Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №8

З дисципліни «Об’єктно-орієнтоване програмування»

Тема: «Робота з потоками в мові програмування Java»

Виконав:

студент групи ІВ-71

Мазан Я. В.

Номер залікової книжки:

7109

Перевірив:

Подрубайло О. О.

Київ 2018

1. Варіант завдання.

Номер залікової книжки — 7109

2. Код програми

Файл Main.java:

```
import java.io.*;

public class Main {

    public static void main(String[] args) throws IOException, ClassNotFoundException,
MyException {
        String destination = System.getProperty("user.dir")+ "/data/";
        new File(destination).mkdirs();
        Serializing worksWithFile = new Serializing();
        TestSerializing tests = new TestSerializing();
        Vegetable a = new Vegetable("Помідор", 12);
        Vegetable b = new Vegetable("Огірок", 8);
        Vegetable c = new Vegetable("Цибуля", 5);
        Vegetable d = new Vegetable("Морква", 13);
        VegetablesCollection salad = new VegetablesCollection();
        salad.add(a);
        salad.add(b);
        salad.add(c);
        salad.add(d);
        worksWithFile.writeCollectionAsObjectsList(salad, destination + "Objects List");
        worksWithFile.writeCollectionAsOneObject(salad, destination + "One Object");
        worksWithFile.writeCollectionAsText(salad, destination + "Text");
        tests.testReadCollectionAsText();
        tests.testReadCollectionAsOneObject();
        tests.testReadCollectionAsObjectsList();
        VegetablesCollection saladRead = worksWithFile.readCollectionAsObjectsList(destination +
"Objects List");
        System.out.println("Зчитаний із текстового файлу салат");
        for (Object i : saladRead) {
```

```
            System.out.println(i);
        }
        System.out.println("\nНезчитаний салат");
        for (Object i : salad) {
            System.out.println(i);
        }
    }
}
```

Файл Serializing.java:

```
mport java.io.*;

/**
 * Class which describes methods for working with input and output streams
 * writes to files and reads data from them
 */
public class Serializing {

    /**
     * Method that writes a  collection into a file as single object
     * @param collection
     * @param nameOfFile
     * @throws IOException
     */
    public void writeCollectionAsOneObject(VegetablesCollection collection, String nameOfFile)
throws IOException{
        File file = new File(nameOfFile);
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        ObjectOutputStream output = new ObjectOutputStream(fileOutputStream);

        output.writeObject(collection);

        output.close();
    }

    /**
```

```
    * Method that writes a collection into a file as list of objects from whose the collection consists
of
    * @param collection
    * @param nameOfFile
    * @throws IOException
    */
   public void writeCollectionAsObjectsList(VegetablesCollection collection, String nameOfFile)
throws IOException{
      File file = new File(nameOfFile);
      FileOutputStream fileOutputStream = new FileOutputStream( file);
      ObjectOutputStream output = new ObjectOutputStream(fileOutputStream);
      for (Object i: collection){
         output.writeObject(i);
      }
      output.close();
   }

   /**
    * Method that writes a collection into a text file with information about its objects
    * @param collection
    * @param nameOfFile
    * @throws IOException
    */
   public void writeCollectionAsText(VegetablesCollection collection, String nameOfFile) throws
IOException {
      File file = new File(nameOfFile);
      FileWriter fileOutputStream = new FileWriter(file);
      BufferedWriter output = new BufferedWriter(fileOutputStream);
      for (Object i: collection){
         output.write(i.toString() + "\n");
      }
      output.close();
   }

   /**
```

```
    * Method that reads collection from a file that includes a raw data with not modified collection
    * @param nameOfFile
    * @return VegetablesCollection collection
    * @throws IOException
    * @throws ClassNotFoundException
    */
   public VegetablesCollection readCollectionAsOneObject(String nameOfFile) throws
IOException, ClassNotFoundException{
      File file = new File(nameOfFile);
      FileInputStream fileInput = new FileInputStream(file);
      ObjectInputStream objectInput = new ObjectInputStream(fileInput);
      VegetablesCollection collection = (VegetablesCollection)objectInput.readObject();
      objectInput.close();
      return collection;
   }

   /**
    * Method that reads collection from a file that includes a raw data with list of objects from
whose the collection consists of
    * @param nameOfFile
    * @return VegetablesCollection collection
    * @throws IOException
    * @throws ClassNotFoundException
    */
   public VegetablesCollection readCollectionAsObjectsList(String nameOfFile) throws
IOException, ClassNotFoundException{
      File file = new File(nameOfFile);
      FileInputStream fileInput = new FileInputStream(file);
      ObjectInputStream objectInput = new ObjectInputStream(fileInput);
      VegetablesCollection collection = new VegetablesCollection();
      try{
      Vegetable veg = (Vegetable) objectInput.readObject();
         while (true) {
            collection.add(veg);
            veg = (Vegetable) objectInput.readObject();
```

```
            }
        } catch (EOFException e) {
            objectInput.close();
        }
        return collection;
    }

    /**
     * Method that reads collection from a file that includes a text with information from which the collection can be revived
     * @param nameOfFile
     * @return VegetablesCollection collection
     * @throws IOException
     * @throws ClassNotFoundException
     */
    public VegetablesCollection readCollectionAsText(String nameOfFile) throws IOException, MyException, ClassNotFoundException{
        File file = new File(nameOfFile);
        FileReader fileInput = new FileReader(file);
        BufferedReader objectInput = new BufferedReader(fileInput);
        VegetablesCollection collection = new VegetablesCollection();
        String inputLine = objectInput.readLine();
        while (inputLine != null) {
            String[] values = inputLine.split(" ");
            String vegName = values[1];
            int nutrition = Integer.parseInt(values[3]);
            Vegetable veg = new Vegetable(vegName, nutrition);
            collection.add(veg);
            inputLine =objectInput.readLine();
        }
        objectInput.close();
        return collection;
    }
}
```

