Pratical Exercices N° 4 - Debriefing

Exercice 1 - Masterize Eclipse :)

Done ✔ ✔ ✔

Exercice 2 - Library

1. Create a class `Libary` with an attribute `book` ;

```java
import java.util.LinkedHashMap;
public class Library {
    private final ArrayList<Book> books;

    public Library() {
        books = new ArrayList<>();
    }

    public void add(Book book) {
        Objects.requireNonNull(book, "book must not be null");
        books.add(book);
    }

    public Book findByTitle(String title) {
        for (Book book : books) {
            if (book.title().equals(title)) {
                return book;
            }
        }
        return null;
    }
}
```

2. Add a method `add()` in the class

```
import java.util.ArrayList;
public class Library {
    private final ArrayList<Book> books;

    public Library() {
        books = new ArrayList<Book>();
    }
+    public void add(Book book) {
+        Objects.requireNonNull(book, "book must not be null");
+        books.add(book);
+    }
}
```

3. Add a method `findByTitle()` to the class

```
import java.util.ArrayList;
public class Library {
    private final ArrayList<Book> books;

    public Library() {
        books = new ArrayList<Book>();
    }

    public void add(Book book) {
        Objects.requireNonNull(book, "book must not be null");
        books.add(book);
    }

+    public Book findByTitle(String title) {
+        for (Book book : books) {
+            if (book.title().equals(title)) {
+                return book;
+            }
+        }
+        return null;
+    }
}
```

4. When the compiler found a `foreachloop`, he convert it to an iterator. We can se the conversion in the following bytecode:

```
Compiled from "Library.java"
public class Library {
public Library();
    Code:
    0: aload_0
    1: invokespecial #12                 // Method java/lang/Object."<init>":()V
    4: aload_0
    5: new           #14                 // class java/util/LinkedHashMap
    8: dup
    9: invokespecial #16                 // Method java/util/LinkedHashMap."
<init>":()V
    12: putfield      #17                 // Field books:Ljava/util/LinkedHashMap;
    15: return

public void add(Book);
    Code:
    ...
    20: return

public Book findByTitle(java.lang.String);
    Code:
    ...
    11: areturn

public void removeAllBooksFromAuthor(java.lang.String);
    Code:
    ...
    19: return

public java.lang.String toString();
    Code:
    0: new           #70                 // class java/lang/StringBuilder
    3: dup
    4: invokespecial #72                 // Method java/lang/StringBuilder."
<init>":()V
    7: astore_1
    8: ldc           #73                 // String
    10: astore_2
    11: aload_0
    12: getfield      #17                 // Field books:Ljava/util/LinkedHashMap;
    15: invokevirtual #54                 // Method
java/util/LinkedHashMap.values:()Ljava/util/Collection;
    18: invokeinterface #75,  1           // InterfaceMethod
java/util/Collection.iterator:()Ljava/util/Iterator;
    23: astore        4
    25: goto          55
    28: aload         4
    30: invokeinterface #79,  1           // InterfaceMethod
java/util/Iterator.next:()Ljava/lang/Object;
    35: checkcast     #34                 // class Book
    38: astore_3
    39: aload_1
    40: aload_2
    41: invokevirtual #85                 // Method
java/lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/StringBuilder;
44: aload_3
    45: invokevirtual #89                 // Method Book.toString:
()Ljava/lang/String;
    48: invokevirtual #85                 // Method
java/lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/StringBuilder;
51: pop
    52: ldc           #91                 // String \n
    54: astore_2
    55: aload         4
    57: invokeinterface #93,  1           // InterfaceMethod
java/util/Iterator.hasNext:()Z
    62: ifne          28
    65: aload_1
    66: invokevirtual #97                 // Method
java/lang/StringBuilder.toString:()Ljava/lang/String;
    69: areturn
}
```

5. The method `findByTitle` has to return `null` instead of raising an exception because it's a normal behavior to not find a book in library.

6. Add a method `toString()` to the class

```
import java.util.ArrayList;

public class Library {
    private final ArrayList<Book> books;

    public Library() {
        books = new ArrayList<Book>();
    }

    public void add(Book book) {
        Objects.requireNonNull(book, "book must not be null");
        books.add(book);
    }

    public Book findByTitle(String title) {
        for (Book book : books) {
            if (book.title().equals(title)) {
                return book;
            }
        }
        return null;
    }

+    @Override
+    public String toString() {
+        var output = new StringBuilder();
+        for (Map.Entry<String, Book> entry : books.entrySet()) {
+            output.append(entry.getKey()).append("\n");
+        }
+        return output.toString();
+    }
}
```

Exercice 3 - Library 2 - The return of vengeance

1. The complexity of the method `findByTitle` is O(n).
2. The Data structure implementing by HashMap is a dictionnary
    - To improve the perfomance of `findByTitle` method, we can use a HashMap to store each book with its title as key.
    - The complexity of the mehod will be O(1);
3. Rewrites the methods `add()` and `findByTitle()` using an HashMap.

```java
import java.util.HashMap;
import java.util.Objects;
public class Library {
    private final HashMap<String, Book> books;

    public Library() {
        books = new HashMap<>();
    }

    public void add(Book book) {
        Objects.requireNonNull(book, "book must not be null");
        books.put(book.title(), book);
    }

    public Book findByTitle(String title) {
        return books.get(title);
    }
}
```

4. It better to use a record instead of a class because with a record, anyone can modified the libray without using any method of the record. The encapsulation is not very strong in case we use a record.

5. To get all the values of the HashMap, we can use the `values()` method.

6. To have a library ordered by the insertion order, we can use a LinkedHashMap.

```diff
import java.util.LinkedHashMap;
import java.util.Objects;

public class Library {
+    private final LinkedHashMap<String, Book> books;

    public Library() {
+        books = new LinkedHashMap<>();
    }
    ...
}
```

7. Add a method `removeAllBooksFromAuthor()` to the class. Done ✔ ✔ ✔
    - The method raise the following exception :

```
Exception in thread "main" java.util.ConcurrentModificationException
   at
java.base/java.util.LinkedHashMap$LinkedHashIterator.nextNode(LinkedHashMap.j
ava:756)
   at
java.base/java.util.LinkedHashMap$LinkedValueIterator.next(LinkedHashMap.java
:783)
   at Library.removeAllBooksFromAuthor(Library.java:44)
   at Main.main(Main.java:19)
```

because while go trough the HashMap, the method `remove()` is called. The method `remove()` is not allowed to be called while iterating through a map.

8. Implement `removeAllBooksFromAuthor()` using an iterator.

```
...
public class Library {
    private final LinkedHashMap<String, Book> books;
    ...
+    public void removeAllBooksFromAuthor(String author) {
+        var iterator = books.values().iterator();
+        while (iterator.hasNext()) {
+            Book book = iterator.next();
+            if (book.author().equals(author)) {
+                iterator.remove();
+            }
+        }
+    }
    ...
}
```

9. Using the `removeIf` method to rewrite the method `removeAllBooksFromAuthor()`.

```diff
...
public class Library {
    private final LinkedHashMap<String, Book> books;
    ...
    public void removeAllBooksFromAuthor(String author) {
        var iterator = books.values().iterator();
-        while (iterator.hasNext()) {
-            Book book = iterator.next();
-            if (book.author().equals(author)) {
-                iterator.remove();
-            }
-        }
+       books.values().removeIf(book -> (book.author().equals(author)));
    }
    ...
}
```