Современные проблемы информатики

Задача №1

Кодирование целых чисел.

МГ-101 Тимофеев Д.А.

СОДЕРЖАНИЕ:

[СОДЕРЖАНИЕ: 1](#_Toc56778425)

[1. Задание 2](#_Toc56778426)

[2. Программный код 2](#_Toc56778427)

[2.1. Вспомогательный код 2](#_Toc56778428)

[2.2. Fi0 кодер 3](#_Toc56778429)

[2.3. Fi > 0 кодер 3](#_Toc56778430)

[2.4. Fi декодер 4](#_Toc56778431)

[3. Тесты 5](#_Toc56778432)

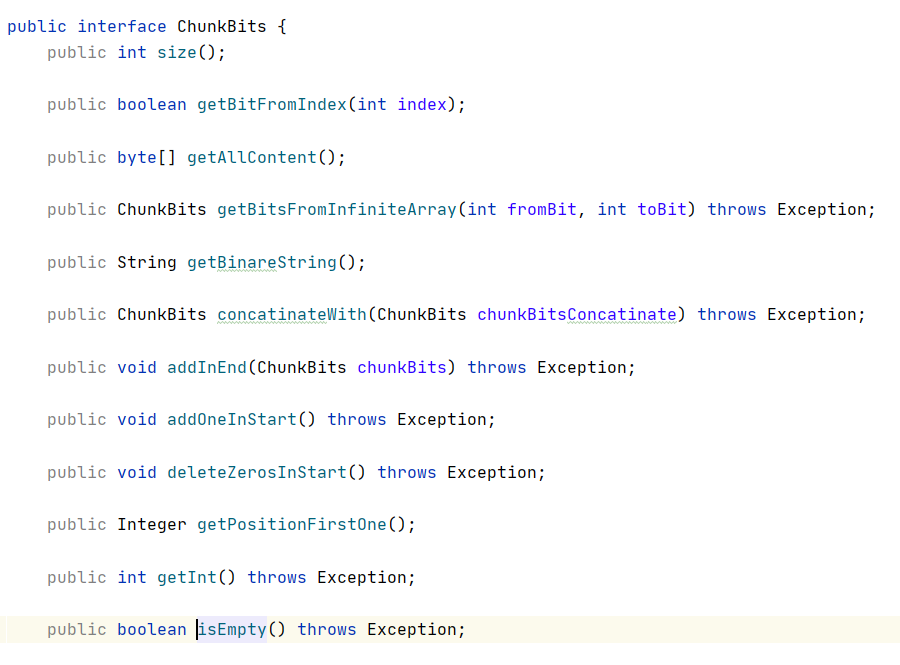
[4. Ссылка на исходники 7](#_Toc56778433)

1. Задание

Запрограммировать коды с соответствующими декодерами. Проверить корректность их работы на тестовых последовательностях целых чисел.

1. Программный код
   1. Вспомогательный код

Для кодирования порций бит, некратных байту был создан вспомогательный класс ChunkBits.



* 1. Fi0 кодер

|  |
| --- |
| public ChunkBits code(ChunkBits number) throws Exception {  if (number.equals(new ChunkBitsClass("0"))) {  return new ChunkBitsClass("1");  }   return f0(number); } private ChunkBits f0(ChunkBits number) throws Exception {  int lengthNumber = number.getInt();  assert(lengthNumber >= 0);   String codeNumber = "";  for (int i = 0; i < lengthNumber; i++) {  codeNumber += "0";  }   codeNumber += "1";  return new ChunkBitsClass(codeNumber); } |

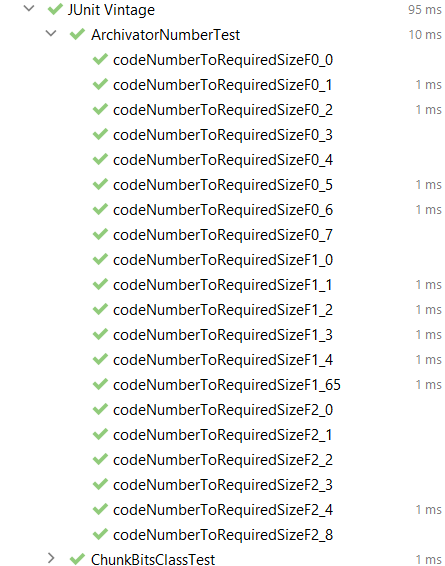
* 1. Fi > 0 кодер

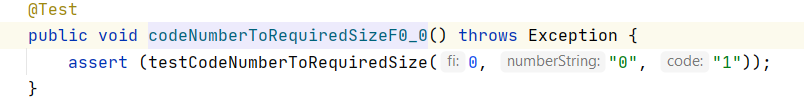
|  |
| --- |
| private ChunkBits codeFiMore0(int fi, ChunkBits number) throws Exception {  if (fi == 0) {  return this.fi0.code(number);  } else if (number.equals(new ChunkBitsClass("0"))) {  return new ChunkBitsClass("1");  } else if (number.equals(new ChunkBitsClass("1"))) {  return new ChunkBitsClass("01");  } else {  ChunkBits codeNumberBitsForRead = codeFiMore0(fi - 1, new ChunkBitsClass(number.size()));  ChunkBits withoutFirtOne = number.getBitsFromInfiniteArray(1, number.size());  return codeNumberBitsForRead.concatinateWith(withoutFirtOne);  } } |

