Plan

2023-11-07

1. Homework Review
2. Iterator

---

1. Разбор домашнего задания
2. Iterator

Theory

▶ **English**

▼ **На русском**

Что такое Iterable?

В Java `Iterable` - это интерфейс, который представляет собой коллекцию объектов, по которой можно итерироваться, то есть пройтись последовательно от одного элемента к другому. Этот интерфейс определен в пакете `java.lang`.

Прежде всего, `Iterable` содержит метод `iterator()`, который возвращает итератор, еще один интерфейс, позволяющий проходить по коллекции объектов.

```
public interface Iterable<T> {
    Iterator<T> iterator();
}
```

Итератор, получаемый из `Iterable`, в свою очередь, должен реализовывать методы `hasNext()` и `next()`, а также `remove()` - который является необязательным.

- `hasNext()` проверяет, есть ли следующий элемент.
- `next()` возвращает следующий элемент.
- `remove()` (необязательный) удаляет текущий элемент.

Пример итератора:

```
Iterator<String> iterator = someCollection.iterator();
while(iterator.hasNext()) {
    String element = iterator.next();
```

```
        // Обрабатываем элемент
    }
```

Примеры из жизни

- Представьте `Iterable` как меню в ресторане. Вы можете просмотреть все блюда (`hasNext()`), выбрать блюдо (`next()`) и, если захотите, отменить заказ (`remove()`).
- `Iterable` - это как плейлист в музыкальном приложении. Вы можете переходить от песни к песне (итерироваться), пока песни не закончатся (`hasNext()` вернет `false`).

Задачи

1. **Задача "Букет"**: Создайте класс `Bouquet`, реализующий `Iterable`, который будет содержать список цветов. Реализуйте итератор, который позволит пройтись по всем цветам в букете.
2. **Задача "Книжная полка"**: Создайте класс `Bookshelf` с коллекцией книг. Реализуйте `Iterable` так, чтобы можно было итерироваться по книгам с помощью for-each цикла.
3. **Задача "Умный дом"**: Имеется набор устройств умного дома. Создайте класс `SmartHome`, реализующий `Iterable`, который позволит итерироваться по устройствам и, например, включать или выключать их.

Homework

▶ **English**

▼ **На русском**

1. Создайте класс `Playlist`, который реализует `Iterable` и содержит список песен. Реализовать итератор, который позволяет переключаться между песнями.
2. Создайте enum `WaterCycleStages` с этапами круговорота воды (испарение, конденсация, осадки). Реализуйте `Iterable` для перечисления, позволяющий пользователю перебирать этапы цикла.
3. Сделать самостоятельную реализацию приложения Forum.

```
public interface Forum {

    boolean addPost(Post post);

    boolean removePost(int postId);
```

```
    boolean updatePost(int postId, String newContent);

    Post getPostById(int postId);

    Post[] getPostsByAuthor(String author);

    Post[] getPostsByAuthor(String author, LocalDate dateFrom, LocalDate da

    int size();

}


public class ForumImpl implements Forum {

    // code...

}


class ForumImplTest {
    Forum forum;
    Post[] posts = new Post[6];

    @BeforeEach
    void setUp() {
        forum = new ForumImpl();
        posts[0] = new Post(0, "author1", "title1", "content");
        posts[1] = new Post(1, "author2", "title2", "content");
        posts[2] = new Post(2, "author2", "title3", "content");
        posts[3] = new Post(3, "author1", "title4", "content");
        posts[4] = new Post(4, "author3", "title1", "content");
        posts[5] = new Post(5, "author1", "title2", "content");
        for (int i = 0; i < posts.length - 1; i++) {
            forum.addPost(posts[i]);
        }
    }

    @Test
    void addPost() {
        //TODO assert throw if forum.addPost(null)
        boolean flag;
```

```
        try {
            forum.addPost(null);
            flag = true;
        } catch (RuntimeException e) {
            flag = false;
        }
        assertFalse(flag);
        assertTrue(forum.addPost(posts[5]));
        assertEquals(6, forum.size());
        assertFalse(forum.addPost(posts[5]));
        assertEquals(6, forum.size());
    }

    @Test
    void removePost() {
        assertTrue(forum.removePost(2));
        assertEquals(4, forum.size());
        assertFalse(forum.removePost(2));
        assertEquals(4, forum.size());
    }

    @Test
    void updatePost() {
        assertTrue(forum.updatePost(1, "new content"));
        assertEquals("new content", forum.getPostById(1).getContent());
    }

    @Test
    void getPostById() {
        assertEquals(posts[3], forum.getPostById(3));
        assertNull(forum.getPostById(5));
    }

    @Test
    void getPostsByAuthor() {
        Post[] actual = forum.getPostsByAuthor("author1");
        Arrays.sort(actual);
        Post[] expected = {posts[0], posts[3]};
        System.out.println(Arrays.toString(actual));
        System.out.println(Arrays.toString(expected));
        // assertArrayEquals(expected, actual);
    }
```

```
    @Test
    void testGetPostsByAuthor() {
        posts[0].setDate(LocalDateTime.now().minusDays(6));
        posts[1].setDate(LocalDateTime.now().minusDays(9));
        posts[2].setDate(LocalDateTime.now().minusDays(5));
        posts[3].setDate(LocalDateTime.now().minusDays(7));
        posts[4].setDate(LocalDateTime.now().minusDays(10));
        posts[5].setDate(LocalDateTime.now().minusDays(8));
        forum = new ForumImpl();
        for (int i = 0; i < posts.length; i++) {
            forum.addPost(posts[i]);
        }
        Post[] actual = forum.getPostsByAuthor("author1", LocalDate.now().m
        Arrays.sort(actual);
        Post[] expected = {posts[0], posts[3], posts[5]};
        assertArrayEquals(expected, actual);
    }

    @Test
    void size() {
        assertEquals(5, forum.size());
    }
}

public class Post {

    private int postId;
    private String title;
    private String author;
    private String content;
    private LocalDateTime date;
    private int likes;

    // коснтруктор и прочие методы


    @Override
    public String toString() {
        return "Post{" +
                "postId=" + postId +
                ", title='" + title + '\'' +
                ", author='" + author + '\'' +
                ", content='" + content + '\'' +
```

```
                ", date=" + date +
                ", likes=" + likes +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass())
            return false;  // вернем false, когда сравниваем с null или с об
        Post post = (Post) o;
        return postId == post.postId;
    }

    @Override
    public int hashCode() {
        return Objects.hash(postId);
    }
}
```

4.

