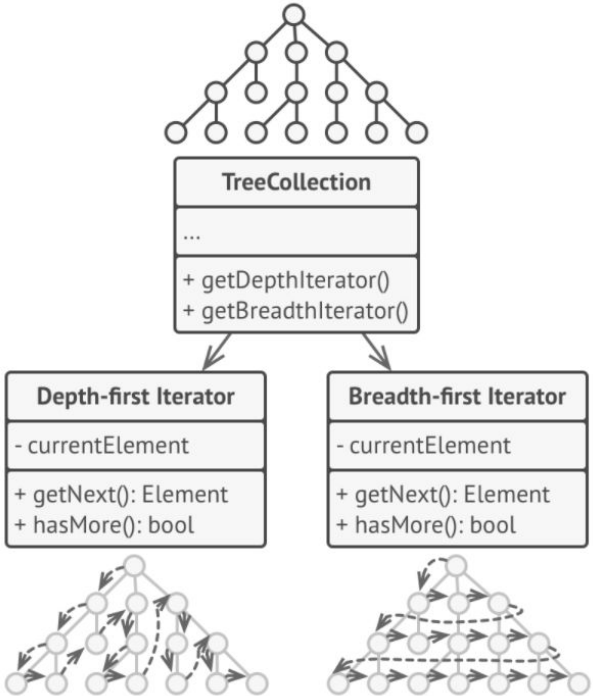
Cons

- Applying the pattern can be overkill if the application only works with simple collections
- Using the iterator may be less efficient that just going through the elements of a collection directly

Solution

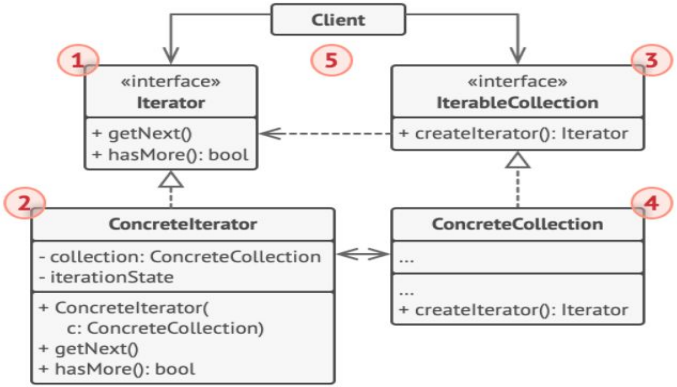
- Extract the traversal behavior of a collection into a separate object - Iterator
- Iterator object encapsulates all the traversal details
- Generalize the Iterator concept to support **Polymorphic Iteration**
- Allows for multiple traversals to occur at the same time independently
- Provides one primary method for fetching elements of the collection
- All iterators must implement the same interface