* 1. Fi декодер

|  |
| --- |
| private ChunkBits decodeFi(int fi) throws Exception {   ChunkBits res = new ChunkBitsClass(null, 0);  Integer startReadNextData = fi0Decode();  if (startReadNextData == null) {  return res;  }   if (fi == 0 || (startReadNextData < 2)) {  this.endFirstCodeNumber = startReadNextData;  return new ChunkBitsClass(startReadNextData);  }   int sizeNextChunk = startReadNextData;  int endReadNextData = startReadNextData + sizeNextChunk;  startReadNextData++; *//потому что первая единица относиться к f0 //FIXME* for (int i = 0; i < fi; i++) {  res = buffer.getBitsFromInfiniteArray(startReadNextData, endReadNextData);  int impliedOneInStartInNextChunk = 1;  res.addOneInStart();  ChunkBits chunkWithImpliedOneInStart = res;  sizeNextChunk = chunkWithImpliedOneInStart.getInt() - impliedOneInStartInNextChunk;  startReadNextData = endReadNextData;  endReadNextData = startReadNextData + sizeNextChunk;  }    this.endFirstCodeNumber = startReadNextData;    return res; } |

1. Тесты





|  |
| --- |
| @Test public void codeNumberToRequiredSizeF0\_0() throws Exception {  assert (testCodeNumberToRequiredSize(0, "0", "1")); }  @Test public void codeNumberToRequiredSizeF0\_1() throws Exception {  assert (testCodeNumberToRequiredSize(0, "1", "01")); }  @Test public void codeNumberToRequiredSizeF0\_2() throws Exception {  assert (testCodeNumberToRequiredSize(0, "10", "001")); }  @Test public void codeNumberToRequiredSizeF0\_3() throws Exception {  assert (testCodeNumberToRequiredSize(0, "11", "0001")); }  @Test public void codeNumberToRequiredSizeF0\_4() throws Exception {  assert (testCodeNumberToRequiredSize(0, "100", "00001")); }  @Test public void codeNumberToRequiredSizeF0\_5() throws Exception {  assert (testCodeNumberToRequiredSize(0, "101", "000001")); }  @Test public void codeNumberToRequiredSizeF0\_6() throws Exception {  assert (testCodeNumberToRequiredSize(0, "110", "0000001")); }  @Test public void codeNumberToRequiredSizeF0\_7() throws Exception {  assert (testCodeNumberToRequiredSize(0, "111", "00000001")); }  @Test public void codeNumberToRequiredSizeF1\_0() throws Exception {  assert (testCodeNumberToRequiredSize(1, "0", "1")); }  @Test public void codeNumberToRequiredSizeF1\_1() throws Exception {  assert (testCodeNumberToRequiredSize(1, "1", "01 ")); }  @Test public void codeNumberToRequiredSizeF1\_2() throws Exception {  assert (testCodeNumberToRequiredSize(1, "10", "001 0")); }  @Test public void codeNumberToRequiredSizeF1\_3() throws Exception {  assert (testCodeNumberToRequiredSize(1, "11", "001 1")); }  @Test public void codeNumberToRequiredSizeF1\_4() throws Exception {  assert (testCodeNumberToRequiredSize(1, "100", "0001 00")); } @Test public void codeNumberToRequiredSizeF1\_65() throws Exception {  assert (testCodeNumberToRequiredSize(1, "1000001", "00000001 000001")); }  @Test public void codeNumberToRequiredSizeF2\_0() throws Exception {  assert (testCodeNumberToRequiredSize(2, "0", "1")); }  @Test public void codeNumberToRequiredSizeF2\_1() throws Exception {  assert (testCodeNumberToRequiredSize(2, "1", "01")); }  @Test public void codeNumberToRequiredSizeF2\_2() throws Exception {  assert (testCodeNumberToRequiredSize(2, "10", "001 0 0")); }  @Test public void codeNumberToRequiredSizeF2\_3() throws Exception {  assert (testCodeNumberToRequiredSize(2, "11", "001 0 1")); }  @Test public void codeNumberToRequiredSizeF2\_4() throws Exception {  assert (testCodeNumberToRequiredSize(2, "100", "001 1 00")); }  @Test public void codeNumberToRequiredSizeF2\_8() throws Exception {  assert (testCodeNumberToRequiredSize(2, "1000", "0001 00 000")); } |

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

