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

наименование института

ОТЧЕТ  
к лабораторной работе по дисциплине

Объектно-ориентированное программирование

Разработка классов и их использование

Наименование темы

Выполнил студент группы ИСТб-19–2 Комогорцева Ю.В.

шифр Фамилия И.О.

Проверила Маланова Т.В.

Фамилия И.О.

Содержание отчета на 17 стр.

Иркутск 2020 г.

Содержание

[1 Постановка задачи 3](#_Toc54807637)

[2 Описание структуры классов. 4](#_Toc54807638)

[3 Описание методов классов и спецификация локальных переменных. 5](#_Toc54807639)

[4 Таблица тестов 7](#_Toc54807640)

[5 Результаты тестирования 9](#_Toc54807641)

[6 Листинг исходного кода 12](#_Toc54807642)

[Заключение 17](#_Toc54807643)

[Список литературы 18](#_Toc54807644)

1. Постановка задачи

Разработать консольное приложение на языке java, реализующее решение задачи в соответствии с индивидуальным заданием. Исходные данные для задачи вводятся как параметры при запуске программы из командной строки, либо – как значения, инициализирующие переменные. Вывод результатов осуществляется в консоль.

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.

Вариант 11

1. Выберите из предложения слово, которое располагается в центре предложения, то есть символ, входящий в его состав, является центральным символом в предложении (с учетом пробелов).
2. Найдите символы в строке, который встречается менее 10%.
3. Найдите длину первого слова в предложении, а затем выберите из предложения все слова той же длины.
4. Описание структуры классов.

**class Main**

**class Strings**

Рисунок 1-Описание структуры классов

1. Описание методов классов и спецификация локальных переменных.

Таблица 1- Описание методов классов и спецификация переменных

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя** | **Описание** | **Тип данных** |
| **public class Main** | | | |
| **public static void main(String[] args)** | | | |
| 1 | String1 | Обьекты класса Strings ,с необходимыми данными для решения задач | Strings |
| 2 | String2 |
| 3 | String3 |
| **class Strings** | | | |
| 4 | S | Глобальная переменная класса для удобной работы с методами | String |
| **public static int GetCenter()** | | | |
| 5 | center | Центральный индекс | int |
| **public static String getResult1(String S)** | | | |
| 6 | center | Центральный индекс | int |
| 7 | results | Центральное слово/символ | String |
| 8 | rez | Строка с итоговым результатом | String |
| **public static String getResult2(String S)** | | | |
| 9 | mas | Преобразованная строка в массив символов | char[] |
| 10 | result | Массив для результата с итоговыми символами | char[] |
| 11 | t | Переменная отвечающая за наличие повторяющихся символов | int |
| 12 | i,j | Переменные для цикла | int |
| 13 | number | Счетчик количества повторений какого либо символа | double |
| 14 | rez | Строка с итоговым результатом | String |
| **public static String getResult3(String S)** | | | |
| 15 | str | Массив слов взятых из строки | String[] |
| 16 | length | Длинна первого слова | int |
| 17 | t | Счетчик количества искомых слов | int |
| 18 | i | Переменная для цикла | int |
| 19 | rez | Строка с итоговым результатом | String |

**Сlass Strings**

1. private String S;
2. public Strings(String S)ꟷконструктор с параметрами
3. public Strings()ꟷ конструктор по умолчанию
4. public Strings(Strings String) ꟷ конструктор копирования
5. public String getS()ꟷВозвращение атрибута
6. public void setS(String s) изменение атрибута
7. public static int GetCenter(String S) ꟷметод возвращающий центральный индекс в строке
8. public static String getResult1(String S) ꟷметод выполняющий задание 1, находящий слово которое содержит центральный символ в строке
9. public static String getResult2(String S) ꟷ метод выполняющий задание 2, находит символы которые повторяются меньше 10%
10. public static String getResult3(String S) ꟷметод выполняющий задание 3, находит слова одинаковые по длинне с первым словом
11. Таблица тестов

