Design Pattern
Iterator

```
public class Book {
    private String name;

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

    public String getName() {
        return name;
    }
}

public interface IIterator {
    boolean hasNext();
    Object next();
}

public interface IAggregate {
    IIterator iterator();
}

public class BookShelf implements IAggregate {
    private Book[] books;
    private int maximumIndex = 0;

    public BookShelf(int maximumSize) {
        books = new Book[maximumSize];
    }

    public Book getBookAt(int index) {
        return books[index];
    }

    public void appendBook(Book book) {
        books[maximumIndex] = book;
        maximumIndex++;
    }

    public int getLength() {
        return maximumIndex;
    }

    public IIterator iterator() {
        return new BookShelfIterator(this);
    }
}
```

```
public class BookShelfIterator implements IIterator {
    private BookShelf bookShelf;
    private int index;

    public BookShelfIterator(BookShelf bookShelf) {
        this.bookShelf = bookShelf;
        this.index = 0;
    }

    public boolean hasNext() {
        return index < bookShelf.getLength();
    }

    public Object next() {
        Book book = bookShelf.getBookAt(index);
        index++;
        return book;
    }
}

public class Application {
    public static void main(String... args) {
        BookShelf bookShelf = new BookShelf(4);

        bookShelf.appendBook(new Book("Book B"));
        bookShelf.appendBook(new Book("Book A"));
        bookShelf.appendBook(new Book("Book C"));
        bookShelf.appendBook(new Book("Book D"));

        IIterator iterator = bookShelf.iterator();

        while (iterator.hasNext()) {
            Book book = (Book)iterator.next();
            System.out.println(book.getName());
        }
    }
}
```