1 Лабораторная работа №3
Исследование коллекций и итераторов в языке Java

1.1. Цель работы

В ходе выполнения данной лабораторной работы необходимо ознакомиться с организацией коллекций объектов на языке Java, приобрести практические навыки использования списков, очередей, хеш-таблиц при создании Java программ.

1.2. Постановка задачи

Был выдан вариант 13.

В соответствии с вариантом задания реализовать класс для представления требуемой информации. Реализовать коллекцию типа HashSet объектов разработанного класса с возможностью ввода элементов из файла, вывода на консоль, проверки членства по введенному с консоли значению поля 1. Имя файла вводить параметром командной строки –i.

Реализовать коллекцию типа LinkedList объектов с возможностями: упорядочивания по полю 1 (использовать Collections.sort(list)); упорядочивания по полю 1 в порядке убывания (использовать Collections.sort(list, myComp), где myComp – экземпляр разработанного класса, реализующего интерфейс Comparator); ввода элементов из файла, вывода на консоль и сохранения в файл. Имена файлов вводить параметрами командной строки –i и –o.

Реализовать коллекцию типа HashMap объектов с ключом по значению поля 1, с возможностью ввода элементов из файла, вывода на консоль в виде «Ключ -> Значения» (значения остальных полей), вывода значения полей по введенному с консоли значению поля 1. Имя файла вводить параметром командной строки –i.

1.3. Ход выполнения работы

1.3.1. Текст программы

Программа составлена на языке Java.

Листинг 1 — Класс App

```
package org.cory7666.lab3;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.List;

import com.opencsv.bean.CsvToBeanBuilder;

public class App
{
  public static void main (String[] args) throws IllegalAccessException
  {
    try
    {
      var params = new CommandArguments(args).parse();
      List<Disc> initialData = readInitialDataFrom(params.input);
      new FirstTask(initialData).execute();
      new SecondTask(initialData).execute().saveTo(params.output);;
      new ThirdTask(initialData).execute();
    }
    catch (IllegalArgumentException | IllegalAccessException e)
    {
      System.err.println(e.getMessage());
    }
    catch (IOException e)
    {
      System.err.println("[ОШИБКА] Ошибка ввода-вывода.");
    }
  }

  private static List<Disc> readInitialDataFrom (File file) throws IOException
  {
    try (var bufReader = Files.newBufferedReader(file.toPath()))
    {
      return new CsvToBeanBuilder<Disc>(bufReader).withSeparator(';').withType(Disc.class).build().parse();
    }
  }
}
```

Листинг 2 — Класс FirstTask

```
package org.cory7666.lab3;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

public class FirstTask
{
  private final Set<Disc> data;

  public FirstTask (Collection<Disc> data)
  {
```

```
    this.data = new HashSet<>(data);
  }

  public void execute ()
  {
    System.out.println("=== Выполнение работы с " + data.getClass().getName() + " ===");

    System.out.println("Вывод содержимого:");
    for (var e : data)
    {
      System.out.println("* " + e);
    }

    try
    {
      var key = readKey(System.in);
      Disc foundedElement = null;
      for (var element : data)
      {
        if (element.albumTitle.equals(key))
        {
          foundedElement = element;
          break;
        }
      }
      System.out.println("Найдено: " + foundedElement.toString());
    }
    catch (IOException | NullPointerException e)
    {
      System.out.println("Указанный элемент не найден.");
    }

    System.out.println("=== Окончание работы с " + data.getClass().getName() + " ===");
  }

  private String readKey (InputStream stream) throws IOException
  {
    var buffReader = new BufferedReader(new InputStreamReader(stream));
    System.out.print("Введите ключ для проверки: ");
    return buffReader.readLine();
  }
}
```

Листинг 3 — Класс SecondTask

```
package org.cory7666.lab3;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.Collection;
import java.util.Collections;
import java.util.LinkedList;
import java.util.List;

public class SecondTask
{
  private final List<Disc> data;

  public SecondTask (Collection<Disc> data)
  {
    this.data = new LinkedList<>(data);
  }

  public SecondTask execute ()
  {
    System.out.println("=== Выполнение работы с " + data.getClass().getName() + " ===");

    System.out.println("Вывод содержимого (до сортировки):");
    data.forEach(this::printElement);

    System.out.println();
    System.out.println("Вывод содержимого (после сортировки):");
    Collections.sort(data);
```

```
    data.forEach(this::printElement);

    System.out.println("=== Окончание работы с " + data.getClass().getName() + " ===");

    return this;
  }

  public void saveTo (File file) throws IOException
  {
    System.out.println("=== Сохранение данных " + data.getClass().getName() + " в \"" + file.getPath() + "\"
===");
    try (var writer = Files.newBufferedWriter(file.toPath()))
    {
      data.forEach(x -> {
        try
        {
          writer.write(x.toString());
          writer.newLine();
        }
        catch (IOException e)
        {
          e.printStackTrace();
        }
      });
    }
  }

  private void printElement (Disc disc)
  {
    System.out.println("* " + disc);
  }
}
```