Таблица 2 – Таблица тестов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Описание** | **Входные данные** | **Выходные данные** | **Примечания** | **Код** |
| Тестирование конструкторов | | | | | |
| 1 | Тестирование конструктора по умолчанию | “beautiful beach” | test1(адрес;содержимое): com.company.Strings@4769b07b ; beautiful beach | Заданы конструктором по умолчанию | Strings test1=new Strings(); |
| 2 | Тестирование конструктора с параметрами | “Это проверка” | test2(адрес;содержимое): com.company.Strings@cc34f4d ; Это проверка | Заданы конструктором с параметрами | Strings test2=new Strings("Это проверка"); |
| 3 | Тестирование конструктора копирования | “beautiful beach” | test3(адрес;содержимое): com.company.Strings@17a7cec2 ; beautiful beach | Заданы конструктором копирования(выполняется если в сравнение с копируемым элементом содержимое одинаковое, а адреса разные) | Strings test3=new Strings(test1); |
| Задача 1 | | | | | |
| 1 | Штатный тест на срабатывание программы-есть центральное слово | Строка:I am not young enough to know everything | Центральное слово: enough | Входные данные заданы конструктором с параметром | Strings String1=new Strings("I am not young enough to know everything"); |
| 2 | Штатный тест на срабатывание программы-нет центрального слова | Строка :white green | Центрального слова не существует | Входные данные заданы конструктором с параметром | Код из теста 1 заменить на:  Strings String1=new Strings("white green "); |
| Задача 2 | | | | | |
| 1 | Штатный тест на срабатывание программы-есть подходящие символы | Строка:beautiful beach | Повторяющиеся символы:  Символ: t  Символ: i  Символ: f  Символ: l  Символ:  Символ: c  Символ: h | Входные данные заданы конструктором по умолчанию | Strings String2=new Strings();  Класс Strings:  //..  this.S = "beautiful beach";  //.. |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | Штатный тест на срабатывание программы-нет подходящих символов | Строка:111 222 | Нет повторяющихся символов меньше 10% | Входные данные заданы конструктором по умолчанию | Код из теста 1 заменить на:  //..  this.S = "111 222";  //.. |
| Задача 3 | | | | | |
| 1 | Штатный тест на срабатывание программы –есть слова одинаковой длинны как и первое | Строка: Life is like riding a bicycle . To keep your balance you must keep moving | Слова такой же длинны как и первое (Life) :  like  keep  your  must  keep | Входные данные заданы конструктором с параметром | Strings String3=new Strings("Life is like riding a bicycle . To keep your balance you must keep moving"); |
| 2 | Штатный тест на срабатывание программы – нет слов одинаковой длинны как и первое | Строка: tomorrow is another day | Сообщение: "Слов одинаковой длинны не существует." | Входные данные заданы конструктором с параметром | Код из теста 1 заменить на:  Strings String3=new Strings("tomorrow is another day "); |

1. Результаты тестирования

В таблице 3 приведены результаты тестов из таблицы 2.

Для удобства, в листинге кода присутствует код тестирования конструкторов. Он находится в конце класса Main в виде комментария .

Таблица 2 – Таблица результатов тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Результат |
| **Конструкторы копирования** | | | |
| 1 | “beautiful beach” | test1(адрес;содержимое): com.company.Strings@4769b07b ; beautiful beach | Тест пройден |
| 2 | “Это проверка” | test2(адрес;содержимое): com.company.Strings@cc34f4d ; Это проверка | Тест пройден |
| 3 | “beautiful beach” | test3(адрес;содержимое): com.company.Strings@17a7cec2 ; beautiful beach | Тест пройден |
| **Задача 1** | | | |  |  | Тест пройден |
| 1 | Строка:I am not young enough to know everything | Центральное слово: enough | Тест пройден |
| 2 | Строка :white green | Центрального слова не существует | Тест пройден |
| **Задача 2** | | | |
| 1 | Строка:beautiful beach | Повторяющиеся символы:  Символ: t  Символ: i  Символ: f  Символ: l  Символ:  Символ: c  Символ: h | Тест пройден |
| 2 | Строка:111 222 | Нет повторяющихся символов меньше 10% | Тест пройден |
| **Задача 3** | | | |
| 1 | Строка: Life is like riding a bicycle . To keep your balance you must keep moving | Слова такой же длинны как и первое (Life) :  like  keep  your  must  keep | Тест пройден |
| 2 | Строка: tomorrow is another day | Сообщение: "Слов одинаковой длинны не существует." | Тест пройден |

Результаты выполнения тестов с приведены на рисунках с 2 по 9 соответственно.