Файл TestSerializing.java:

```
/**
 * Testing my Serializator that works with files
 */
import org.junit.Test;
import java.io.*;
import static org.junit.Assert.*;

public class TestSerializing {

    /**
     * Initialising variables for testing
     */
    private static final String destination = System.getProperty("user.dir")+ "/data/";

    private VegetablesCollection testCollection;
    private Serializing workWithFile;

    /**
     * Testing of method writeCollectionAsOneObject()
     */
    @Test
    public void testWriteCollectionAsOneObject() throws IOException,MyException{
        new File(destination).mkdirs();
        workWithFile = new Serializing();
        Vegetable a = new Vegetable("Помідор", 12);
        Vegetable b = new Vegetable("Огірок", 8);
        testCollection = new VegetablesCollection();
        testCollection.add(a);
        testCollection.add(b);
        workWithFile.writeCollectionAsOneObject(testCollection, destination + "raw collection");
        boolean check = new File(destination, "raw collection").exists();
        assertTrue(check);
    }

    /**
```

```
 * Testing of method writeCollectionAsObjectsList()
 */
@Test
public void testWriteCollectionAsObjectsList() throws IOException,MyException{
    workWithFile = new Serializing();
    Vegetable a = new Vegetable("Помідор", 12);
    Vegetable b = new Vegetable("Огірок", 8);
    testCollection = new VegetablesCollection();
    testCollection.add(a);
    testCollection.add(b);
    workWithFile.writeCollectionAsObjectsList(testCollection, destination + "objects list");
    boolean check = new File(destination, "objects list").exists();
    assertTrue(check);
}

/**
 * Testing of method writeCollectionAsText()
 */
@Test
public void testWriteCollectionAsText() throws IOException,MyException{
    workWithFile = new Serializing();
    Vegetable a = new Vegetable("Помідор", 12);
    Vegetable b = new Vegetable("Огірок", 8);
    testCollection = new VegetablesCollection();
    testCollection.add(a);
    testCollection.add(b);
    workWithFile.writeCollectionAsText(testCollection, destination+"text");
    boolean check = new File(destination, "text").exists();
    assertTrue(check);
}

/**
 * Testing of method readCollectionAsOneObject(
 */
@Test
```

