Современные проблемы информатики

Задача №2

Метод «Стопка книг»

МГ-101 Тимофеев Д.А.

СОДЕРЖАНИЕ:

[СОДЕРЖАНИЕ: 1](#_Toc61878364)

[1. Задание 2](#_Toc61878365)

[2. Программный код 2](#_Toc61878366)

[2.1. StackBooks (interface) 2](#_Toc61878367)

[2.2. ByteStackBooksClass 2](#_Toc61878368)

[3. Тесты 5](#_Toc61878369)

[4. Ссылка на исходники 7](#_Toc61878370)

1. Задание

Разработать программу, которая с помощью метода "Стопка книг" и кодирования целых чисел сжимает заданный файл. В качестве буквы алфавита выступает произвольный байт.

1. Программный код
   1. StackBooks (interface)

|  |
| --- |
| public interface StackBooks {  public void archiveToBuffer(byte[] data);  public ChunkBits getArchivedData();  public void code(String inputFile, String outputFile) throws Exception;  public void decode(String decodeObject, String outputFile) throws Exception; } |

* 1. ByteStackBooksClass

|  |
| --- |
| public class ByteStackBooksClass implements StackBooks {  @Override  public void code(String data, String archivedFile) throws Exception {  List<Byte> alphabet = alphabetCreate();   byte[] actual = Files.*readAllBytes*(Path.*of*((new File(data)).getAbsolutePath()));   OutputFileBits writerBits = new OutputFileBitsClass(archivedFile);  byte[] offset = new byte[256];    ArchivatorNumber archivator = new ArchivarotNumberClass(1);    for (int i = 0; i < actual.length; i++) {  byte currentSimbol = actual[i];   *//поиск кода (позиция в алфавите)* int indexOf = alphabet.indexOf(currentSimbol);    *//сжатие кода // byte code = UnsignedByte.toSignedByte(indexOf);* ChunkBits archivedIndexOf = archivator.codeNumberToRequiredSize(new ChunkBitsClass(indexOf));  *//запись кода в файл* writerBits.writeChunkDataInEnd(archivedIndexOf);   *//найденную букву ставим в начало* alphabet.remove(indexOf);  alphabet.add(0, currentSimbol);  }  writerBits.close();         byte[] archivedStackBooksArray = Files.*readAllBytes*(Path.*of*((new File(archivedFile)).getAbsolutePath()));  byte[] alphabetByte = listToArrayByte(alphabet);   FileOutputStream file = new FileOutputStream(archivedFile);  file.write(alphabetByte);  file.write(archivedStackBooksArray);  file.close();  RandomAccessFile raf = new RandomAccessFile(archivedFile, "rw");  raf.seek(0);    raf.write(alphabetByte);  raf.close();  }  private byte[] listToArrayByte(List<Byte> pdu) {  Byte[] bytes = pdu.toArray(new Byte[pdu.size()]);    byte[] buasdfasdfsdf = new byte[bytes.length];   for (int i = 0; i < bytes.length; i++) {  buasdfasdfsdf[i] = bytes[i];  }   return buasdfasdfsdf;  }   private ArrayList<Byte> alphabetCreate() {  ArrayList<Byte> res = new ArrayList<Byte>();  int placeForAlpabet = 256;  for (int i = 0; i < placeForAlpabet; i++) {  res.add(UnsignedByte.*toSignedByte*(i));  }   return res;  }   @Override  public void decode(String archiveStackBooks, String data) throws Exception {  byte[] all = Files.*readAllBytes*(Path.*of*((new File(archiveStackBooks)).getAbsolutePath()));    ArrayList<Byte> alphabet = new ArrayList<Byte>();  int placeForAlpabet = 256;  for (int i = 0; i < placeForAlpabet; i++) {  alphabet.add(all[i]);  }     ArrayList<Byte> array = new ArrayList<>();   int sizeArrayCode = all.length - placeForAlpabet;  byte[] arrayCode = new byte[sizeArrayCode];   for (int i = placeForAlpabet, j = 0; i < all.length; j++, i++) {  arrayCode[j] = all[i];  }   String tempNameFile = "./test\_files/codeTail.bin";  Files.*write*(Path.*of*((new File(tempNameFile)).getAbsolutePath()), arrayCode);   ArchivatorNumber archivator = new ArchivarotNumberClass(1);  byte[] decodeArray = archivator.decodeFileByte(tempNameFile);   for (int i = decodeArray.length - 1; i >= 0 ; i--) {   *//0 символ алфавита добавляется в конец строки* byte firstItem = alphabet.get(0);  array.add(0, firstItem);   *//0 символ из алфавита вставляется в позицию нормер (код) в массив* firstSymbolInsertPosition(decodeArray, alphabet, i, firstItem);  }   assert (alphabetReturnedToOriginalOrder(alphabet));   byte[] res = listToArrayByte(array);    Files.*write*(Path.*of*((new File(data)).getAbsolutePath()), res);  }   private void firstSymbolInsertPosition(byte[] decodeArray, ArrayList<Byte> alphabet, int position, byte firstItem) {  alphabet.remove(0);  int code = UnsignedByte.*getInt*((byte) decodeArray[position]);  alphabet.add(code, firstItem);  }   private boolean alphabetReturnedToOriginalOrder(ArrayList<Byte> alphabet) {  for (int i = 1; i < alphabet.size(); i++) {  int indexPrevious = i - 1;  System.*out*.println(alphabet.get(indexPrevious));  boolean rightOrder = alphabet.get(indexPrevious) < alphabet.get(i);  if (!rightOrder) {  System.*out*.println("-------> ne poriadok: " + alphabet.get(indexPrevious) + " , " + alphabet.get(i));  return false;  }  }  return true;  } |

