### Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Центр программной инженерии

ОТЧЕТ

к лабораторной работе №2 по дисциплине:

|  |
| --- |
| «Объектно-ориентированное программирование» |
| Разработка классов и их использование |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб 19-2 |  |  |  | Козлов М.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2020 г.

**Содержание**

[1 Постановка задачи 3](#_Toc54563915)

[2 Описание структуры классов 5](#_Toc54563916)

[3 Описание методов класса и спецификация локальных переменных 8](#_Toc54563917)

[4 Таблица тестов 11](#_Toc54563918)

[5 Листинг исходного кода 16](#_Toc54563919)

[Заключение 23](#_Toc54563920)

[Список использованных источников 24](#_Toc54563921)

1 Постановка задачи

Провести объектную декомпозицию задачи из индивидуального задания и разработать класс, содержащий соответствующие поля для хранения необходимых данных и методы, обеспечивающие достаточную для решения задачи функциональность класса. Создание объекта класса решающего задание и вызов его методов должны осуществляться из главного класса. Ввод и вывод данных осуществляется как в предыдущей лабораторной работе, для передачи исходных данных в экземпляр класса решающего задачу должны быть разработаны соответствующие методы.

Индивидуальное задание выполняется на основе заданий к лабораторной работе 1.

Вариант № 9.

a. Сгенерируйте числовой ряд длиной не более 50, представляющий собой числа Фибоначчи (каждый последующий элемент представляют суммы двух предыдущих, первые два элемента равны 1). Выведите на печать первые 10, отмечая четные числа каким-нибудь символом.

b. Дан массив чисел, знак операции и операнд. Выведите на печать массив, полученный в результате арифметического действия, совершенного над элементами массива.

c. Дана строка содержащая ряд чисел в двоичном виде, необходимо проверить упорядочены ли они.

*Вопросы и замечания по работе*

*1. Как выполняли объектную декомпозицию?+++*

*2. Что является объектом класса CompleteTasks? Значениями каких полей определяется состояние объектов указанного класса?+++*

*3. Какие методы изменяют состояние объектов класса CompleteTasks?+++*

*4. Для чего нужны геттеры и сеттеры?+++*

*5. Ответьте на теоретические вопросы:*

*- Конструкторы.+++*

*- Инкапсуляция. Как реализован этот принцип в Вашей программе?+++*

*6. Для чего нужна сериализация?+++*

*Также исправьте замечания по оформлению:*

*1. Весь текст в отчете кроме кода должен быть 14 пт (в том числе в таблицах в оглавлении). Отступ первой строки 1,25. Код должен быть Courier New, 10пт.+++*

*2. В тексте (кроме заголовков) интервал перед и после абзаца должен быть нулевым+++*

*3. Никакие объекты (ни рисунки, ни таблицы, ни текст) не должны выходить за поля. +++*

*4. В заголовках отступы первой строки должны быть 1,25*

*5. Пустых строк не должно быть+++*

*6. Текст должен быть выровнен по ширине +++*

*7. Заголовки Заключение и Список литературы не нумеруются и выравниваются по ширине. +++*

*8. Заголовок Содержание должен быть жирным +++*

1. Я выполнил объектную декомпозицию представлением решений задач A, B, C в виде объекта класса **CompleteTasks**.
2. **Объект класса** – это объявление **переменной**. **Тип** переменной – это есть тип с именем класса (**CompleteTasks**). При объявлении объекта класса создается **экземпляр класса**, в данной лабораторной работе **экземплярами** класса **CompleteTasks** являются: **cls, str1, str2, str3.**

**Состояние** объектов данного класса определяется значениями полей:

String taskA – решение задания А;

int[] taskB – решение задания B;

String[] taskC – решение задачи С

1. Состояния объектов класса **CompleteTasks** изменяют методы:

completeTaskA() – метод, решающий задачу А;

completeTasksB() – метод, решающий задачу В;

completeTasksC() – метод, решающий задачу С;

1. **Геттеры** **и** **сеттеры** - это методы, позволяющие получать значение полей класса из других объектов.