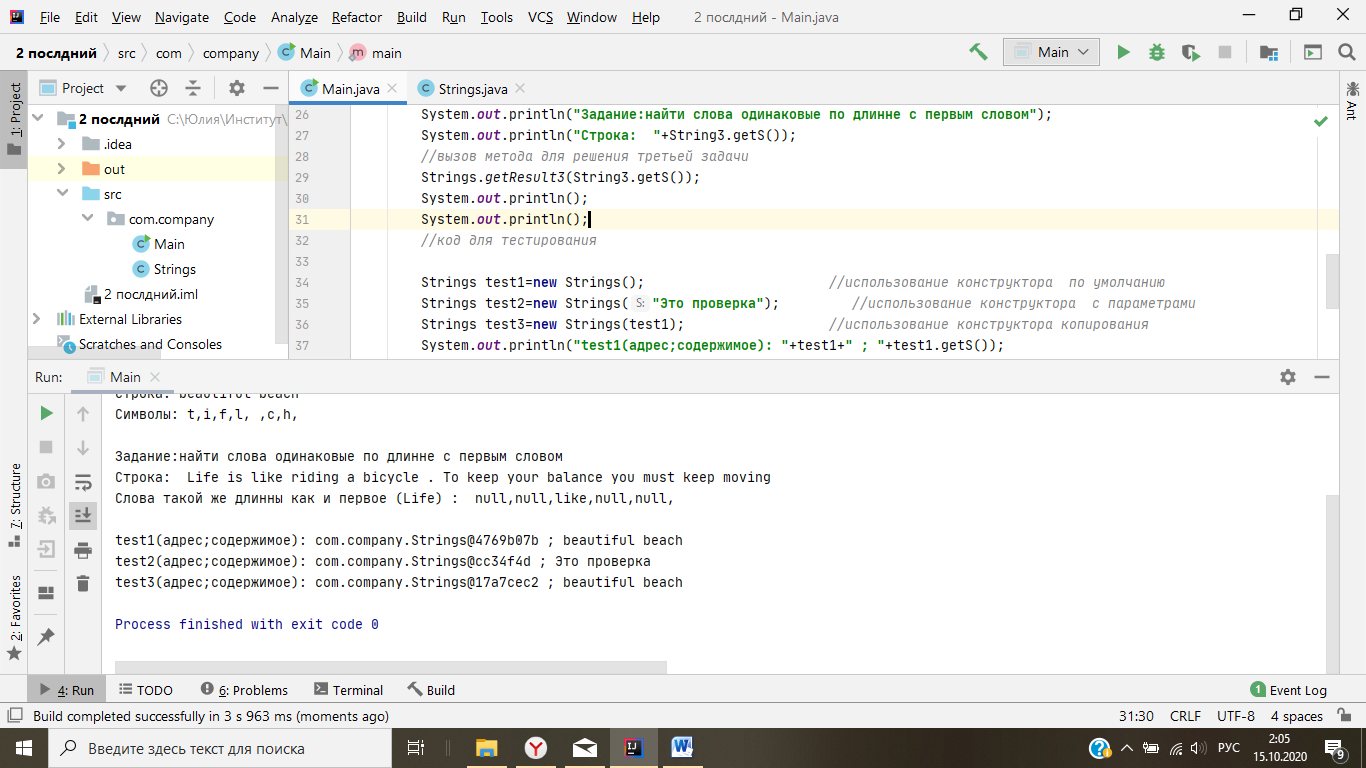


Рисунок 2 — Результат выполнения тестов 1, 2, 3 раздела тестирования конструкторов