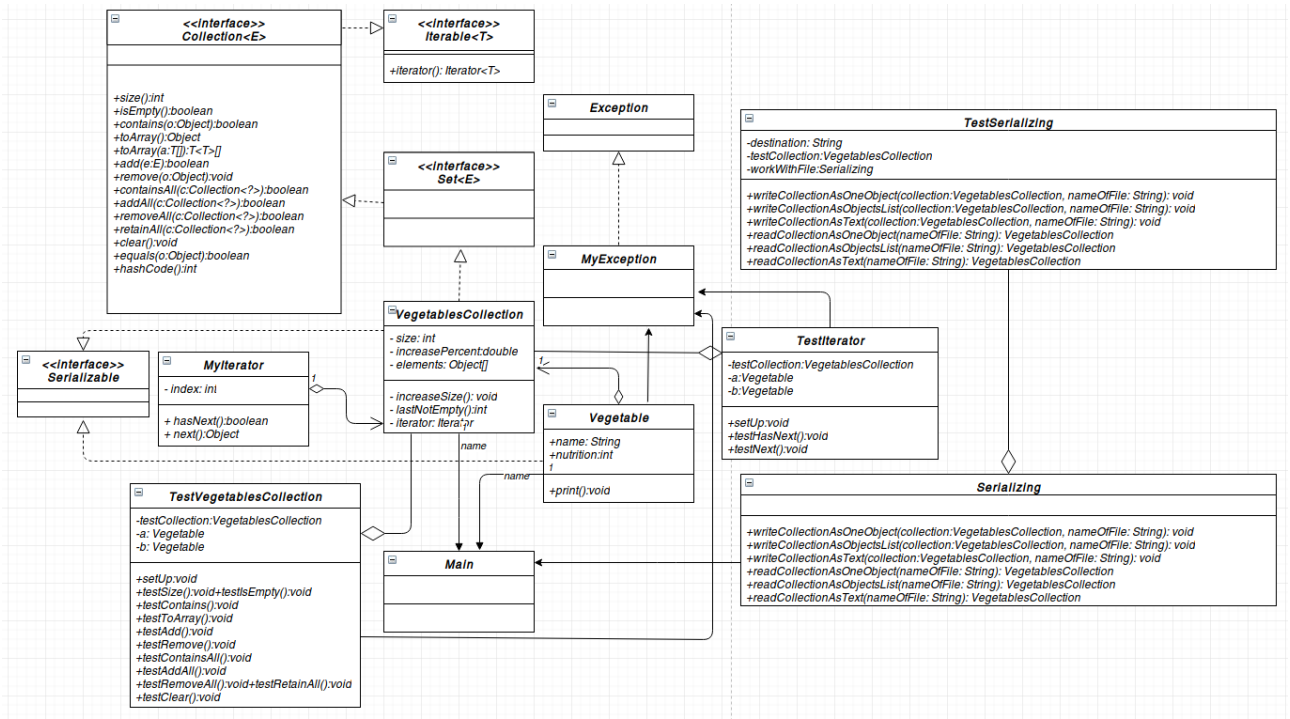
```
    public void testReadCollectionAsOneObject() throws IOException, ClassNotFoundException,
MyException {
        workWithFile = new Serializing();
        Vegetable a = new Vegetable("Помідор", 12);
        Vegetable b = new Vegetable("Огірок", 8);
        testCollection = new VegetablesCollection();
        testCollection.add(a);
        testCollection.add(b);
        workWithFile.writeCollectionAsOneObject(testCollection, destination+"raw collection");
        testCollection = workWithFile.readCollectionAsOneObject(destination + "raw collection");
        assertEquals(testCollection.size(),2);
    }

    /**
     * Testing of method readCollectionAsObjectsList
     */
    @Test
    public void testReadCollectionAsObjectsList() throws IOException, ClassNotFoundException,
MyException{
        workWithFile = new Serializing();
        Vegetable a = new Vegetable("Помідор", 12);
        Vegetable b = new Vegetable("Огірок", 8);
        testCollection = new VegetablesCollection();
        testCollection.add(a);
        testCollection.add(b);
        workWithFile.writeCollectionAsObjectsList(testCollection, destination+"objects list");
        testCollection = workWithFile.readCollectionAsObjectsList(destination + "objects list");
        assertEquals(testCollection.size(),2);
    }

    /**
     * Testing of method readCollectionAsText()
     */
    @Test
```

```
    public void testReadCollectionAsText() throws IOException, ClassNotFoundException,
MyException{
        workWithFile = new Serializing();
        Vegetable a = new Vegetable("Помідор", 12);
        Vegetable b = new Vegetable("Огірок", 8);
        testCollection = new VegetablesCollection();
        testCollection.add(a);
        testCollection.add(b);
        workWithFile.writeCollectionAsText(testCollection, destination+"text");
        testCollection = workWithFile.readCollectionAsText(destination + "text");
        assertEquals(testCollection.size(),2);
    }
}
```

3. UML-діаграма класів



4. Висновок

При виконанні цієї лабораторної роботи я вирішив зберігати робочі файли в окрему директорію, тому мені довелось зробити її створення автоматичним залежно від операційної ситеми.