Листинг 4 — Класс ThirdTask

```
package org.cory7666.lab3;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

public class ThirdTask
{
  private final Map<String, Disc> data;

  public ThirdTask (Collection<Disc> data)
  {
    this.data = new HashMap<>();
    data.forEach(disc -> this.data.put(disc.albumTitle, disc));
  }

  public void execute ()
  {
    System.out.println("=== Выполнение работы с " + data.getClass().getName() + " ===");

    System.out.println("Вывод содержимого:");
    data.forEach(this::printRecord);

    try
    {
      System.out.println("Найдено: " + data.get(readKey(System.in)).toString());
    }
    catch (IOException | NullPointerException e)
    {
      System.out.println("Указанный элемент не найден.");
    }

    System.out.println("=== Окончание работы с " + data.getClass().getName() + " ===");
  }
```

```
  private void printRecord (String key, Disc disc)
  {
    System.out.println("* " + key + " -> " + disc);
  }

  private String readKey (InputStream stream) throws IOException
  {
    var buffReader = new BufferedReader(new InputStreamReader(stream));
    System.out.print("Введите ключ для проверки: ");
    return buffReader.readLine();
  }
}
```

Листинг 5 — Класс Disc

```
package org.cory7666.lab3;

import com.opencsv.bean.CsvBindByName;

public class Disc implements Comparable<Disc>
{
  @CsvBindByName (column = "Album Title") public final String albumTitle;
  @CsvBindByName (column = "Artist") public final String artist;
  @CsvBindByName (column = "Tracks Count") public final int tracksCount;
  @CsvBindByName (column = "Duration") public final int duration;

  public Disc (String albumTitle, String artist, int tracksCount, int duration)
  {
    this.albumTitle = albumTitle;
    this.artist = artist;
    this.tracksCount = tracksCount;
    this.duration = duration;
  }

  public Disc ()
  {
    this("", "", -1, -1);
  }

  @Override
  public int compareTo (Disc o)
  {
    return albumTitle.length() - o.albumTitle.length();
  }

  @Override
  public String toString ()
  {
    return String.format("Альбом <%s>. Автор <%s>. Песен %d (%d).", albumTitle, artist, tracksCount, duration);
  }
}
```

Листинг 6 — Класс CommandArguments

```
package org.cory7666.lab3;

import java.io.File;
import java.io.IOException;

public class CommandArguments
{
  private final String[] args;

  public CommandArguments (String[] args)
  {
    this.args = args;
  }

  public Params parse () throws IllegalAccessException, IOException
  {
    int i = 0;
```

```
    try
    {
      File input = null, output = File.createTempFile("lab3-", "-output");

      for (; i < args.length; ++i)
      {
        if (args[i].startsWith("-"))
        {
          switch (args[i])
          {
          case "-i", "--input":
            input = new File(args[++i]);
            break;
          case "-o", "--output":
            output = new File(args[++i]);
            break;
          }
        }
      }
      if (input == null)
        throw new IllegalAccessException("Excpected input file, no such given.");
      else
        return new Params(input, output);
    }
    catch (IndexOutOfBoundsException e)
    {
      throw new IllegalArgumentException("After parametr " + i + " expected value.");
    }
  }
}
```

Листинг 7 — Класс Params

```
package org.cory7666.lab3;

import java.io.File;

public class Params
{
  public final File input;
  public final File output;

  public Params (File input, File output)
  {
    this.input = input;
    this.output = output;
  }
}
```

1.3.2. Результаты тестирования

Программа была скомпилированна и запущена. Рисунок демонстрирует запуск программы без параметра -i.

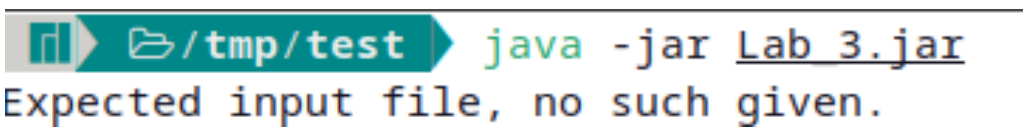

Рисунок 1 — Запуск программы без аргументов

Рисунки демонстрируют вывод программы, разделённый на три части. Ввод данных осуществлялся из переданного программе аргумента.

```
=== Выполнение работы с java.util.HashSet ===
Вывод содержимого:
* Альбом <AlbumBAC>. Автор <Artist2>. Песен 9 (300).
* Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
* Альбом <AlbumA>. Автор <Artist1>. Песен 13 (609).
* Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).
Введите ключ для проверки: AlbumB
Найдено: Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
=== Окончание работы с java.util.HashSet ===
```