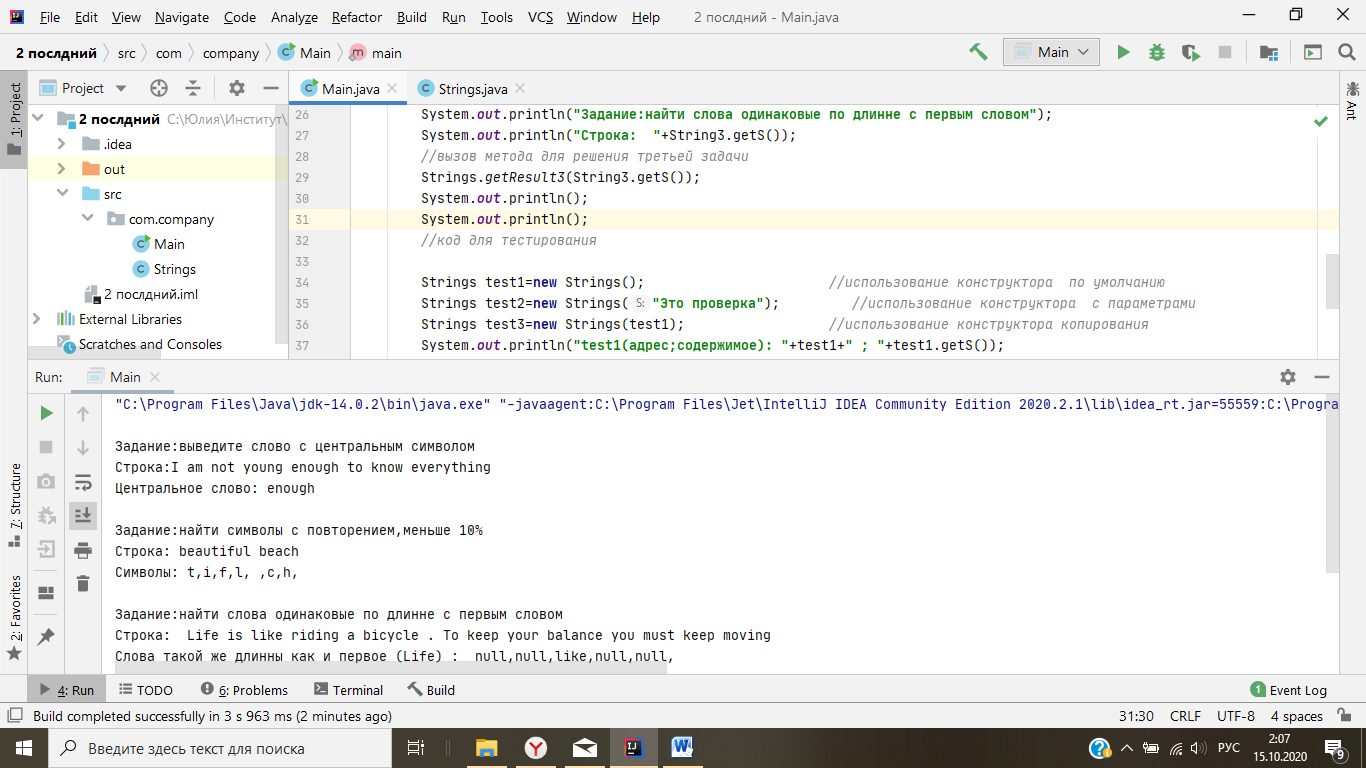


Рисунок 3 — Результат выполнения теста 1 задачи 1

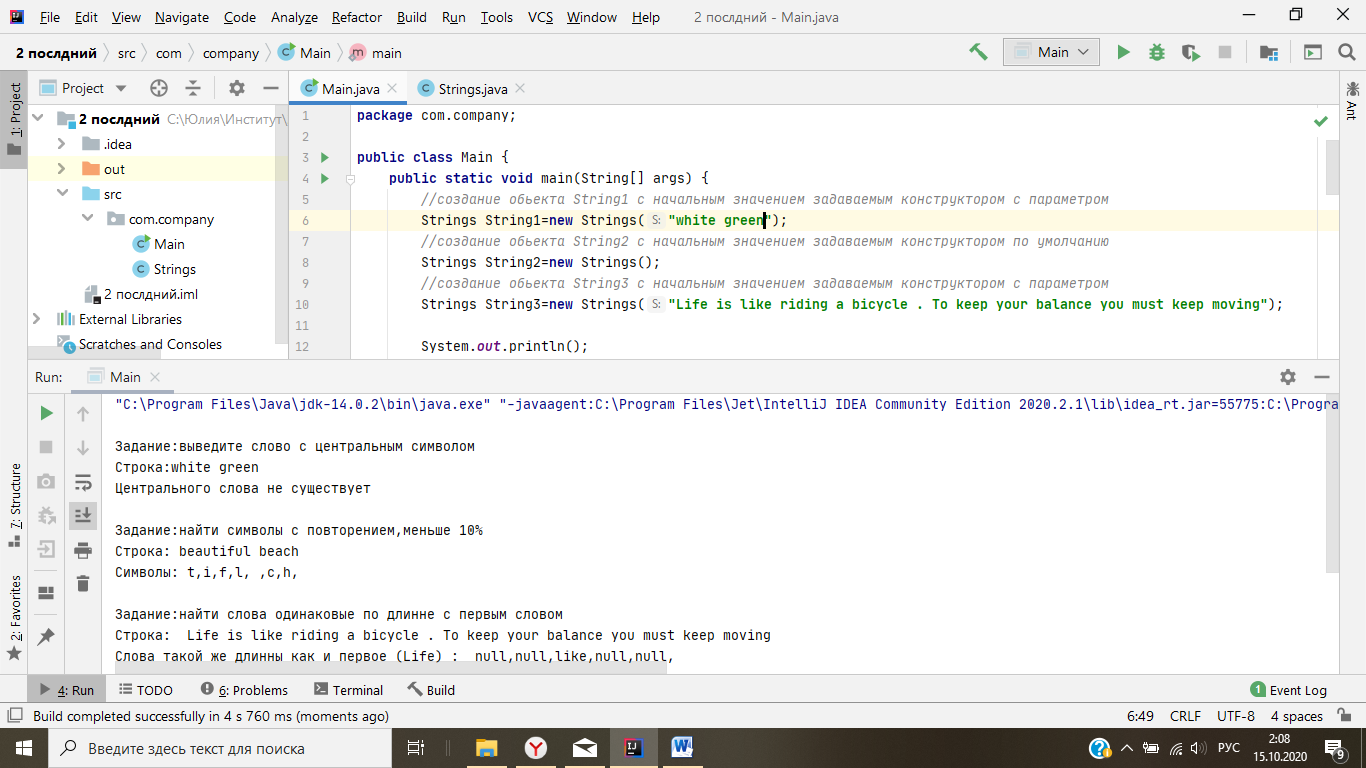


