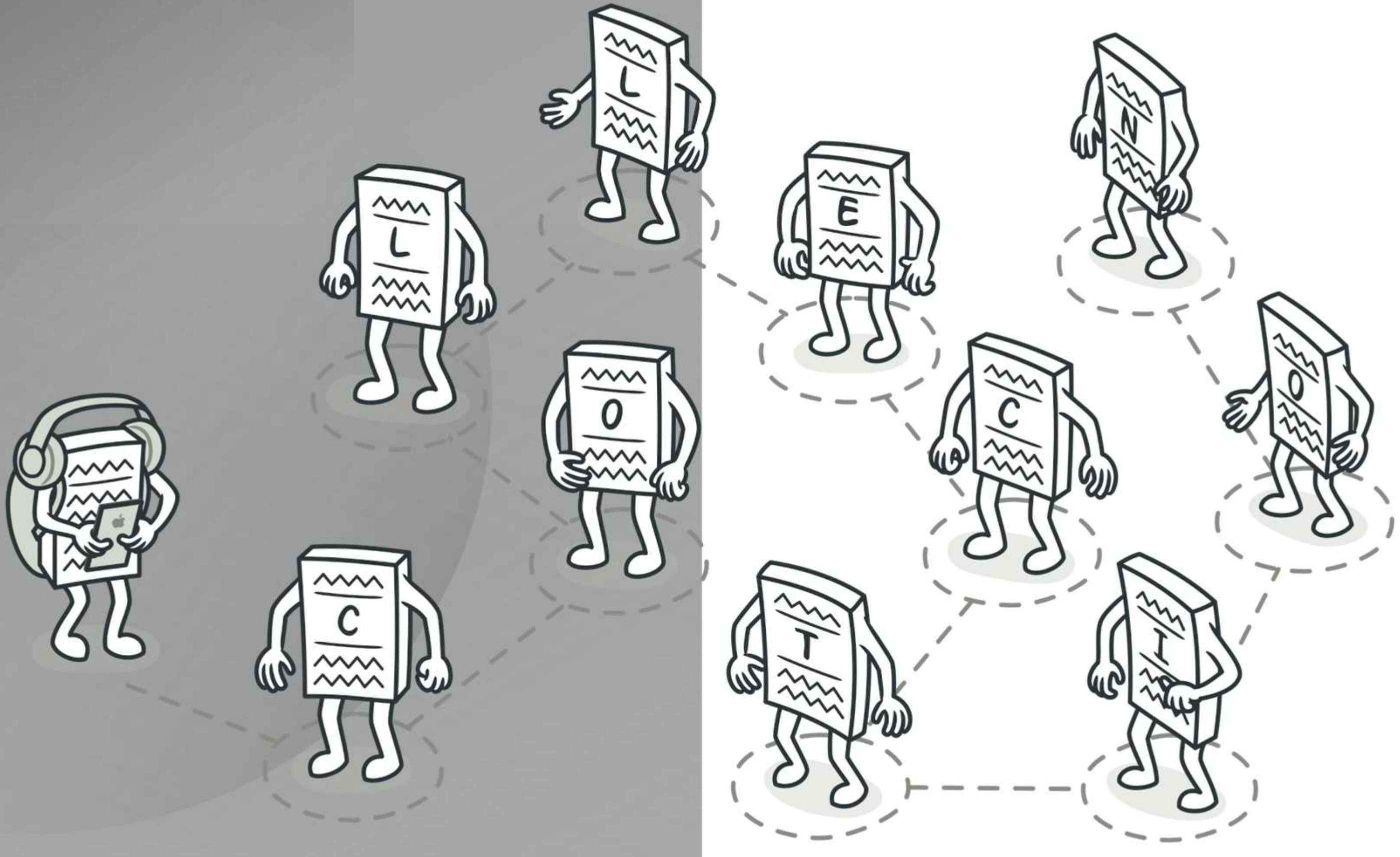
ITERATOR DESIGN PATTERN



INTRODUCTION

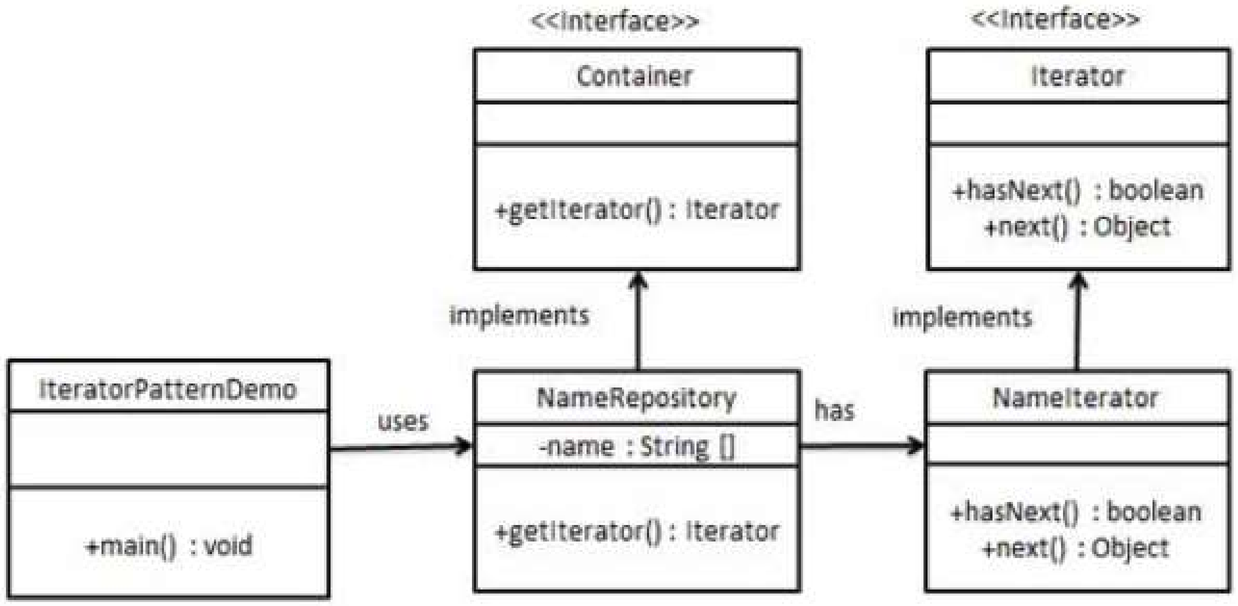
- This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation.

- Iterator pattern falls under behavioral pattern category.

IMPLEMENTATION

- Iterator interface will be created to narrate navigation method and a Container interface to return the iterator.
- Concrete classes implementing the Container interface will be responsible to implement Iterator interface and use it.
- IteratorPatternDemo, our demo class will use NamesRepository, a concrete class implementation to print a Names stored as a collection in NamesRepository.

