DESIGN(ANTI-)PATTERNS

Coffee Break

Iterator

WHAT ARE DESIGN PATTERNS

A software design pattern is a general, reusable solution to a commonly occurring problem. It is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

Types

- Creational (singleton, builder, lazy init, abstract factory, object pool etc.)
- Structural (adapter, bridge, decorator, facade, composite, proxy etc.)
- Behavioral (**iterator**, strategy, template method, visitor, command etc.)
- Concurrency (thread pool, double checked locking, monitor object, reactor etc.)

ITERATOR

ITERATOR

GoF Definition

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

Concept

- It is often used to traverse the nodes of a tree-like structure. So, in many scenarios, you may notice the use of iterator patterns with composite patterns.
- The role of an iterator is not limited to traversing. This role can vary to support various requirements.
- Clients cannot see the actual traversal mechanism. A client program only uses the iterator methods that are public in nature.

ITERATOR

Design pattern methods

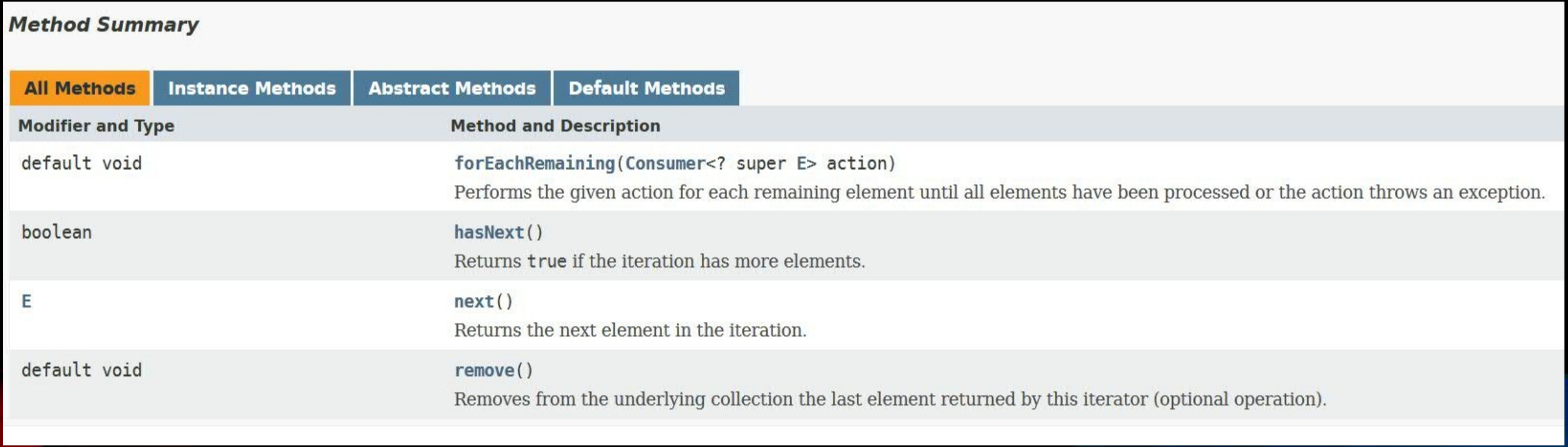
- Object first()
- Object next()
- boolean isDone()
- Object currentItem()

Java reality

```
public interface Enumeration {

    public boolean hasMoreElements();
    public Object  nextElement();
}

public interface Iterator<E> {

    public boolean hasNext();
    public E  next();
}
```

