Iterator Design Pattern

By Maddie Ledesma and Aiden Varano

What is the Iterator Design pattern?

- The Iterator design pattern is used for the object-oriented family of software.

- The iterator pattern is a behavioral pattern, which are about identifying common patterns between objects

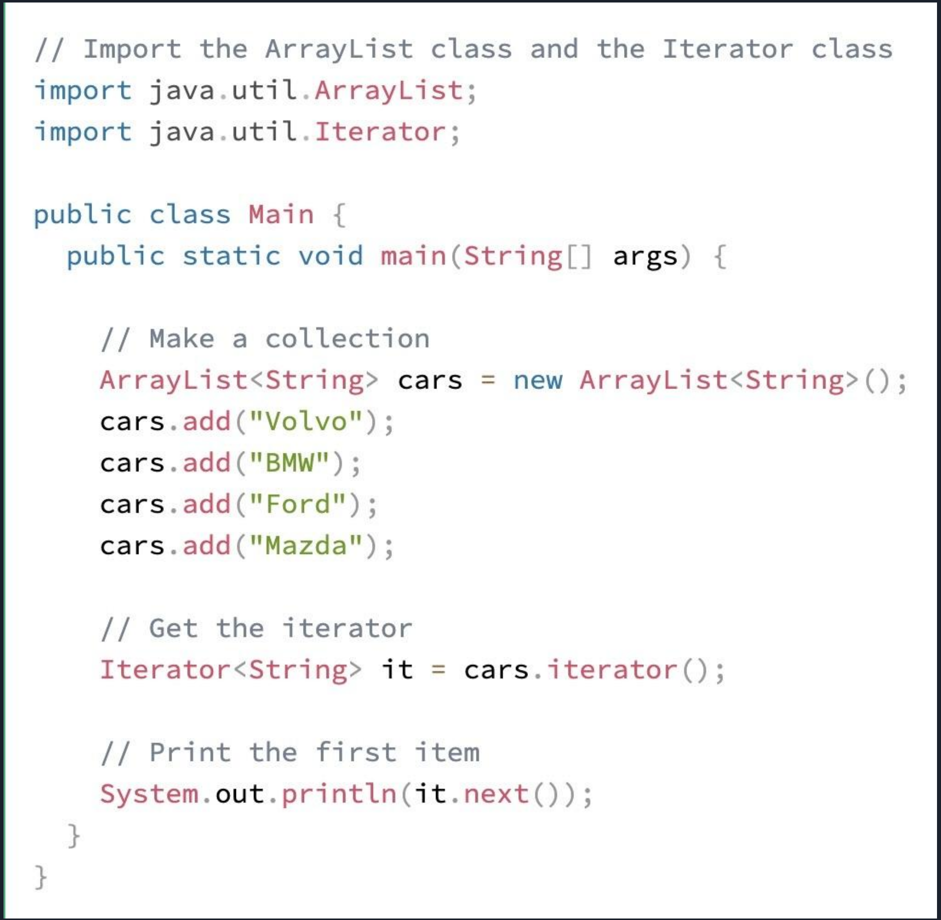
- The iterator is used to solve the problem of traversing through collections without knowing its internals or worrying about its implementation.

- It solves the problem by creating a uniform way to access the elements without worrying about their implementation, or what’s inside. Allowing for ease of use by decoupling iteration logic.

Common Use Cases

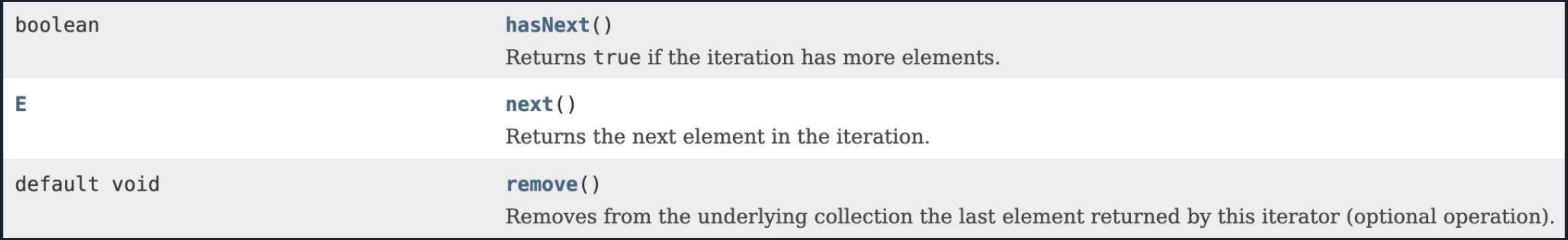
- Iterator Patterns are often used to obsfucate a collection’s underlying complex data structure from clients.
- It can “encapsulate” the details of working with a data structure “under the hood”
    - This can be for both convenience and Security Reasons.
- Protects underlying data from accidental, careless or malicious actions

The Code



```java
// Import the ArrayList class and the Iterator class
import java.util.ArrayList;
import java.util.Iterator;

public class Main {
  public static void main(String[] args) {

    // Make a collection
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");

    // Get the iterator
    Iterator<String> it = cars.iterator();

    // Print the first item
    System.out.println(it.next());
  }
}
```

The functions of Iterator



`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).

Pros and Cons of Iterator Design

Pros

- Single Responsibility Princple
- Open Closes Principle
- Allows for collections to be iterated over in parallel
- Likewise, iteration can be delayed and continued.

Cons

- Efficiency
- Compleximity

Citations

- refactoring guru . “Iterator.” Refactoring.guru, refactoring.guru/design-patterns/iterator.
- https://www.linkedin.com/pulse/single-responsibility-principle-software-design-sanjoy-kumar-malik#:~:text=The%20Single%20Responsibility%20Principle%20(SRP)%20is%20a%20fundamental%20concept%20in,well%2Ddefined%20responsibility%20or%20purpose.
- https://www.freecodecamp.org/news/open-closed-principle-solid-architecture-concept-explained/
-

Thank You!