**Сеттеры -** это методы, позволяющие изменять значение полей класса из других объектов.

**Геттеры** **и** **сеттеры** используются для того, чтобы напрямую не обращаться к полю (не делая его public).

1. Теоретические вопросы:

* Конструкторы предоставляют способ задавать начальное состояние объекту. Например, **конструктор по умолчанию**, который определяет состояние объекта при отсутствии входных параметров.
* Конструктор - это метод, который вызывается в момент создания объекта и служит для инициализации его полей.
* Конструктор может иметь аргументы, через которые можно установить желаемые значения инициализируемым полям. В данной программе это конструктор с параметрами, вводимыми пользователем.

**Инкапсуляция** - это сокрытие реализации класса и отделение его внутреннего представления от внешнего.

В моей программе этот принцип реализован посредством:

* изменения состояния объекта только результатом вызова его поведения (методов)
* сокрытия собственных ресурсов класса модификатором доступа private (например, нельзя просто так задать размерность массива поля taskB равной 1 или 0). Кроме того, значение поле объекта может быть получено только с помощью **геттера**.
* изоляции пользователя от особенностей реализации (пользователю остаётся только ввести данные, чтобы получить результат, не вдаваясь в подробности технической реализации).

1. **Сериализация** - процесс перевода какой-либо структуры данных (объекта) в последовательность битов. Следовательно, задача требующая сохранения состояния объектов или групп объектов (любой сложности) сводится к простому сохранению последовательности бит данных, которую можно легко передавать, сохранять и при необходимости восстанавливать в исходном виде.

2 Описание структуры классов

Таблица 1 – Описание структуры классов

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя метода** | **Назначение** | **Класс** |
| 1 | main(String[]) | Точка входа в программу | L2V9 |
| 2 | userInput() | Функция, проверки ввода данных | L2V9 |
| 3 | outputAnswers(String,int[],String[]) | Функция, вывода решений задач | L2V9 |
| 4 | completeTaskA() | Функция, решающая задачу А | CompleteTasks |
| 5 | completeTaskВ() | Функция, решающая задачу В | CompleteTasks |
| 6 | completeTaskС() | Функция, решающая задачу С | CompleteTasks |
| 7 | binaryCodeToDecimal(String) | Вспомогательная функция решения задачи С: перевод двоичного числа в десятичное | CompleteTasks |
| 8 | CompleteTasks() | Конструктор по умолчанию | CompleteTasks |
| 9 | CompleteTasks (int[],int, string) | Конструктор с параметрами | CompleteTasks |
| 10 | CompleteTasks (CompleteTasks) | Конструктор копирования | CompleteTasks |
| 11 | setTasks (String, int[],String[]) | Метод, изменяющий значения полей на значения,  из файла, полученные путём десериализации. | CompleteTasks |
| 12 | getTaskA() | Метод, возвращающий данные поля taskA. | CompleteTasks |
| 13 | getTaskB() | Метод, возвращающий данные поля taskB. | CompleteTasks |
| 14 | getTaskC() | Метод, возвращающий данные поля taskC. | CompleteTasks |

**Словесное описание класса L2V9**

Класс L2V9 содержит в себе 3 статических метода: сам main, usetInput,outputAnswers. Их назначения приведены в таблице 1.

**Словесное описание класса CompleteTasks**:

Класс не статический, содержит 3 поля:

String – taskA (решение задачи А)

int[] – taskB (решение задачи B)

String[] – taskC (решение задачи C)

**Данные поля являются результатами работы методов, решающих поставленные задачи:**

taskA – результат функции completeTaskA()

taskB – результат функции completeTaskB()

taskC – результат функции completeTaskC()

Значения остальных методов приведены в таблице 1.

**Словесное описание конструктора копирования:**

Конструктор копирования реализован с использованием механизмов сериализации и десериализации объекта. При сериализации происходит перевод состояния объекта в последовательность байт и последующая запись их в файл “file.ser”, после происходит десериализация (восстановление) объекта из этих байт. После восстановления объекта вызывается метод setTasks (String ,int[],String[]), который изменяет значения основных полей класса на значения востановленного объекта.