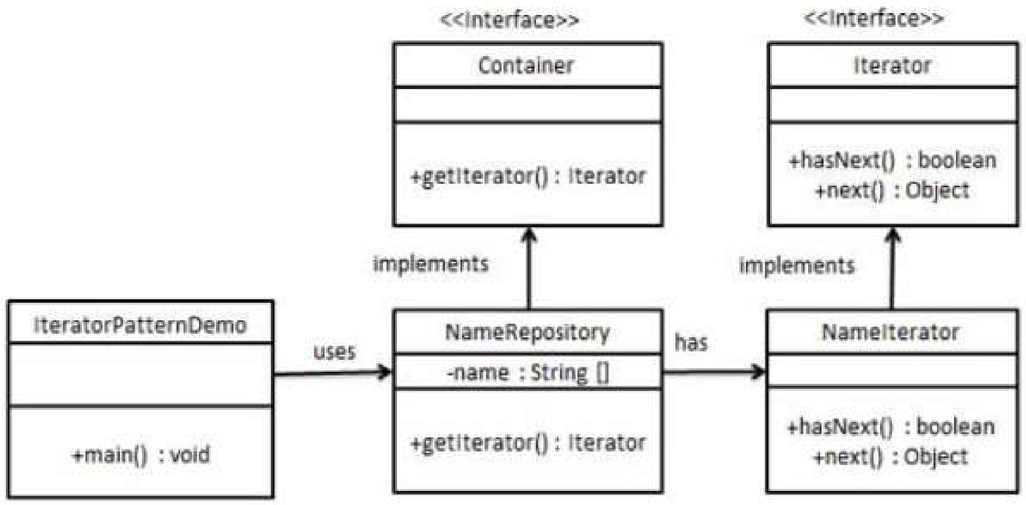
CLASS DIAGRAM



EXAMPLE WITH CODE

Iterator.java

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



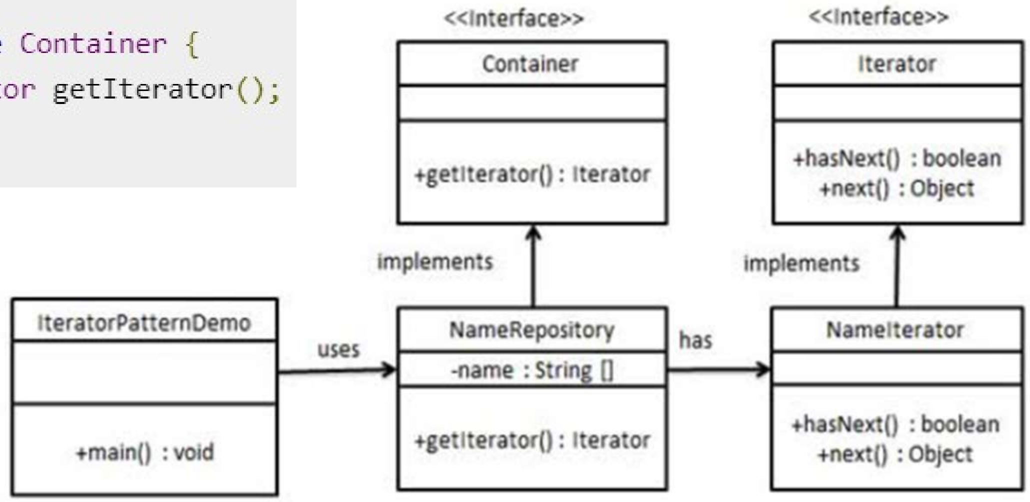
Step 1: Create Interfaces

Iterator.java

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

Container.java

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

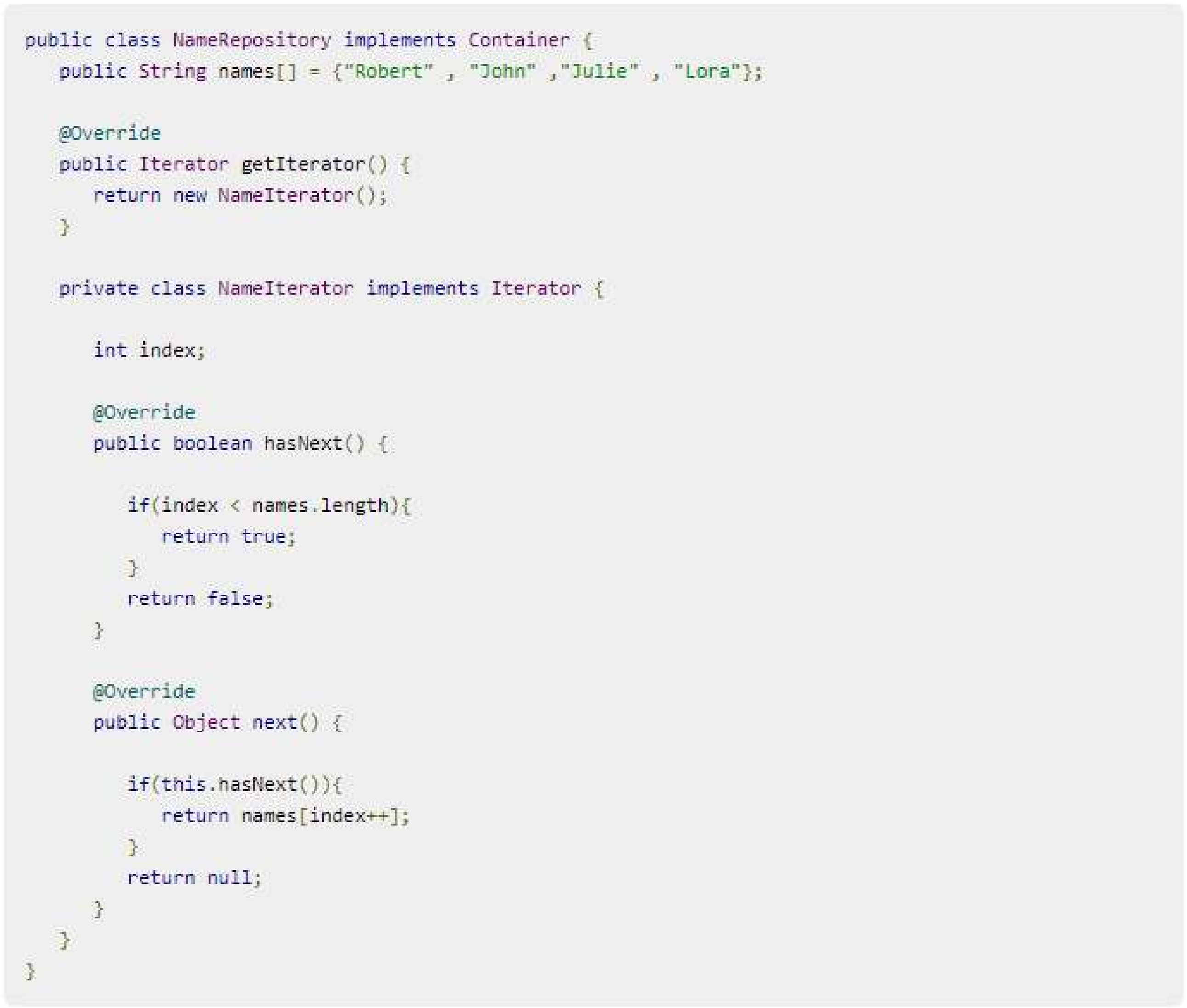


Step 2:

Create concrete class implementing the Container interface.

This class has inner class NameIterator implementing the Iterator interface.

NameRepository.java



```java
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;
      }
   }
}
```

Step 3

Use the *NameRepository* to get iterator and print names.

IteratorPatternDemo.java

```java
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
```