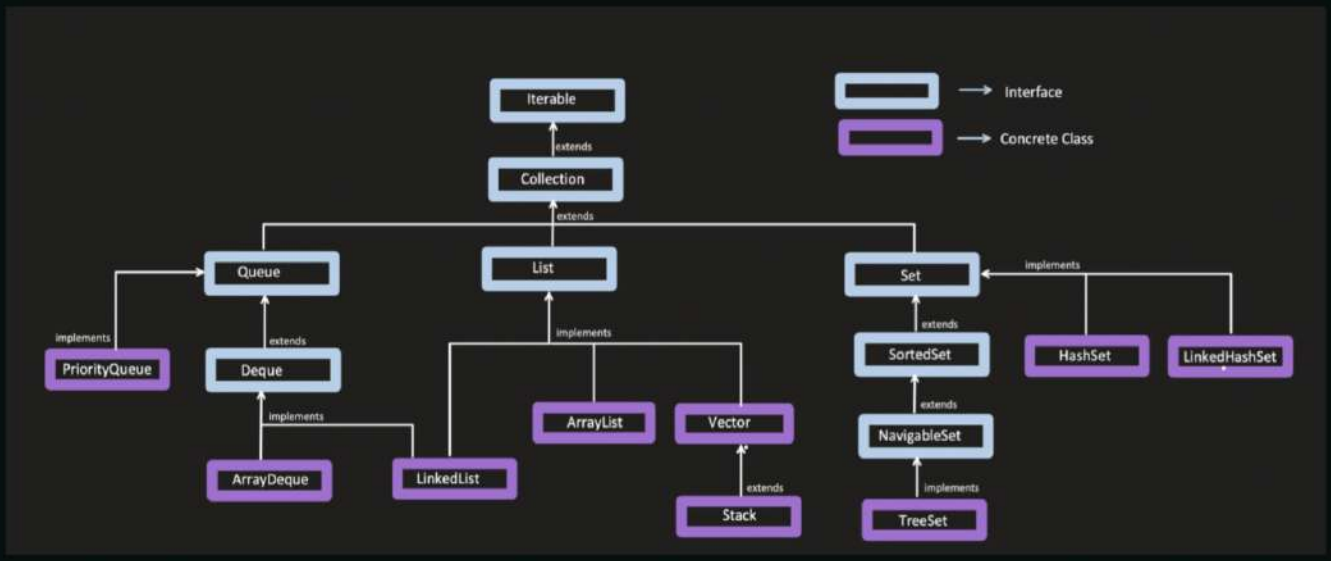
8. **Iterator Pattern:** that provides a way to access element of a Collection sequentially without exposing the underlying representation of the collection.

Understand the Need for an ITERATOR Pattern:



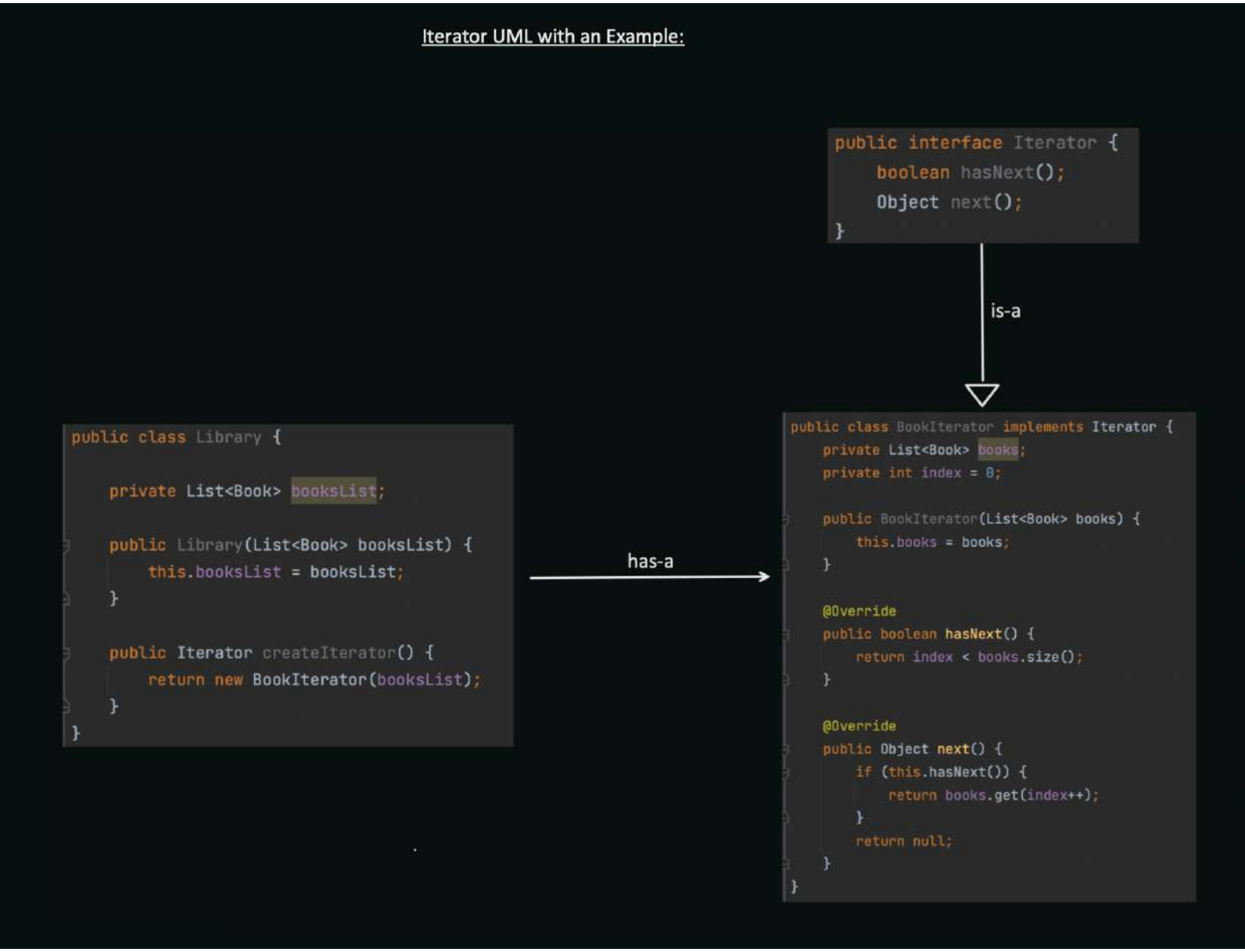

```java
public class LinkedHashSetExample {

    public static void main(String args[]){

        Set<Integer> intSet = new LinkedHashSet<>();
        intSet.add(2);
        intSet.add(77);
        intSet.add(82);
        intSet.add(63);
        intSet.add(5);

       Iterator<Integer> iterable =  intSet.iterator();
       while(iterable.hasNext()){
           int val = iterable.next();
           System.out.println(val);
       }
    }
}
```

Iterator UML with an Example:



```java
public class Client {

    public static void main(String[] args) {
        List<Book> booksList = Arrays.asList(
                new Book( price: 100,  bookName: "Science"),
                new Book( price: 200,  bookName: "Maths"),
                new Book( price: 300,  bookName: "GK"),
                new Book( price: 400,  bookName: "Drawing")
        );

        Library lib = new Library(booksList);
        Iterator iterator = lib.createIterator();

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

```java
public class Book {

    private int price;
    private String bookName;

    Book(int price, String bookName){
        this.price = price;
        this.bookName = bookName;
    }
    public int getPrice() {
        return price;
    }

    public String getBookName() {
        return bookName;
    }
}
```