3 Описание методов класса и спецификация локальных переменных

Таблица 2 – Описание методов класса и спецификация локальных переменных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Спецификатор** | **Входные данные** | **Назначение** | **Тип** | **Выходные данные** | **Назначение** | **Тип** |
| **Класс L2V9** | | | | | | | | |
| 1 | main | public | args | Аргументы командной строки | String[] | A | Решение задачи А. | String |
| B | Решение задачи B. | int[] |
| C | Решение задачи C. | String[] |
| 2 | userInput | public | message | Сообщение, уведомляющее о том, что нужно ввести. | String | i | Проверенное на корректность ввода значение. | int |
| 3 | outputAnswers | public | taskA | Решение задачи А. | String | taskA | Решение задачи А. | String |
| taskB | Решение задачи B. | int[] | taskB | Решение задачи B. | int[] |
| taskC | Решение задачи C. | String[] | taskC | Решение задачи C. | String[] |
| **Класс CompleteTasks** | | | | | | | | |
| **Поля класса** | | | | | | | | |
| 1 | taskA | private | - | Решение задачи А | String | - | - | - |
| 2 | taskB | private | - | Решение задачи B | int[] | - | - | - |
| 3 | taskC | private | - | Решение задачи C | String[] | - | - | - |
| **Методы класса** | | | | | | | | |
| 1 | CompleteTasks | public | - | - | - | - | - | - |
| 2 | CompleteTasks | public | Arr | Массив чисел для функции **completeTaskB()**, вызываемой в конструкторе | int[] | taskA | Решение задачи А | String |
| operand | Число для функции **completeTaskB()**, вызываемой в конструкторе | int | taskB | Решение задачи В | int[] |
| sign | Строка, означающая арифметическое действие, для функции **completeTaskB()**, вызываемой в конструкторе | String | taskC | Решение задачи С | String[] |
| 3 | CompleteTasks | public | str | Объект, для копированияметодом сериализации в файл “file.ser” | CompleteTasks | copy | Десериализованный объект из из файла “file.ser” | CompleteTasks |
| 4 | getTaskA | public | - | - | - | taskA | Решение задачи А | String |
| 5 | getTaskB | public | - | - | - | taskB | Решение задачи B | int[] |
| 6 | getTaskC | public | - | - | - | taskC | Решение задачи C | String[] |
| 7 | setTasks | public | taskA | Строка из десериализованного объекта | String | taskA | Решение задачи А скопированного объекта | String |
| taskB | Массив чисел из десериализованного объекта | int[] | taskB | Решение задачи В скопированного объекта | int[] |
| taskC | Массив строк из десериализованного объекта | String[] | taskC | Решение задачи С скопированного объекта | String[] |

4 Таблица тестов

Входные данные используются **только для решения задачи В**, для других задач данные задаются автоматически. Для задачи С данные задаются **случайным образом** при каждом запуске программы, следовательно, наборы тестов и результаты тестирования могут отличаться в решении задачи С.

