



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.

Мета: Реалізувати паттерн iterator в проєкті на тему Download Manager

Код з коментарями:

Download Task

```
1  /**
2   * Клас, що представляє завдання для завантаження файлу з Інтернету.
3   */
4   package model;
5
6   /**
7    * Використання бібліотеки Lombok для генерації геттерів, сеттерів та конструктора з усіма аргументами.
8    */
9   import lombok.AllArgsConstructor;
10  import lombok.Getter;
11  import lombok.Setter;
12  import java.util.List;
13
14  @Kerka09 *
15  @Getter
16  @Setter
17  @AllArgsConstructor
18
19  public class DownloadTask {
20
21      private String fileName;
22      private String url;
23      private int downloadSpeed; // в кілобайтах в секунду
24      private boolean isInProgress;
25
26
27      * Метод, який розпочинає завантаження файлу.
28      * Встановлює значення {@code isInProgress} в {@code true}.
29      */
30      no usages @Kerka09
31      public void start() {
32          isInProgress = true;
33          //Todo: тут надо будет сделать реальную загрузку файла по параметрам
34      }
35
36      /**
37       * Метод, який призупиняє завантаження файлу.
38       * Встановлює значення {@code isInProgress} в {@code false}.
39       */
40      no usages @Kerka09
41      public void pause() {
42          isInProgress = false;
43          //todo: тут надо будет остановить загрузку файла
44      }
45
46      /**
47       * Метод, який видаляє завдання для завантаження.
48       * Встановлює значення {@code isInProgress} в {@code false}.
49       */
50      no usages @Kerka09
51      public void delete() {
52          isInProgress = false;
```

AlphabeticalDownloadIterator

```
1 package service.iterator;
2
3 import model.DownloadTask;
4 import java.util.Comparator;
5 import java.util.Iterator;
6 import java.util.List;
7 import java.util.NoSuchElementException;
8 import java.util.stream.Collectors;
9
10 /**
11  * Клас-ітератор, що працює зі списком завдань для завантаження ітеруючи їх у відсортованому алфавітному порядку за назвою файлу.
12  */
13 no usages 1 Kerpka09 *
14 public class AlphabeticalDownloadIterator implements Iterator<DownloadTask> {
15     3 usages
16     private final List<DownloadTask> tasks;
17     2 usages
18     private int currentIndex = 0;
19
20     /**
21     * Конструктор класу.
22     *
23     * @param tasks Список завдань для завантаження.
24     */
25     no usages 1 Kerpka09
26     public AlphabeticalDownloadIterator(List<DownloadTask> tasks) {
```

```
23         this.tasks = tasks.stream()
24             .sorted(Comparator.comparing(DownloadTask::getFileName))
25             .collect(Collectors.toList());
26     }
27     /**
28     * Перевіряє наявність наступного завдання для завантаження.
29     *
30     * @return true, якщо існує наступне завдання, false - в іншому випадку.
31     */
32     1 Kerpka09
33     @Override
34     public boolean hasNext() { return currentIndex < tasks.size(); }
35
36
37     /**
38     * Повертає наступне завдання для завантаження.
39     *
40     * @return Наступне завдання для завантаження.
41     * @throws NoSuchElementException Якщо наступний елемент не існує.
42     */
43     1 Kerpka09
44     @Override
45     public DownloadTask next() {
46         if (!hasNext()) {
47             throw new NoSuchElementException();
48         }
49         return tasks.get(currentIndex++);
50     }
```