Code

code/ClassWorkIterator/src/student/Student.java

```
package student;

import java.util.Arrays;
import java.util.Comparator;

public class Student {
    private int id;
    private String name;
    private double GPA;

    // Конструктор
    public Student(int id, String name, double GPA) {
        this.id = id;
        this.name = name;
        this.GPA = GPA;
    }
```

```
    // Геттеры и сеттеры
    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getGPA() {
        return GPA;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setGPA(double GPA) {
        this.GPA = GPA;
    }

    // Метод для сортировки массива студентов по GPA и выборки топ-5
    // Реализуйте метод, который сортирует массив студентов по GPA и исполь
    public static Student[] getTopStudents(Student[] students) {
        Arrays.sort(students, Comparator.comparingDouble((Student student)
        Student[] topStudents = new Student[5];
        System.arraycopy(students, 0, topStudents, 0, 5);
        return topStudents;
    }

    // Метод для поиска студента с ближайшим GPA
    // Напишите метод, который принимает средний балл и находит студента с
    public static Student findClosestGPA(Student[] students, double targetG
        // отсортируем массив студентов по GPA
        Arrays.sort(students, Comparator.comparingDouble((Student student)


        Student template = new Student(0, null, targetGPA);
        int index = Arrays.binarySearch(students, template);
```

```
        if (index >= 0) {
            return students[index];
        }

        index = Math.abs(index - 1);

        if (index == students.length) {
            return students[students.length - 1]; // Если нашли точное совп
        }
        // Сравнить соседние GPA
        /*
            получает индекс элемента, который находится непосредственно пер
            если бы он был в массиве. Студент, которого мы получаем на этой
            имеет GPA меньше или равный targetGPA.
         */
        Student left = students[index - 1];
        /*
            позиция, на которую должен был бы вставиться targetGPA. Студент
            имеет GPA больше targetGPA.
         */
        Student right = students[index];
        double leftGPA = left.getGPA();
        double rigthGPA = right.getGPA();

        /*
        Выполняется проверка: если абсолютное значение разности между GPA с
        меньше или равно абсолютному значению разности между GPA студента с
        то метод возвращает left. Это означает, что GPA left студента наход
         */
        if (Math.abs(leftGPA - targetGPA) <= Math.abs(rigthGPA - targetGPA)
            return left; // Вернуть студента с ближайшим GPA
            /*
            Если условие выше не выполняется, то это означает, что GPA студ
             и метод возвращает этого студента.
             */
        } else {
            return right; // Вернуть студента с ближайшим GPA
        }
    }

    // Для удобства вывода информации о студенте
    @Override
```

```
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", GPA=" + GPA +
                '}';
    }
}
```

code/ClassWorkIterator/src/MyString.java

```
import java.util.Iterator;

/**
 * @author Andrej Reutow
 * created on 07.11.2023
 */
public class MyString implements Iterable<Character> {

    private StringBuilder source;

    public MyString(String value) {
        this.source = new StringBuilder(value);
    }

    /**
     * добавляет символ
     *
     * @param c символ для добавления
     */
    public void addChar(char c) {
        source.append(c);
    }

    public StringBuilder getSource() {
        return source;
    }

    public void setSource(StringBuilder source) {
        this.source = source;
    }
```

```
    @Override
    public String toString() {
        return source.toString();
    }

    @Override
    public Iterator<Character> iterator() {
        return new Iterator<>() {

            private int size = source.length();
            private int currentPos;

            /**
             * Есть ли у нас еще символы
             * @return true, если символы есть
             */
            @Override
            public boolean hasNext() {
                return currentPos < size;
            }

            /**
             * Возвращает текущий символ и переходит к следующему
             * @return Возвращает текущий символ
             */
            @Override
            public Character next() {
                Character currentCharacter = source.charAt(currentPos);
                currentPos++;
                return currentCharacter;
            }
        };
    }
}
```

code/ClassWorkIterator/src/MyStringApp.java

```
import java.util.Iterator;

/**
 * @author Andrej Reutow
 * created on 07.11.2023
 */
public class MyStringApp {
```

```
    public static void main(String[] args) {
        MyString myString = new MyString("Hello world!");
        System.out.println(myString);

        myString.addChar('!');
        System.out.println(myString);

//        StringBuilder myStringSource = myString.getSource();
//        for (int i = 0; i < myStringSource.length(); i++) {
//            System.out.println(myStringSource.charAt(i));
//            System.out.println(Character.toUpperCase(myStringSource.charA
//        }
//
//        System.out.println("#".repeat(30));
//        myStringSource.setLength(3);
//        for (int i = 0; i < myStringSource.length(); i++) {
//            System.out.println(myStringSource.charAt(i));
//            System.out.println(Character.toUpperCase(myStringSource.charA
//        }


        for (Character c : myString) {
            System.out.println(c);
        }

        System.out.println();


        Iterator<Character> iterator = myString.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```