Таблица 3 – Таблица тестов для решения задачи B

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Корректный ввод размерности массива, значений элементов массива, операнда и знака арифметического действия (умножение). | size = 2  array =  num = 4  sign = “\*” | Задание A: числовой ряд Фибоначи: 1 1 \*2 3 5 \*8 13 21 \*34 55  Задание B: 4 4  Задание С:  Сгенерированная строка двоичных чисел: 101101100,1110000011,1010001010,11100010,100000011  Упорядоченная строка двоичных чисел в десятичном представлении : 226,259,364,650,899  Упорядоченная строка двоичных чисел: 11100010,100000011,101101100,1010001010,1110000011 |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 3 | | | |
| 2 | Корректный ввод размерности массива, значений элементов массива, операнда и знака арифметического действия (сложение). | size = 2  array =  num = 4  sign = “+” | Задание A: числовой ряд Фибоначи: 1 1 \*2 3 5 \*8 13 21 \*34 55  Задание B: 5 5  Задание С:  Сгенерированная строка двоичных чисел: 1001110110,100111010,100001010,1000000101,1001101011  Упорядоченная строка двоичных чисел в десятичном представлении : 266,314,517,619,630  Упорядоченная строка двоичных чисел: 100001010,100111010,1000000101,1001101011,1001110110 |
| 3 | Корректный ввод размерности массива, значений элементов массива, операнда и знака арифметического действия (вычитание). | size = 2  array =  num = 4  sign = “-” | Задание A: числовой ряд Фибоначи: 1 1 \*2 3 5 \*8 13 21 \*34 55  Задание B: 4 4  Задание С:  Сгенерированная строка двоичных чисел: 10001100,101100010,1001010010,11001000,1000001110  Упорядоченная строка двоичных чисел в десятичном представлении : 140,200,354,526,594  Упорядоченная строка двоичных чисел: 10001100,11001000,101100010,1000001110,1001010010 |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 3 | | | |
| 4 | Корректный ввод размерности массива, значений элементов массива, операнда и знака арифметического действия  (деление). | size = 2  array =  num = 4  sign = “/” | Задание A: числовой ряд Фибоначи: 1 1 \*2 3 5 \*8 13 21 \*34 55  Задание B: 2 2  Задание С:  Сгенерированная строка двоичных чисел: 1100100001,110111010,100011111,1001011101,101111011  Упорядоченная строка двоичных чисел в десятичном представлении : 287,379,442,605,801  Упорядоченная строка двоичных чисел: 100011111,101111011,110111010,1001011101,1100100001 |
| 5 | Некорректный ввод размерности массива. | size = к | Неверный ввод! |
| 6 | Некорректный ввод размерности массива. | size<=0 | Неверный ввод! Размерность массива не может быть меньше или равной 0 |
| 7 | Некорректный ввод значений элементов массива, | size = 2  array = | Неверный ввод! |
| 8 | Некорректный ввод операнда | size = 2  array =  num = м | Неверный ввод! |
| 9 | Некорректный ввод операнда | size = 2  array =  num = 0 | Неверный ввод! Попробуйте ещё раз |
| 10 | Некорректный ввод знака арифметического действия | size = 2  array =  num = 4  sign = “м” | Неверный ввод арифметического знака! |

