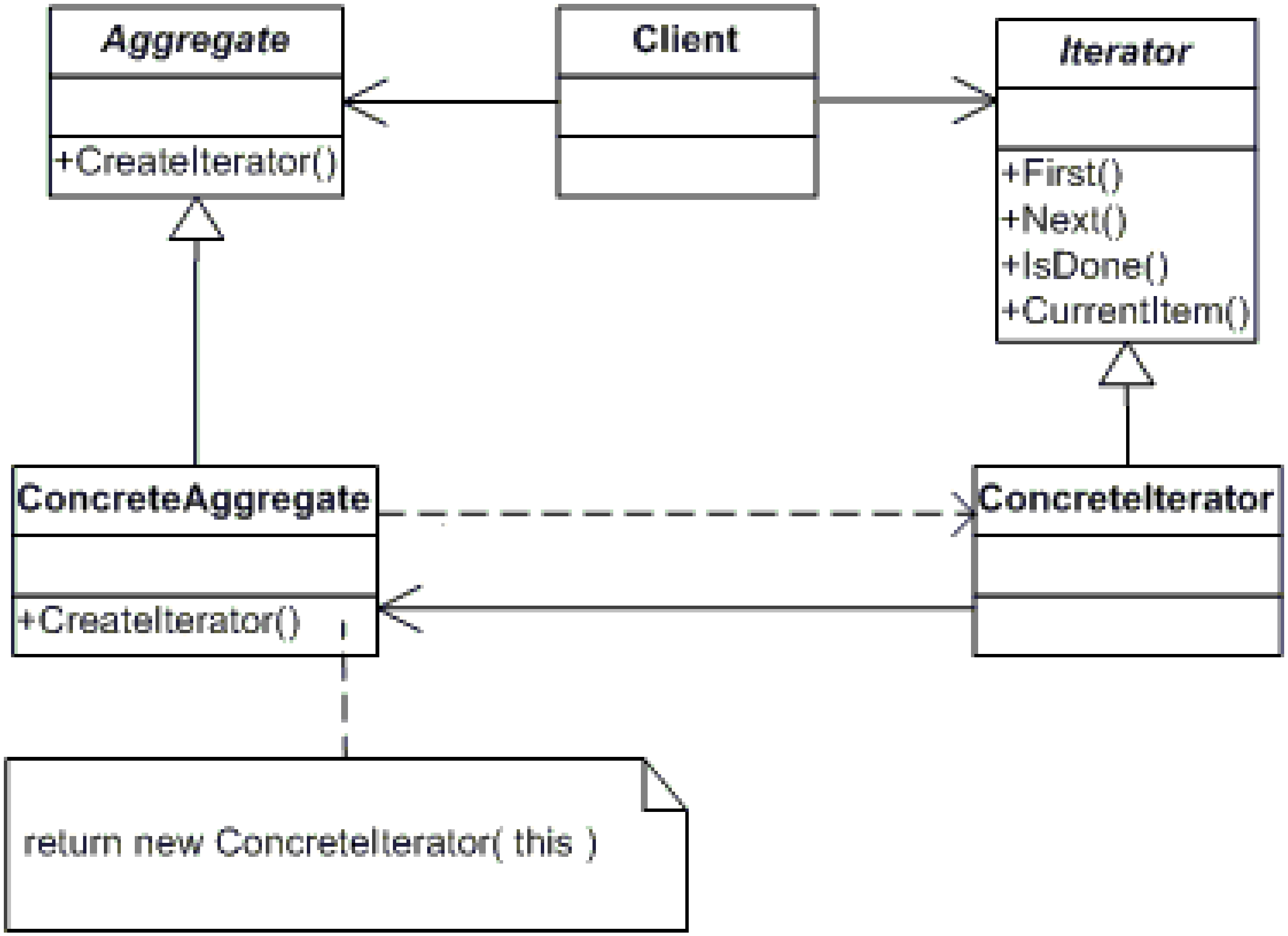
Motivation – Page 283~284, Jia Book

- iterators are used to access the elements of an aggregate object sequentially without exposing its underlying representation.
- An Iterator object encapsulates the internal structure of how the iteration occurs.


Aggregate
+CreateIterator()
Client
Iterator
+First()
+Next()
+IsDone()
+CurrentItem()
ConcreteAggregate
+CreateIterator()
ConcreteIterator
return new ConcreteIterator(this)

Participant of the Iterator Pattern

- Iterator
- ConcreteIterator
- AbstractCollection
- ConcreteCollection

```
public interface Aggregate {
    public abstract Iterator iterator();
}
```

```
public class Book {
    private String name;
    public Book(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}
```