1. Тесты

|  |
| --- |
| @Test void sharedTest() throws Exception {  String dataPath = rootPath + "35dwt2ty2\_data.bin";   byte[] expectedArray = {1, 127, 120, -128, 0};  Files.*write*(Path.*of*((new File(dataPath)).getAbsolutePath()), expectedArray);    StackBooks archive = new ByteStackBooksClass();  String codePath = rootPath + "35dwt2ty2\_code.bin";  archive.code(dataPath, codePath);  String decodePath = rootPath + "35dwt2ty2\_decode.bin";  archive.decode(codePath, decodePath);    byte[] actual = Files.*readAllBytes*(Path.*of*((new File(decodePath)).getAbsolutePath()));   *assertArrayEquals*(expectedArray, actual);  assert(Arrays.*equals*(expectedArray, actual)); } @Test void sharedTest\_oneNumber() throws Exception {  String randomName = "35dwt2ty2246";  String dataPath = rootPath + randomName + "\_data.bin";   byte[] expectedArray = {3};  Files.*write*(Path.*of*((new File(dataPath)).getAbsolutePath()), expectedArray);    StackBooks archive = new ByteStackBooksClass();  String codePath = rootPath + randomName + "\_code.bin";  archive.code(dataPath, codePath);  String decodePath = rootPath + randomName + "\_decode.bin";  archive.decode(codePath, decodePath);    byte[] actual = Files.*readAllBytes*(Path.*of*((new File(decodePath)).getAbsolutePath()));   *assertArrayEquals*(expectedArray, actual);  assert(Arrays.*equals*(expectedArray, actual)); } @Test void sharedTest\_threeNumber() throws Exception {  String randomName = "35dwt2ty2246";  String dataPath = rootPath + randomName + "\_data.bin";   byte[] expectedArray = {3, 15, 40};  Files.*write*(Path.*of*((new File(dataPath)).getAbsolutePath()), expectedArray);    StackBooks archive = new ByteStackBooksClass();  String codePath = rootPath + randomName + "\_code.bin";  archive.code(dataPath, codePath);  String decodePath = rootPath + randomName + "\_decode.bin";  archive.decode(codePath, decodePath);    byte[] actual = Files.*readAllBytes*(Path.*of*((new File(decodePath)).getAbsolutePath()));   *assertArrayEquals*(expectedArray, actual);  assert(Arrays.*equals*(expectedArray, actual)); } |

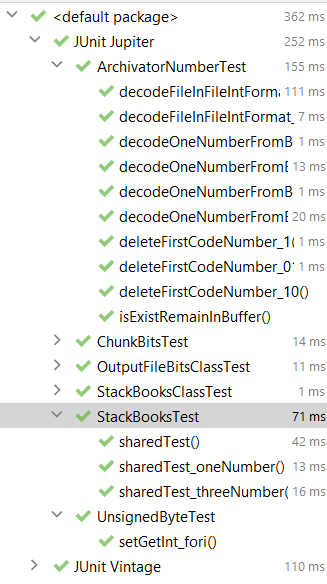


Рисунок 1 – результаты тестов

1. Ссылка на исходники

[SibGUTY\_git/5k1s/СПИ - Современные проблемы информатики (Фионов)/labs\_realisation/lab\_1-3 at master · GeorgiaFrankinStain/SibGUTY\_git](https://github.com/GeorgiaFrankinStain/SibGUTY_git/tree/master/5k1s/%D0%A1%D0%9F%D0%98%20-%20%D0%A1%D0%BE%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8%20(%D0%A4%D0%B8%D0%BE%D0%BD%D0%BE%D0%B2)/labs_realisation/lab_1-3)

https://github.com/GeorgiaFrankinStain/SibGUTY\_git/tree/master/5k1s/%D0%A1%D0%9F%D0%98%20-%20%D0%A1%D0%BE%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8%20(%D0%A4%D0%B8%D0%BE%D0%BD%D0%BE%D0%B2)/labs\_realisation/lab\_1-3