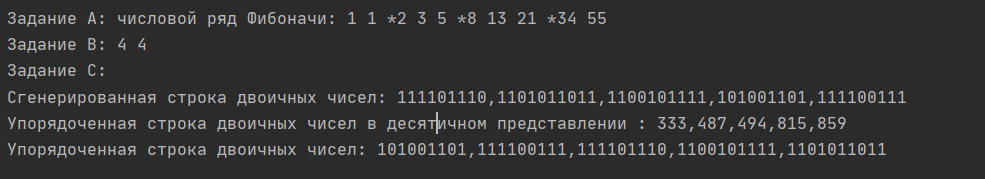


Рисунок 1 – Результат теста №1

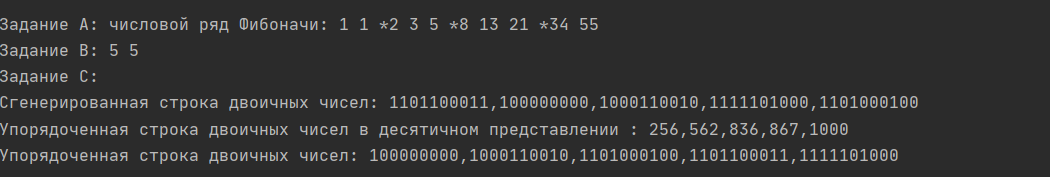


Рисунок 2 – Результат теста №2

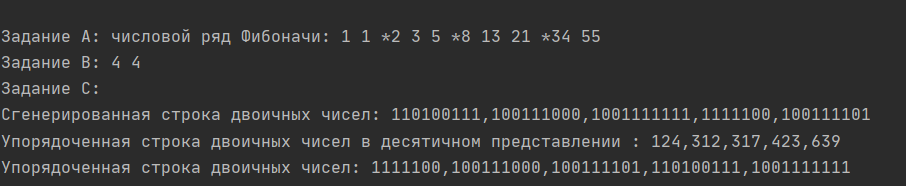


Рисунок 3 – Результат теста №3

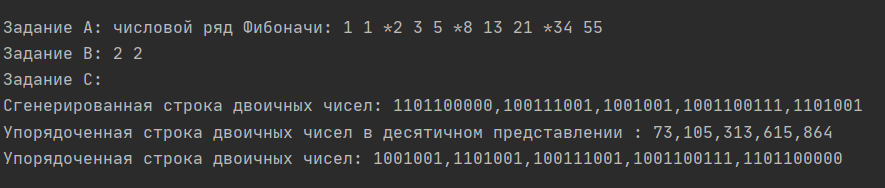


Рисунок 4 – Результат теста №4

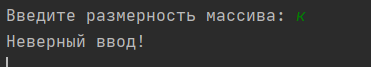


Рисунок 5 – Результат теста №5



Рисунок 6 – Результат теста №6

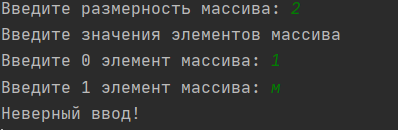


Рисунок 7 – Результат теста №7

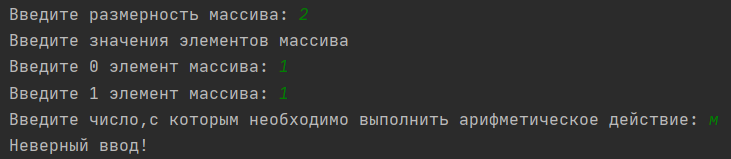


Рисунок 8 – Результат теста №8

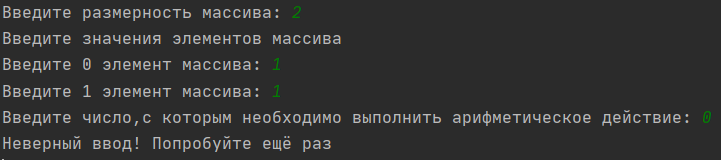


Рисунок 9 – Результат теста №9

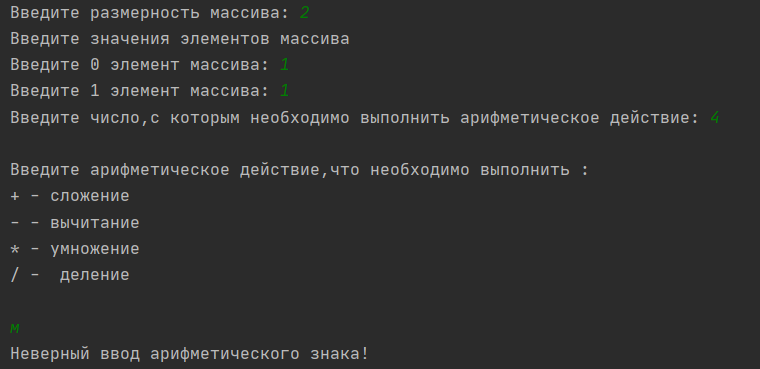


Рисунок 10 – Результат теста №10

5 Листинг исходного кода

Класс CompleteTasks

package com.company;  
import java.io.\*;  
  
public class CompleteTasks implements Serializable {  
 private String taskA;  
 private int[] taskB;  
 private String[] taskC;

public CompleteTasks(){ //Конструктор по умолчанию  
  
 this.taskA = "Недостаточно данных для решения задач!";  
 this.taskB = new int[]{0,0,0};  
 this.taskC = new String[]{"Недостаточно данных для решения задач!"};  
 }  
 public CompleteTasks(int[] Arr, int operand, String sign){ //Конструктор с параметрами  
 this.taskA = completeTaskA();  
 this.taskB = completeTaskB(Arr,operand,sign);  
 this.taskC = completeTaskC();  
 }  
 public CompleteTasks (CompleteTasks str) throws IOException, ClassNotFoundException// Конструктор копирования  
 {  
 // Сохранение объекта в файл  
 FileOutputStream file = new FileOutputStream("file.ser");  
 ObjectOutputStream out = new ObjectOutputStream(file);  
 // Метод сериализации объекта  
 out.writeObject(str);  
 out.close();  
 file.close();  
  