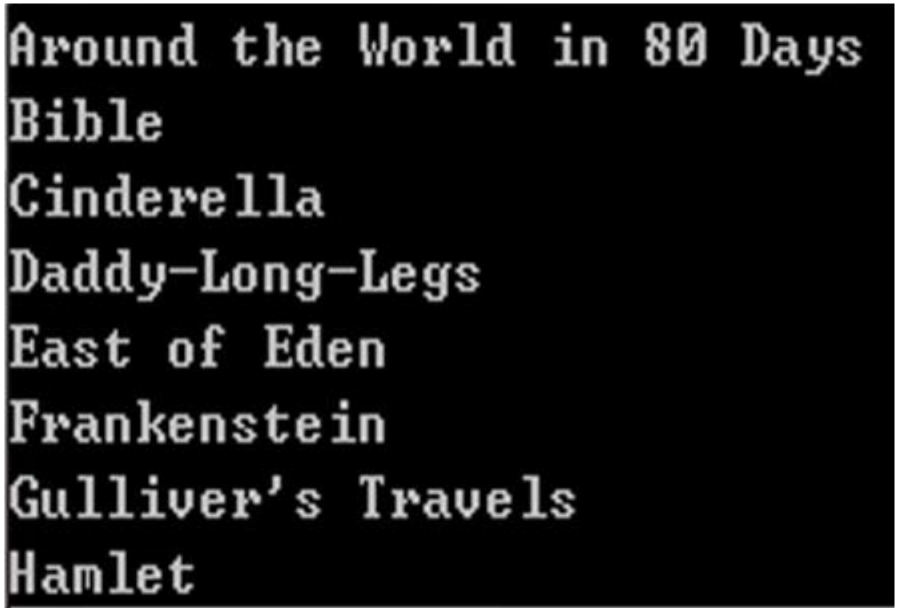
```java
import java.util.ArrayList;
public class BookShelf implements Aggregate {
    private ArrayList books;
    public BookShelf(int initialsize) {
        this.books = new ArrayList(initialsize);
    }
    public Book getBookAt(int index) {
        return (Book)books.get(index);
    }
    public void appendBook(Book book) {
        books.add(book);
    }
    public int getLength() {
        return books.size();
    }
    public Iterator iterator() {
        return new BookShelfIterator(this);
    }
}
```

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

```
public class BookShelfIterator implements Iterator {
    private BookShelf bookShelf;
    private int index;
    public BookShelfIterator (BookShelf bookShelf) {
        this.bookShelf = bookShelf;
        this.index = 0;
    }
```

```
  public boolean hasNext() {
         if (index < bookShelf.getLength()) {
              return true;
         } else {
              return false;
         }
    }
    public Object next() {
         Book book = bookShelf.getBookAt(index);
         index++;
         return book;
    }
}
```

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        BookShelf bookShelf = new BookShelf(4);
        bookShelf.appendBook(
                new Book("Around the World in 80 Days"));
        bookShelf.appendBook(new Book("Bible"));
        bookShelf.appendBook(new Book("Cinderella"));
        bookShelf.appendBook(
                new Book("Daddy-Long-Legs"));
        bookShelf.appendBook(
                new Book("East of Eden"));
        bookShelf.appendBook(
                new Book("Frankenstein"));
        bookShelf.appendBook(new Book("Gulliver's Travels"));
        bookShelf.appendBook(new Book("Hamlet"));
        Iterator it = bookShelf.iterator();
        while (it.hasNext()) {
            Book book = (Book)it.next();
            System.out.println(book.getName());
        }
    }
}
```



```
Around the World in 80 Days
Bible
Cinderella
Daddy-Long-Legs
East of Eden
Frankenstein
Gulliver's Travels
Hamlet
```

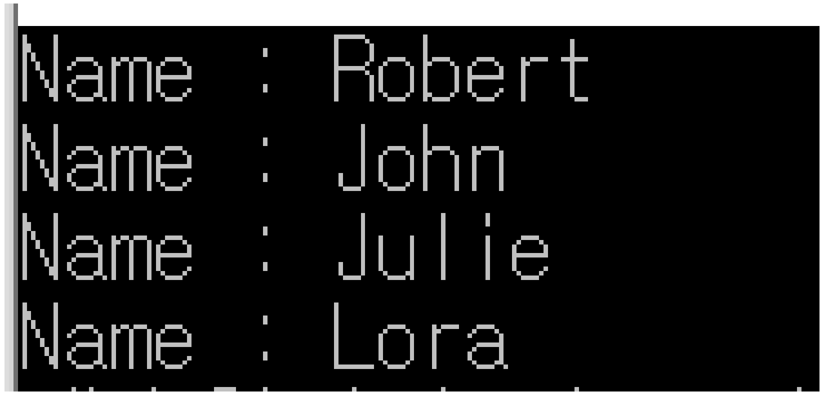
```
public interface Iterator {
   public boolean hasNext();
   public Object next();
}
```

```
public interface Container {
   public Iterator getIterator();
}
```

```
public class NameRepository implements Container {
   public String names[] = {"Robert" , "John" ,"Julie" , "Lora"};
   @Override
   public Iterator getIterator() {
      return new NameIterator();
   }
   private class NameIterator implements Iterator {
      int index;
      @Override
      public boolean hasNext() {
         if(index < names.length){
            return true;
         }
         return false;
      }
```

```
      @Override
      public Object next() {
         if(this.hasNext()){
            return names[index++];
         }
         return null;
      }
   }
}
```

```
public class IteratorPatternDemo {
   public static void main(String[] args) {
      NameRepository namesRepository =
                  new NameRepository();
      for(Iterator iter =
         namesRepository.getIterator(); iter.hasNext();){
         String name = (String)iter.next();
         System.out.println("Name : " + name);
      }
   }
}
```



```
Name : Robert
Name : John
Name : Julie
Name : Lora
```