Рисунок 4 — Результат выполнения теста 2 задачи 1

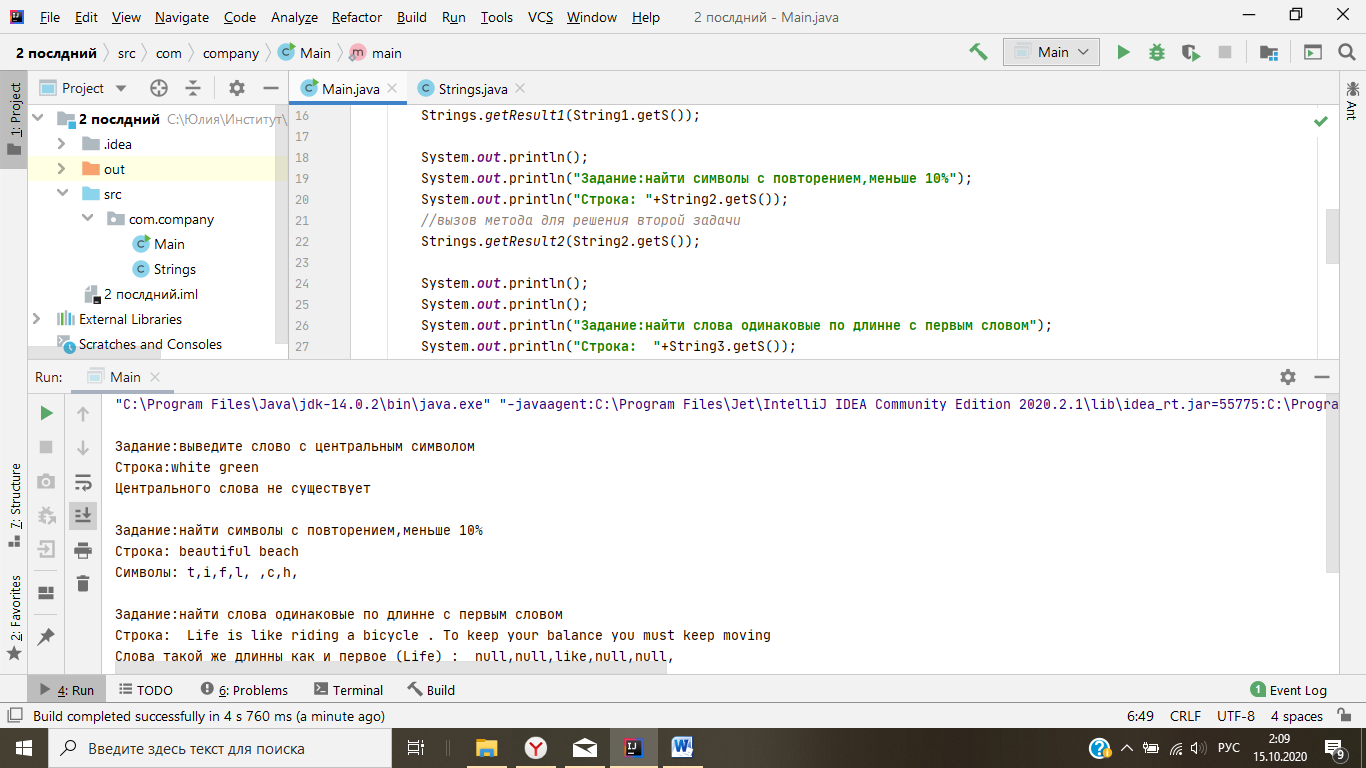