 CompleteTasks copy = null;  
 // Чтение объекта из файла  
 FileInputStream file1 = new FileInputStream("file.ser");  
 ObjectInputStream in = new ObjectInputStream(file1);  
 // Способ десериализации объекта  
 copy = (CompleteTasks) in.readObject();  
 in.close();  
 file1.close();  
 setTasks(copy.taskA, copy.taskB,copy.taskC);  
 }  
  
 public String getTaskA(){  
 return this.taskA;  
 }  
  
 public int [] getTaskB() {  
 return this.taskB;  
 }  
 public String[] getTaskC() {  
 return this.taskC;  
 }  
  
 public void setTasks(String taskA,int[]taskB,String[] taskC) {  
 this.taskA = taskA;  
 this.taskB = taskB;  
 this.taskC = taskC;  
 }  
  
 public String completeTaskA() {  
 int[] array = new int[50];  
  
 array[0] = array[1] = 1;  
  
 int i = 0;  
  
 while (i < 50 - 2) {  
 array[i + 2] = array[i + 1] + array[i];  
  
 i++;  
 }  
 i = 0;  
  
 String answerA = "";  
 while (i < 10) {  
 if (array[i] % 2 == 0) {  
 answerA+= "\*" + array[i] + " ";  
 } else {  
 answerA+=array[i] + " ";  
 }  
 i++;  
 }  
 answerA = "\nЗадание A: числовой ряд Фибоначи: "+ answerA;  
 return answerA;  
 }  
  
 public int[] completeTaskB(int[] Arr,int operand,String sign) {  
  
 switch (sign) {  
 case "+":  
 for (int i = 0; i < Arr.length; i++) {  
 Arr[i] = Arr[i] + operand;  
 }  
 break;  
 case "-":  
 for (int i = 0; i < Arr.length; i++) {  
 Arr[i] = Arr[i] - operand;  
 }  
 break;  
 case "\*":  
 for (int i = 0; i < Arr.length; i++) {  
 Arr[i] = Arr[i] \* operand;  
 }  
 break;  
 case "/":  
 for (int i = 0; i < Arr.length; i++) {  
 Arr[i] = Arr[i] / operand;  
 }  
 break;  
 }  
  
 return Arr;  
 }  
 public static int binaryCodeToDecimal(String str) {  
 char[] chars = str.toCharArray();  
 int result = 0;  
 int multiply = 1;  
 for (int i = str.length() - 1; i >= 0; i--) {  
 if (chars[i] == '1') {  
 result += multiply;  
 }  
 multiply \*= 2;  
 }  
 return result;  
 }  
  
 public String[] completeTaskC() {  
  
 String answerC[] =new String[3];  
 String str = "";  
  
 for (int i = 0; i < 5; i++) {  
 if (i != 4) {  
 str += Integer.*toBinaryString*(1 + (int) (Math.*random*() \* 1000)) + ",";  
 } else {  
 str += Integer.*toBinaryString*(1 + (int) (Math.*random*() \* 1000));  
 }  
 }  
 answerC[0]="\nСгенерированная строка двоичных чисел: " + str;  
  
  
 String[] StrToArr = str.split(",");  
  
 int[] OrderedArr = new int[StrToArr.length];  
  
 for (int i = 0; i < OrderedArr.length; i++) {  
 OrderedArr[i] = *binaryCodeToDecimal*(StrToArr[i]);  
 }  
  