Рисунок 2 — Обработка HashSet

```
=== Выполнение работы с java.util.LinkedList ===
Вывод содержимого (до сортировки):
* Альбом <AlbumA>. Автор <Artist1>. Песен 13 (609).
* Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
* Альбом <AlbumBAC>. Автор <Artist2>. Песен 9 (300).
* Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).

Вывод содержимого (после сортировки):
* Альбом <AlbumA>. Автор <Artist1>. Песен 13 (609).
* Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
* Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).
* Альбом <AlbumBAC>. Автор <Artist2>. Песен 9 (300).
=== Окончание работы с java.util.LinkedList ===
=== Сохранение данных java.util.LinkedList в "/tmp/lab3-7469296914828932385-output" ===
```

Рисунок 3 — Обработка LinkedList

```
=== Выполнение работы с java.util.HashMap ===
Вывод содержимого:
* AlbumB -> Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
* AlbumA -> Альбом <AlbumA>. Автор <Artist1>. Песен 13 (609).
* AlbumSY -> Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).
* AlbumBAC -> Альбом <AlbumBAC>. Автор <Artist2>. Песен 9 (300).
Введите ключ для проверки: AlbumSY
Найдено: Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).
=== Окончание работы с java.util.HashMap ===
```

Рисунок 4 — Обработка HashMap

При обработке HashSet был введён существующий ключ. Программа вывела ожидаемый элемент. Рисунок демонстрирует случай, когда введённый ключ не существует в структуре данных.

После обработки LinkedList его содержимое было сохранено в файл. Содержимое файла представлено на рисунке.

```
=== Выполнение работы с java.util.HashSet ===
Вывод содержимого:
* Альбом <AlbumBAC>. Автор <Artist2>. Песен 9 (300).
* Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
* Альбом <AlbumA>. Автор <Artist1>. Песен 13 (609).
* Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).
Введите ключ для проверки: NotExistenAlbum
Указанный элемент не найден.
=== Окончание работы с java.util.HashSet ===
```

Рисунок 5 — Проверка членства несуществующего ключа в HashSet

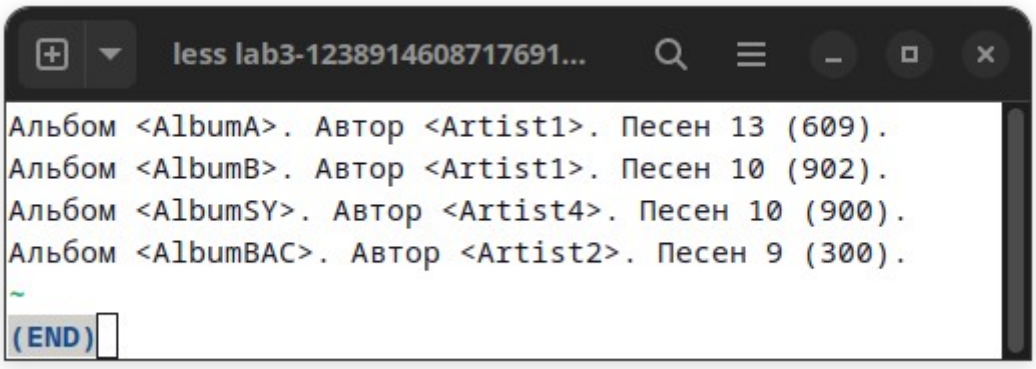

Рисунок 6 — Содержимое файла

При обработке HashMap был введён существующий ключ. Программа вывела ожидаемый элемент. Рисунок демонстрирует случай, когда введённый ключ не существует в структуре данных.

```
=== Выполнение работы с java.util.HashMap ===
Вывод содержимого:
* AlbumB -> Альбом <AlbumB>. Автор <Artist1>. Песен 10 (902).
* AlbumA -> Альбом <AlbumA>. Автор <Artist1>. Песен 13 (609).
* AlbumSY -> Альбом <AlbumSY>. Автор <Artist4>. Песен 10 (900).
* AlbumBAC -> Альбом <AlbumBAC>. Автор <Artist2>. Песен 9 (300).
Введите ключ для проверки: Abracadabra
Указанный элемент не найден.
=== Окончание работы с java.util.HashMap ===
```

Рисунок 7 — Проверка членства несуществующего ключа в HashMap

Вывод

При выполнении данной лабораторной работы были получены практические навыки работы с коллекция и итераторами, предоставляемые языком Java, изучены способы перебора коллекций.