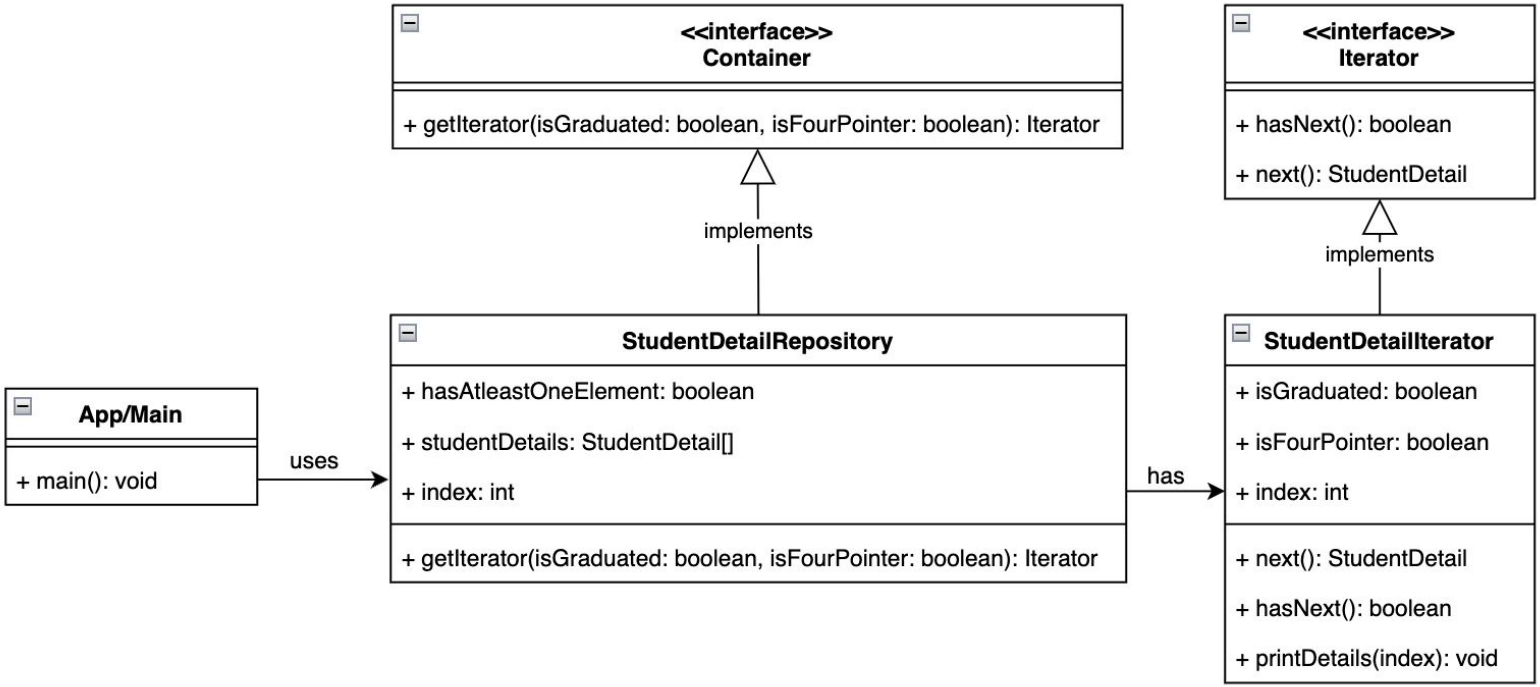


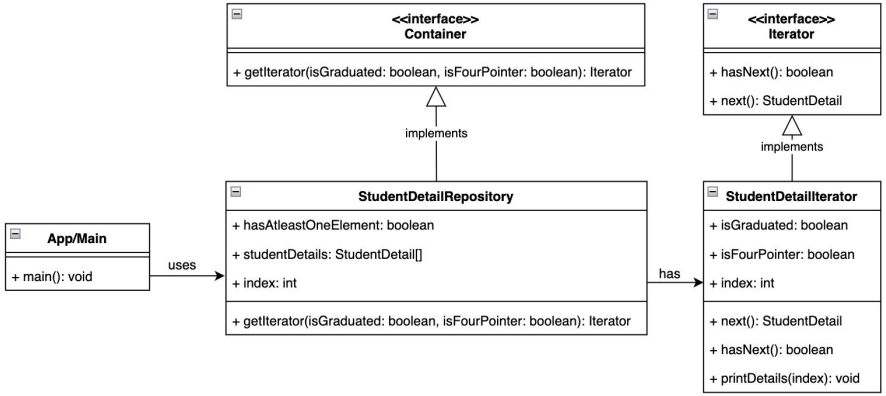
Participants and Collaborations

- **Participants**
  - **Iterator**
    - Defines an interface for accessing and traversing elements
  - **Concrete Iterator**
    - Implements the Iterator interface
    - Keeps track of the current position in the traversal of the aggregate
  - **Aggregate/Collection**
    - Defines an interface for creating an Iterator object
  - **Concrete Aggregate/Collection**
    - Implements the iterator creation interface to return an instance of the proper **Concrete Iterator**
  - **Client**
    - Works with both Aggregates and Iterators via the interfaces
    - Not coupled to concrete classes
- **Collaborations**
  - A **Concrete Iterator** keeps track of the current object in the aggregate and can compute the succeeding object in the traversal



Our Implementation






```java
public interface Container {
  public Iterator getIterator(
    boolean isGraduated,
    boolean isFourPointer
  );
}
```

```java
public interface Iterator {
  public boolean hasNext();
  public void next();
}
```

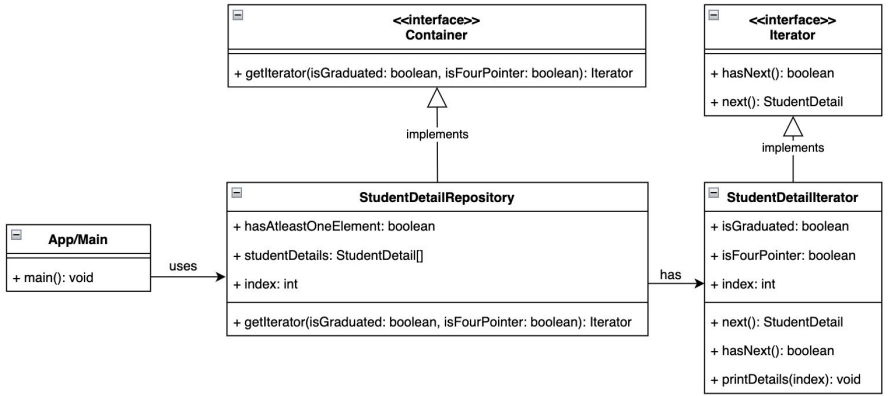
```java
class StudentDetail {
  String id;
  String name;
  boolean isGraduated;
  double GPA;

  StudentDetail(
    String id,
    String name,
    boolean isGraduated,
    double GPA
  ) {
    this.id = id;
    this.name = name;
    this.isGraduated = isGraduated;
    this.GPA = GPA;
  }
}
```