 for (int i = 0; i < StrToArr.length - 1; i++) {  
 for (int j = i + 1; j < OrderedArr.length; j++) {  
 if (OrderedArr[i] > OrderedArr[j]) {  
 int temp = OrderedArr[i];  
 OrderedArr[i] = OrderedArr[j];  
 OrderedArr[j] = temp;  
 }  
 }  
 }  
 String BinaryString = "";  
 String Decimals = "";  
 for (int i = 0; i < OrderedArr.length; i++) {  
 if (i != OrderedArr.length - 1) {  
 Decimals += OrderedArr[i] + ",";  
 BinaryString += Integer.*toBinaryString*(OrderedArr[i]) + ",";  
 } else {  
 Decimals += OrderedArr[i];  
 BinaryString += Integer.*toBinaryString*(OrderedArr[i]);  
 }  
 }  
 answerC[1]="Упорядоченная строка двоичных чисел в десятичном представлении : " + Decimals;  
 answerC[2]="Упорядоченная строка двоичных чисел: " + BinaryString;  
 return answerC;  
 }  
  
}

Класс L2V9

package com.company;  
  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class L2V9 {  
 public static int userInput(String message) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print(message);  
 int i;  
 while (true) {  
 try {  
 i = Integer.*parseInt*(sc.next());  
 break;  
 } catch (NumberFormatException e) {  
 System.*out*.println("Неверный ввод!");  
 }  
 }  
 return i;  
 }  
 public static void main(String[] args) throws IOException, ClassNotFoundException {  
  
 Scanner input = new Scanner(System.*in*);  
 int size = *userInput*("Введите размерность массива: ");  
  
 while (size <= 0) {  
 size = *userInput*("Неверный ввод! Размерность массива не может быть меньше или равной 0\n");  
 }  
  
 int array[] = new int[size];  
 System.*out*.println("Введите значения элементов массива");  
 for (int i = 0; i < size; i++) {  
 array[i] = *userInput*("Введите " + (i) + " элемент массива: ");  
 }  
  
 int num = *userInput*("Введите число,с которым необходимо выполнить арифметическое действие: ");  
  
 while (num == 0) {  
 num = *userInput*("Неверный ввод! Попробуйте ещё раз\n");  
 }  
  
 System.*out*.println("\nВведите арифметическое действие,что необходимо выполнить : " + "\n" +  
 "+ - сложение" + "\n" +  
 "- - вычитание" + "\n" +  
 "\* - умножение" + "\n" +  
 "/ - деление" + "\n");  
 String sign;  
  
 sign = input.nextLine();  
  
 while (!sign.equals("+") && !sign.equals("-") && !sign.equals("\*") && !sign.equals("/")) {  
 System.*out*.println("Неверный ввод арифметического знака!");  
 sign = input.nextLine();  
 }  
  
 CompleteTasks cls = new CompleteTasks(array,num,sign);  
  
 System.*out*.println("Пример с введёнными данными");  
 *outputAnswers*(cls.getTaskA(),cls.getTaskB(),cls.getTaskC());  
  
  
  
  
 int[] test = new int[]{ 10,100,1000,10000,100000,1000000,10000000,100000000,1000000000,1000000005 };  
 CompleteTasks str1 = new CompleteTasks();  
 CompleteTasks str2 = new CompleteTasks(test,5,"/");  
 CompleteTasks str3 = new CompleteTasks(str2);  
  
 String A;  
 int[] B;  
 String[] C;  
  
 System.*out*.println("\n");  
 System.*out*.println("Пример с данными значениями");  
 System.*out*.println("\n");  
 System.*out*.println("Конструктор по умолчанию");  
 A = str1.getTaskA();  
 B = str1.getTaskB();  
 C = str1.getTaskC();  
 *outputAnswers*(A,B,C);  
 System.*out*.println("\n");  
 System.*out*.println("Конструктор с параметрами");  
 A = str2.getTaskA();  
 B = str2.getTaskB();  
 C = str2.getTaskC();  
 *outputAnswers*(A,B,C);  
  
