Exemple

Gestion d'une Bibliothèque avec le Pattern Iterator

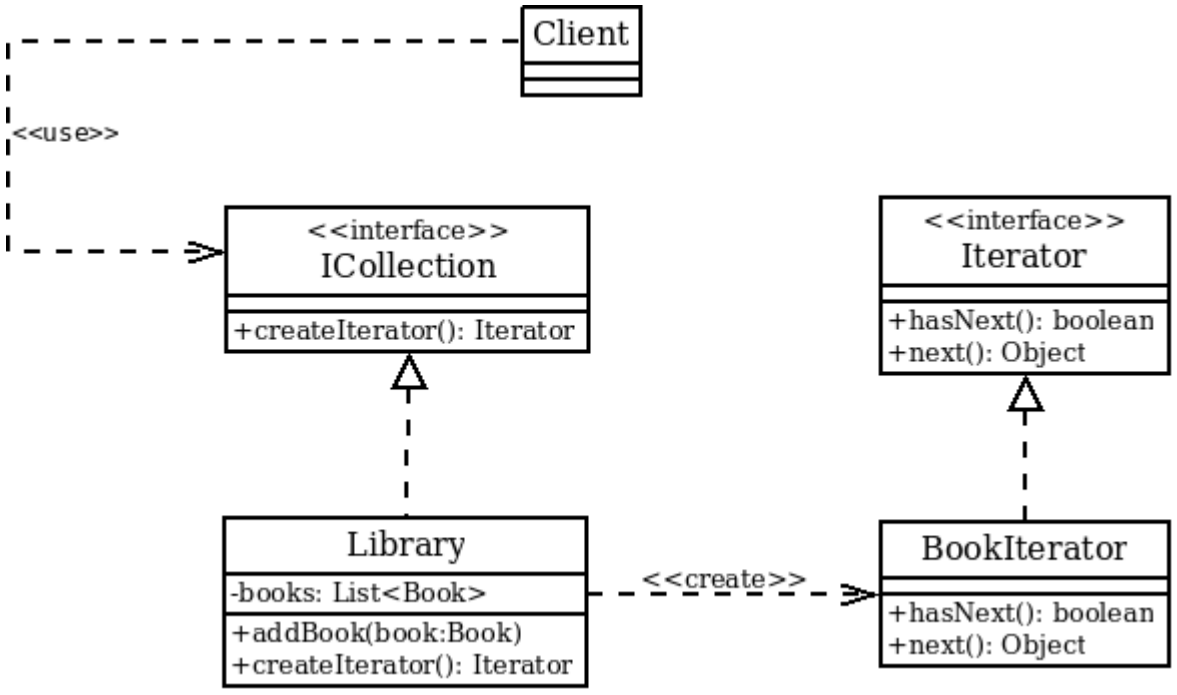
Contexte : Vous travaillez sur un projet de gestion de bibliothèque, où vous devez mettre en place une structure permettant de stocker des livres et de fournir une manière générique de parcourir cette collection.

Objectif : Concevoir un système de gestion de bibliothèque utilisant le pattern Iterator afin de permettre une itération facile et indépendante de la représentation interne de la collection de livres.

Contraintes :

- La solution doit utiliser le pattern Iterator pour assurer une itération propre et indépendante de la structure interne de la collection.
- Les classes doivent respecter le principe de séparation des préoccupations, assurant une bonne modularité et réutilisabilité du code.

Modélisation



Code source

ICollection.java

```java
public interface ICollection {
    public Iterator createIterator();

    // ------------------------------------
    public void addBook(Book book);
}
```

Library.java

```java
import java.util.ArrayList;
import java.util.List;

public class Library implements ICollection {
    private List<Book> books;

    public Library() {
        this.books = new ArrayList<>();
    }

    public void addBook(Book book) {
        books.add(book);
    }

    @Override
    public Iterator createIterator() {
        return new BookIterator(books);
    }
}
```

Iterator.java

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

BookIterator.java

```java
import java.util.List;

public class BookIterator implements Iterator {
    private List<Book> books;
    private int position;

    public BookIterator(List<Book> books) {
        this.books = books;
        this.position = 0;
    }

    @Override
    public boolean hasNext() {
        return position < books.size();
    }

    @Override
    public Object next() {
        if (hasNext()) {
            Book book = books.get(position);
            position++;
            return book;
        }
        return null;
    }
}
```

Book.java

```java
public class Book {
    private String title;

    public Book(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }
}
```

Client.java

```java
public class Client {
    Run | Debug
    public static void main(String[] args) {

        /**
         * Utilisation de l'iterateur provenant d'une liste
         * ----------------------------------------
         */
        ICollection library = new Library();

        // Ajout de quelques livres à la bibliothèque
        library.addBook(new Book(title:"Livre 1"));
        library.addBook(new Book(title:"Livre 2"));
        library.addBook(new Book(title:"Livre 3"));

        // Obtention de l'itérateur pour parcourir la collection de livres
        Iterator iterator = library.createIterator();

        // Parcours de la collection à l'aide de l'itérateur
        while (iterator.hasNext()) {
            Book book = (Book) iterator.next();
            System.out.println("Titre : " + book.getTitle());
        }

    }
}
```