Main class

src > App.java > ...

```
49
50      StudentDetailRepository studentDetailRepository = new StudentDetailRepository();
51
52      for (
53        Iterator iter = studentDetailRepository.getIterator(
54          graduated,
55          fourPointer
56        );
57        iter.hasNext();
58      ) {
59        iter.next();
60      }
61    }
62  }
```



```
public class StudentDetailRepository implements Container {
  private boolean hasAtleastOneElement = false;
  public StudentDetail[] studentDetails = DataSet.getStudentList();
  //   public StudentDetail[] studentDetails = DataSet.studentDetails;

  @Override
  public Iterator getIterator(boolean isGraduated, boolean isFourPointer) {
    return new StudentDetailIterator(isGraduated, isFourPointer);
  }
```

```
 private class StudentDetailIterator implements Iterator {
   int index;
   boolean isGraduated, isFourPointer;

   StudentDetailIterator(boolean isGraduated, boolean isFourPointer) {
     this.isGraduated = isGraduated;
     this.isFourPointer = isFourPointer;
   }
```

```java
@Override
public boolean hasNext() {
  if (index < studentDetails.length) {
    if (
      studentDetails[index].isGraduated == isGraduated &&
      !hasAtleastOneElement
    ) {
      System.out.println("ID          Name                    GPA");
      hasAtleastOneElement = true;
    } else if (
      index == studentDetails.length - 1 && !hasAtleastOneElement
    ) {
      System.out.println("No students found");
      hasAtleastOneElement = true;
    }
    return true;
  }
  return false;
}
```

```
@Override
public void next() {
  if (this.hasNext()) {
    if (isGraduated || isFourPointer) {
      if (
        isGraduated && isFourPointer && studentDetails[index].isGraduated == true &&
        studentDetails[index].GPA == 4.0
      ) {
        print(index);
      } else if (
        !isGraduated && isFourPointer && studentDetails[index].GPA == 4.0
      ) {
        print(index);
      } else if (
        isGraduated && !isFourPointer && studentDetails[index].isGraduated == true
      ) {
        print(index);
      }
    } else {
      print(index);
    }
    index++;
  }
}
```

Demo

Sources

https://www.tutorialspoint.com/design_pattern/iterator_pattern.htm

https://refactoring.guru/design-patterns/iterator

https://sourcemaking.com/design_patterns/iterator

https://learn-us-east-1-prod-fleet02-xythos.content.blackboardcdn.com/5f5fe08e84e1e/2818733?X-Blackboard-Expiration=1649030400000&X-Blackboard-Signature=5rBhdTFleUdct8H81GiAMK6hHWsryDpiNWLjT1CBrqY%3D&X-Blackboard-Client-Id=787702&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27Iterator.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEMz%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQDO9Tlidxa1an2qEdnYeMBKYQAsj5hcJM6ljSBYEEyUjwIhAKp%2BwoCCJw2nztRu1FPUWOrMNjOkLDLBruFZzFdAumagKvoDCGUQAhoMNjM1NTY3OTI0MTgzIgwCvmv9iMfl57lMQgoq1wM3f8riXpij9NtzM7kVSHvjG0xyFKComfGXZ4z27axQCRi7fDH%2Fj%2BjxFeVIADVF0%2BN378nLQVf%2BrmIdSGqlkRL1EQWD8zLFZAJAfvpkTK3PLDY2V%2BRTnt0cZXCrv0mGJZdr3hz1z3o4h2zriQ%2BtAb%2F9Era%2FKoO7h%2B2FCfrZsA60A%2F%2BkXy8I1c4blmOomkUh3jJwr0wxT%2FCxa4Un2InBlPOpPoYYTbTB1HkdNaUjq0dS%2FvuILUYN%2BtifIN2tV658IwNZH17CE%2F2XImZ1hWhTJ8uWmdRpeq1vh0rCdBQwFb6CXf1sfDx3i72RvXEZJqZtLlKxfYmvCvMQ8J7nD%2F4MNPCcyq%2FQwGd8dSWdK8cusyq7dv%2B0VvjRFHscE4BwsjwLXQ1xCKlfDiYf5LYtqNmBDyv6wkrNGVu7hXCyUcrpdHFCCyDZmXN1O4EgDMI9moU0mXFaFrMVgAT03aTqQEC5ouaookFUrlNVM8FDMmZMTCgbFjP3RJtnpfusRmn2naT%2BCzy3yaF33tmh%2F30o26N7me%2F29TiF8BxE5vBmI6X9oQPI3VlxOTWfCXSWkx%2BgHl0YPSkx0oteesBnsoCIB%2FfIB6u4quGj%2BYNqWW8Ok1tK01MRzXZk20gBWZ8wsPunkgY6pAEevTZak1%2Fw6AbEY%2FOWAw2XvmzbmEcn%2BcTK%2BTTul8vHe2jjZ0BxMuKnOcnT9r0Zdc9tgq%2B3jV8tBD7RTg16vOdRlkoNjAFuY8XkVOh%2FMchfmUzoP4TKrJlc7K0ibZ%2FHP%2FJg3GOUmIKiNIjxafamoX7UajN%2FXVukz9mOLCBjE8%2BgYeCJ4OmavNZYBAltRxG07E7hGVzjqvNcYD3D1nd8mrvBdWTgyg%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20220403T180000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAZH6WM4PL4BKJ7RHB%2F20220403%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=58f0dd678e38cc6e994a45924aa758a37c690cd45652b5d1219a1cac84741119