InProgressDownloadIterator

```
1 package service.iterator;
2
3 import model.DownloadTask;
4 import java.util.Iterator;
5 import java.util.List;
6 import java.util.NoSuchElementException;
7 import java.util.stream.Collectors;
8
9 /**
10  * Клас-ітератор, що працює зі списком завдань для завантаження,
11  * фільтруючи лише ті, які знаходяться у процесі завантаження.
12  */
13 no usages Kepka09 *
14 public class InProgressDownloadIterator implements Iterator<DownloadTask> {
15     3 usages
16     private final List<DownloadTask> tasks;
17     2 usages
18     private int currentIndex = 0;
19
20     no usages Kepka09
21     @ public InProgressDownloadIterator(List<DownloadTask> tasks) {
22         this.tasks = tasks.stream()
23             .filter(DownloadTask::isInProgress)
24             .collect(Collectors.toList());
25     }
26
27     /**
28      * Перевіряє наявність наступного завдання для завантаження, яке знаходиться у процесі.
29      *
30      * @return true, якщо існує наступне завдання у процесі, false - в іншому випадку.
31      */
32     Kepka09
33     @Override
34     public boolean hasNext() { return currentIndex < tasks.size(); }
35
36     /**
37      * Повертає наступне завдання для завантаження, яке знаходиться у процесі.
38      *
39      * @return Наступне завдання для завантаження у процесі.
40      * @throws NoSuchElementException Якщо наступний елемент не існує.
41      */
42     Kepka09
43     @Override
44     public DownloadTask next() {
45         if (!hasNext()) {
46             throw new NoSuchElementException();
47         }
48         return tasks.get(currentIndex++);
49     }
50 }
```

InterruptedDownloadIterator

```
9      /**
10       * Клас-ітератор, що працює зі списком завдань для завантаження,
11       * фільтруючи лише завдання, які перебувають у стані перерваного завантаження.
12       */
13      no usages Kepka09 *
14      public class InterruptedDownloadIterator implements Iterator<DownloadTask> {
15          3 usages
16          private final List<DownloadTask> tasks;
17          2 usages
18          private int currentIndex = 0;
19
20      /**
21       * Конструктор класу.
22       *
23       * @param tasks Список завдань для завантаження.
24       */
25      no usages Kepka09
26      @ public InterruptedDownloadIterator(List<DownloadTask> tasks) {
27          this.tasks = tasks.stream()
28              .filter(task -> !task.isInProgress())
29              .collect(Collectors.toList());
30      }
```

```
28      /**
29       * Перевіряє наявність наступного завдання для завантаження, яке перебуває у стані перерваного завантаження.
30       *
31       * @return true, якщо існує наступне перерване завдання для завантаження, false - в іншому випадку.
32       */
33      Kepka09
34      @Override
35      public boolean hasNext() { return currentIndex < tasks.size(); }
36
37      /**
38       * Повертає наступне завдання для завантаження, яке перебуває у стані перерваного завантаження.
39       *
40       * @return Наступне перерване завдання для завантаження.
41       * @throws NoSuchElementException Якщо наступний елемент не існує.
42       */
43      Kepka09
44      @Override
45      public DownloadTask next() {
46          if (!hasNext()) {
47              throw new NoSuchElementException();
48          }
49          return tasks.get(currentIndex++);
50      }
51  }
```

SpeedbasedDownloadIterator

```
10  /**
11   * Клас-ітератор, що працює зі списком завдань для завантаження,
12   * сортуючи їх за швидкістю завантаження.
13   */
14  no usages Kepka09 *
15  public class SpeedBasedDownloadIterator implements Iterator<DownloadTask> {
16      3 usages
17      private final List<DownloadTask> tasks;
18      2 usages
19      private int currentIndex = 0;
20
21      /**
22       * Конструктор класу.
23       *
24       * @param tasks Список завдань для завантаження.
25       */
26      no usages Kepka09
27      @ public SpeedBasedDownloadIterator(List<DownloadTask> tasks) {
28          this.tasks = tasks.stream()
29              .sorted(Comparator.comparingInt(DownloadTask::getDownloadSpeed))
30              .collect(Collectors.toList());
31      }
```

```
29  /**
30   * Перевіряє наявність наступного завдання для завантаження.
31   *
32   * @return true, якщо існує наступне завдання, false - в іншому випадку.
33   */
34  Kepka09
35  @Override
36  public boolean hasNext() { return currentIndex < tasks.size(); }
37
38  /**
39   * Повертає наступне завдання для завантаження.
40   *
41   * @return Наступне завдання для завантаження.
42   * @throws NoSuchElementException Якщо наступний елемент не існує.
43   */
44  Kepka09
45  @Override
46  public DownloadTask next() {
47      if (!hasNext()) {
48          throw new NoSuchElementException();
49      }
50      return tasks.get(currentIndex++);
51  }
52 }
```

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