Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота 4
З дисципліни: «Теорія розробки програмного забезпечення» Download Manager
Iterator

Виконав:
студент групи ІА-14
Юлдашев А.Х.

Київ 2023

Мета: Реалізувати паттерн iterator в проекті на тему Download Manager

Код з коментарями:

Download Task

```java
/**
 * Клас, що представляє завдання для завантаження файлу з Інтернету.
 */
package model;

/**
 * Використання бібліотеки Lombok для генерації геттерів, сеттерів та конструктора з усіма аргументами.
 */
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;
import java.util.List;

Kepka09 *
@Getter
@Setter
@AllArgsConstructor


    public class DownloadTask {

    private String fileName;
    private String url;
    private int downloadSpeed; // в килобайтах в секунду
    private boolean isInProgress;

```

```java
     * Метод, який розпочинає завантаження файлу.
     * Встановлює значення {@code isInProgress} в {@code true}.
     */
    no usages  Kepka09
    public void start() {
        isInProgress = true;
        //Todo: тут надо будет сделать реальную загрузку файла по параметрам
    }

    /**
     * Метод, який призупиняє завантаження файлу.
     * Встановлює значення {@code isInProgress} в {@code false}.
     */
    no usages  Kepka09
    public void pause() {
        isInProgress = false;
        //todo: тут надо будет остановить загрузку файла
    }

    /**
     * Метод, який видаляє завдання для завантаження.
     * Встановлює значення {@code isInProgress} в {@code false}.
     */
    no usages  Kepka09
    public void delete() {
        isInProgress = false;
```

AlphabeticalDownloadIterator

```java
package service.iterator;

import model.DownloadTask;
import java.util.Comparator;
import java.util.Iterator;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.stream.Collectors;

/**
 * Клас-ітератор, що працює зі списком завдань для завантаження ітеруючи їх у відсортованому алфавітному порядку за назвою файлу.
 */
no usages  Kepka09 *
public class AlphabeticalDownloadIterator implements Iterator<DownloadTask> {
    3 usages
    private final List<DownloadTask> tasks;
    2 usages
    private int currentIndex = 0;

    /**
     * Конструктор класу.
     *
     * @param tasks Список завдань для завантаження.
     */
    no usages  Kepka09
    public AlphabeticalDownloadIterator(List<DownloadTask> tasks) {
        this.tasks = tasks.stream()
                .sorted(Comparator.comparing(DownloadTask::getFileName))
                .collect(Collectors.toList());
    }
    /**
     * Перевіряє наявність наступного завдання для завантаження.
     *
     * @return true, якщо існує наступне завдання, false - в іншому випадку.
     */
    Kepka09
    @Override
    public boolean hasNext() { return currentIndex < tasks.size(); }

    /**
     * Повертає наступне завдання для завантаження.
     *
     * @return Наступне завдання для завантаження.
     * @throws NoSuchElementException Якщо наступний елемент не існує.
     */
    Kepka09
    @Override
    public DownloadTask next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        return tasks.get(currentIndex++);
```

InProgressDownloadIterator

```java
package service.iterator;

import model.DownloadTask;
import java.util.Iterator;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.stream.Collectors;

/**
 * Клас-ітератор, що працює зі списком завдань для завантаження,
 * фільтруючи лише ті, які знаходяться у процесі завантаження.
 */
public class InProgressDownloadIterator implements Iterator<DownloadTask> {
    private final List<DownloadTask> tasks;
    private int currentIndex = 0;

    public InProgressDownloadIterator(List<DownloadTask> tasks) {
        this.tasks = tasks.stream()
                          .filter(DownloadTask::isInProgress)
                          .collect(Collectors.toList());
    }

    /**
     * Перевіряє наявність наступного завдання для завантаження, яке знаходиться у процесі.
     *
     * @return true, якщо існує наступне завдання у процесі, false - в іншому випадку.
     */
    @Override
    public boolean hasNext() { return currentIndex < tasks.size(); }

    /**
     * Повертає наступне завдання для завантаження, яке знаходиться у процесі.
     *
     * @return Наступне завдання для завантаження у процесі.
     * @throws NoSuchElementException Якщо наступний елемент не існує.
     */
    @Override
    public DownloadTask next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        return tasks.get(currentIndex++);
    }
}
```

InterruptedDownloadIterator

```java
/**
 * Клас-ітератор, що працює зі списком завдань для завантаження,
 * фільтруючи лише завдання, які перебувають у стані перерваного завантаження.
 */
no usages  Kepka09 *
public class InterruptedDownloadIterator implements Iterator<DownloadTask> {
    3 usages
    private final List<DownloadTask> tasks;
    2 usages
    private int currentIndex = 0;

    /**
     * Конструктор класу.
     *
     * @param tasks Список завдань для завантаження.
     */
    no usages  Kepka09
    public InterruptedDownloadIterator(List<DownloadTask> tasks) {
        this.tasks = tasks.stream()
                .filter(task -> !task.isInProgress())
                .collect(Collectors.toList());
    }
```

```java
    /**
     * Перевіряє наявність наступного завдання для завантаження, яке перебуває у стані перерваного завантаження.
     *
     * @return true, якщо існує наступне перерване завдання для завантаження, false - в іншому випадку.
     */
    Kepka09
    @Override
    public boolean hasNext() { return currentIndex < tasks.size(); }

    /**
     * Повертає наступне завдання для завантаження, яке перебуває у стані перерваного завантаження.
     *
     * @return Наступне перерване завдання для завантаження.
     * @throws NoSuchElementException Якщо наступний елемент не існує.
     */
    Kepka09
    @Override
    public DownloadTask next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        return tasks.get(currentIndex++);
    }
}
```

SpeedbasedDownloadIterator

```java
/**
 * Клас-ітератор, що працює зі списком завдань для завантаження,
 * сортуючи їх за швидкістю завантаження.
 */
no usages  Kepka09 *
public class SpeedBasedDownloadIterator implements Iterator<DownloadTask> {
    3 usages
    private final List<DownloadTask> tasks;
    2 usages
    private int currentIndex = 0;

    /**
     * Конструктор класу.
     *
     * @param tasks Список завдань для завантаження.
     */
    no usages  Kepka09
    public SpeedBasedDownloadIterator(List<DownloadTask> tasks) {
        this.tasks = tasks.stream()
                          .sorted(Comparator.comparingInt(DownloadTask::getDownloadSpeed))
                          .collect(Collectors.toList());
    }
```

```java
    /**
     * Перевіряє наявність наступного завдання для завантаження.
     *
     * @return true, якщо існує наступне завдання, false - в іншому випадку.
     */
    Kepka09
    @Override
    public boolean hasNext() { return currentIndex < tasks.size(); }

    /**
     * Повертає наступне завдання для завантаження.
     *
     * @return Наступне завдання для завантаження.
     * @throws NoSuchElementException Якщо наступний елемент не існує.
     */
    Kepka09
    @Override
    public DownloadTask next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        return tasks.get(currentIndex++);
    }
}
```

Висновок: В даній лабораторній роботі була написана частина програми “Download Manager” з використанням патерну Iterator.