 System.*out*.println("\n");  
 System.*out*.println("Конструктор копирования");  
 A = str3.getTaskA();  
 B = str3.getTaskB();  
 C = str3.getTaskC();  
 *outputAnswers*(A, B, C);  
 }  
 public static void outputAnswers(String taskA,int[]Arr,String[] taskC){  
 System.*out*.println(taskA);  
 System.*out*.print("Задание B: ");  
 for(int i = 0;i< Arr.length;i++){  
 System.*out*.print(Arr[i]+" ");  
 }  
 System.*out*.print("\nЗадание С: ");  
 for(int i = 0;i< taskC.length;i++){  
 System.*out*.println(taskC[i]+" ");  
 }  
 }  
}

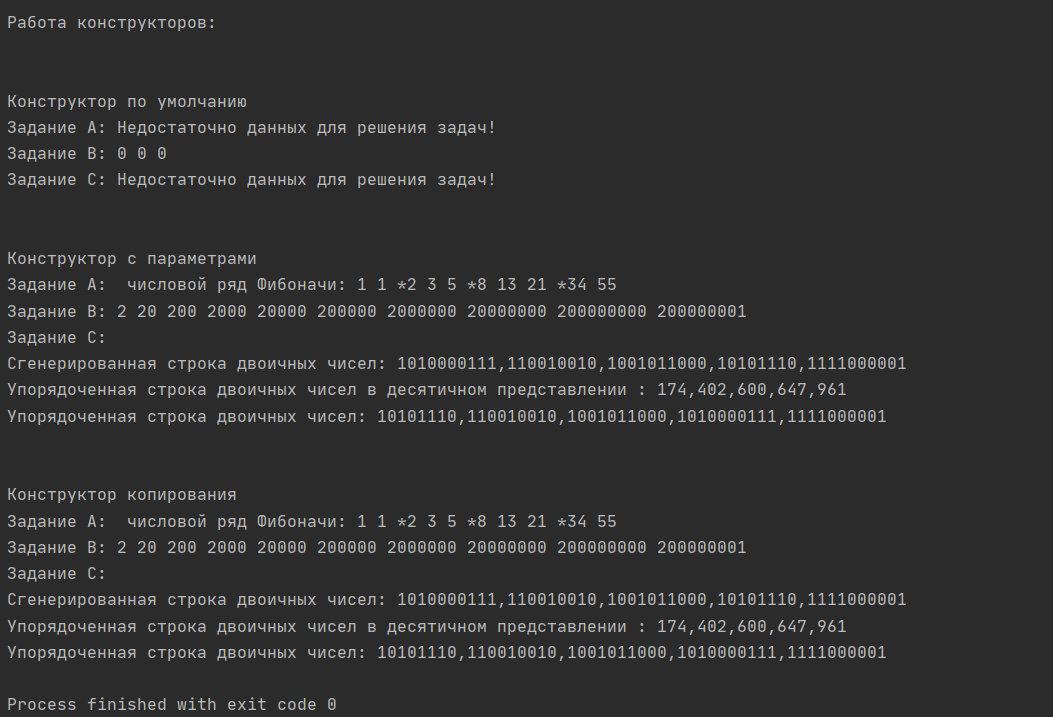


Рисунок 11 – Результат работы конструкторов

В данном примере конструктор копирования копирует состояние объекта конструктора с параметрами.

Заключение

Знакомство с интегрированной средой разработки приложений IDE IntelliJ IDEA прошло успешно. На практике были применены элементы объектного подхода: применена объектная декомпозиция, инкапсуляция механизма реализации объекта. Был получен практический опыт написания и реализации классов, создания объектов, вызова их методов, работа с модификаторами доступа, так же был получен опыт работы с механизмами сериализации и десериализации. Все тесты были успешно пройдены, ошибок или не корректной обработки данных не выявлено.

Список использованных источников

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: <http://developer.alexanderklimov.ru/android/java/java.php> (дата обращения 12.09.20)

2. API Specification for the Java 7 SE. [официальный сайт] URL: <http://docs.oracle.com/javase/7/docs/api/> (дата обращения 12.09.20)

Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированны анализ и проек-тирование с примерами приложений. Третье издание. М.: "Вильямс", 2010, -720 с.

3. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014. – 768 с.

4. Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров. Базовый курс по объектно-ориентированному программированию / А. Н. Васильев . – СПб.: Питер, 2012. – 395 с.