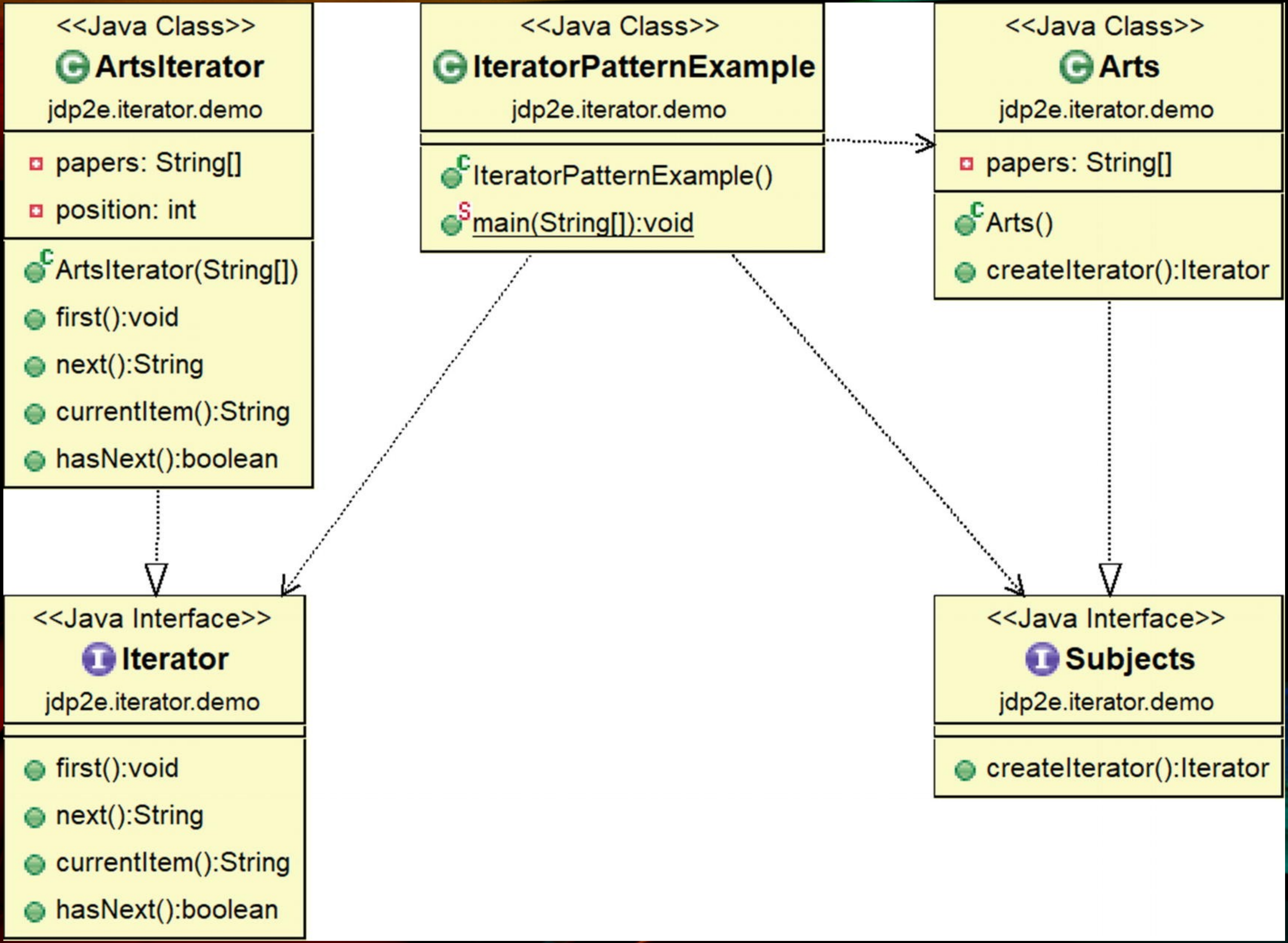
ITERATOR

java.util.Iterator



Modifier and Type	Method and Description
`default void`	`forEachRemaining(Consumer<? super E> action)` Performs the given action for each remaining element until all elements have been processed or the action throws an exception.
`boolean`	`hasNext()` Returns `true` if the iteration has more elements.
`E`	`next()` Returns the next element in the iteration.
`default void`	`remove()` Removes from the underlying collection the last element returned by this iterator (optional operation).