Рисунок 5 — Результат выполнения теста 1 задачи 2

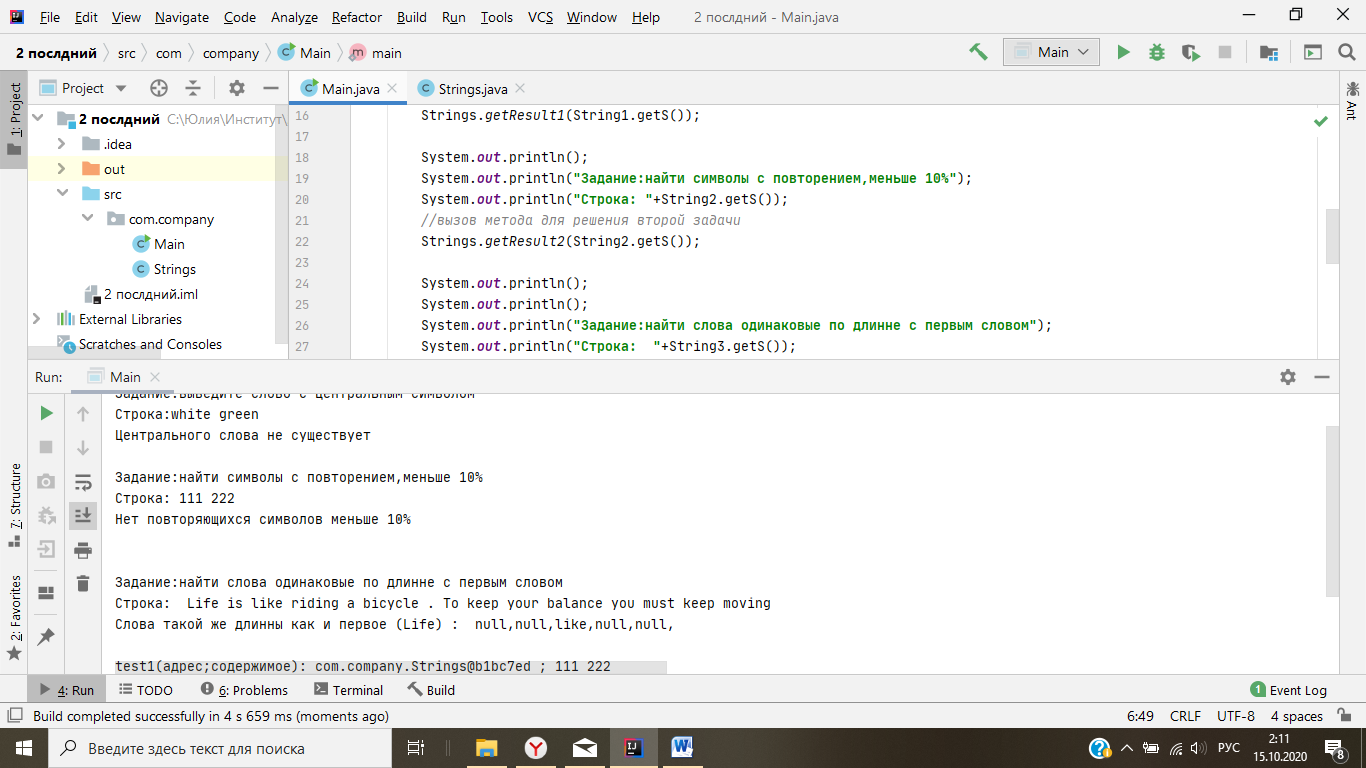


Рисунок 6 — Результат