ITERATOR



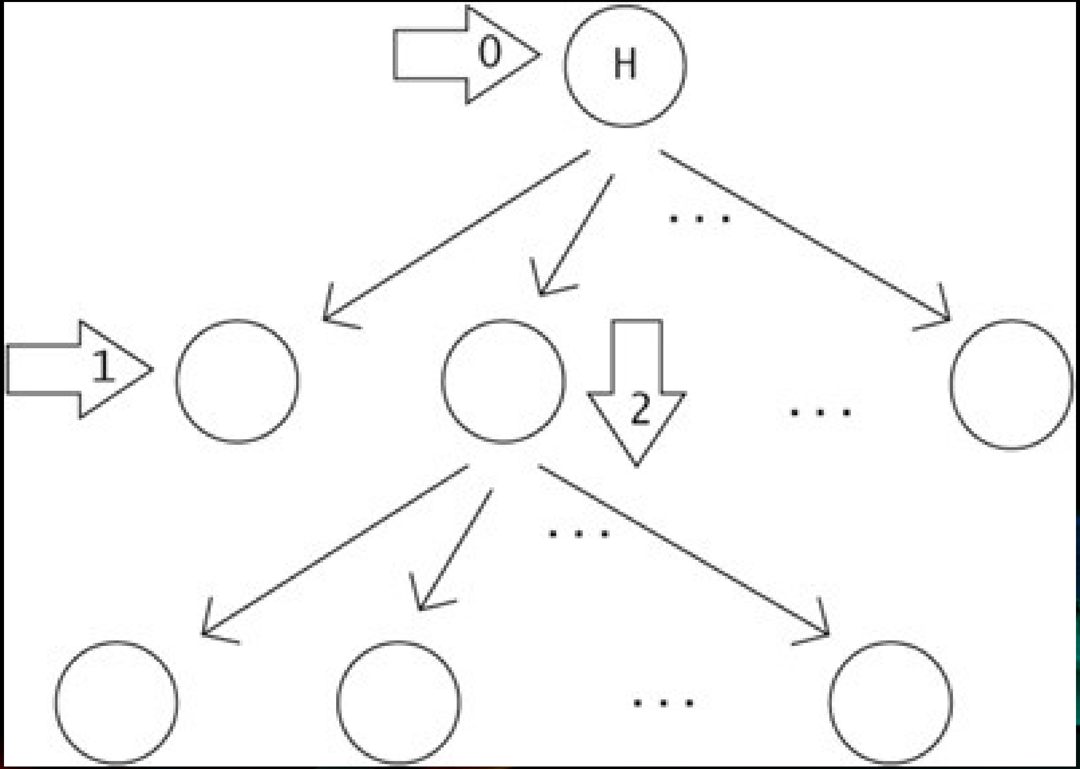
<<Java Class>>
ArtsIterator
jdp2e.iterator.demo
papers: String[]
position: int
ArtsIterator(String[])
first():void
next():String
currentItem():String
hasNext():boolean
<<Java Class>>
IteratorPatternExample
jdp2e.iterator.demo
IteratorPatternExample()
main(String[]):void
<<Java Class>>
Arts
jdp2e.iterator.demo
papers: String[]
Arts()
createIterator():Iterator
<<Java Interface>>
Iterator
jdp2e.iterator.demo
first():void
next():String
currentItem():String
hasNext():boolean
<<Java Interface>>
Subjects
jdp2e.iterator.demo
createIterator():Iterator

ITERATOR

Iterator and Composite pattern

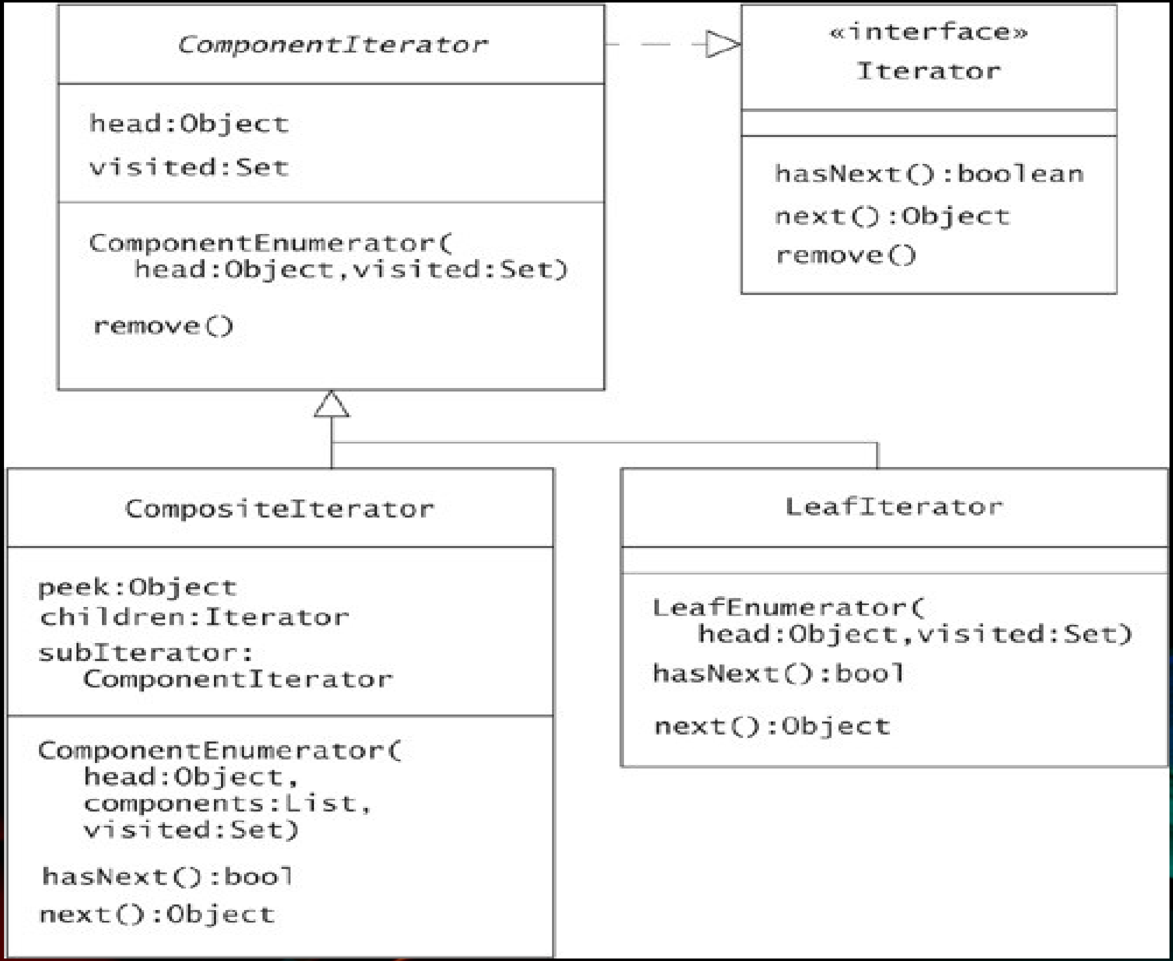
- To iterate over a composite, we iterate over its children, although this is a bit more complex than it may initially sound
- preorder (node before it’s descendants) and postorder (node after it’s descendants) traversal
- Maintaining more iterators

ITERATOR



ITERATOR

Iterator and Composite pattern



ITERATOR

Iterator consequences

- **Data modification**

  The most significant question resulting from the use of iterators concerns iterating through data while it is being changed

- **Privileged access**

  privileged access to the underlying data structures of the original container, usually done by getter

- **External versus internal Iterators**

  external – standard iterator through collection returning element

  internal - methods that move through the entire collection, performing some operation on each element directly, without any specific requests from the user

ITERATOR

Sources

- Java Design Patterns by James W. Cooper
- Design Patterns in Java, Second edition by William C. Wake; Steven John Metsker
- https://www.geeksforgeeks.org/iterator-pattern/
- https://sourcemaking.com/design_patterns/iterator

Mini quiz

ITERATOR

What are base methods in iterator according design patterns ?

ITERATOR

Questions and discussion