выполнения теста 2 задачи 2

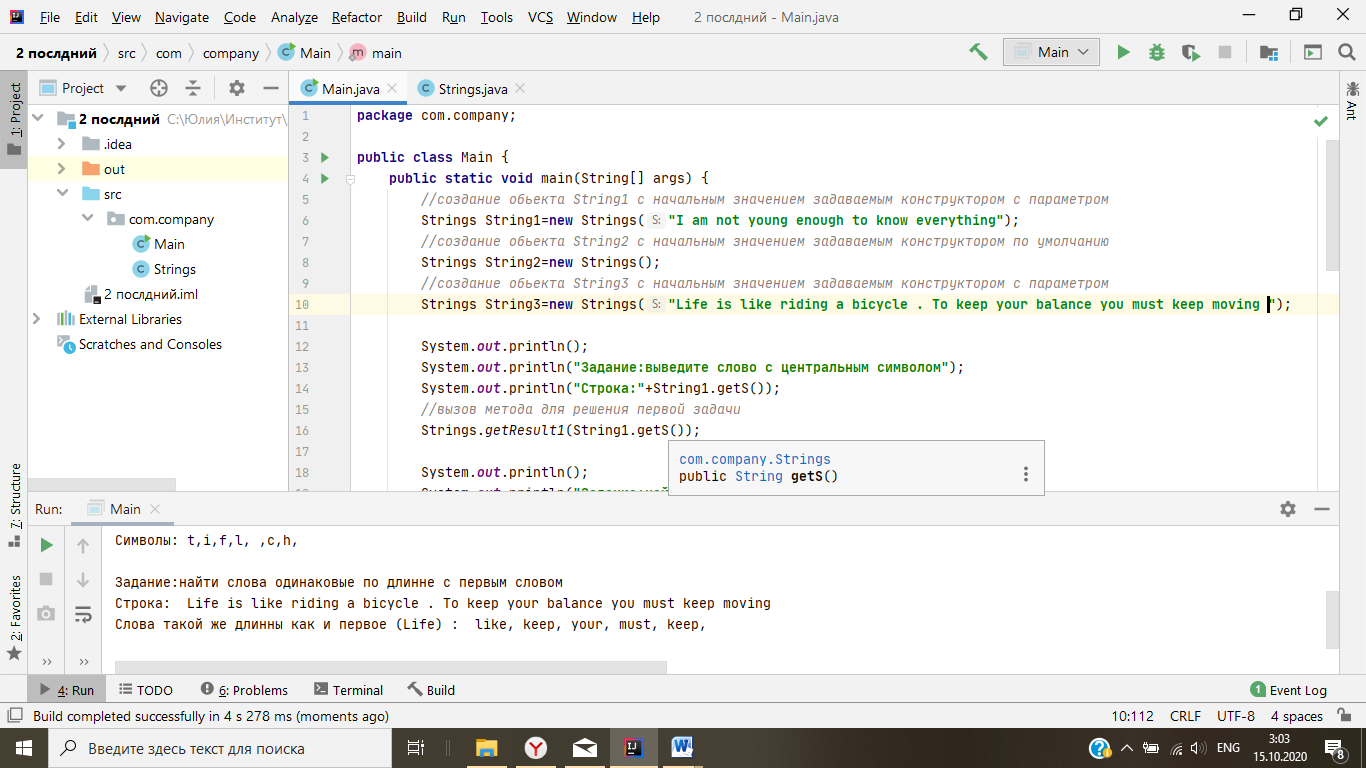


Рисунок 7 — Результат выполнения теста 1 задачи 3

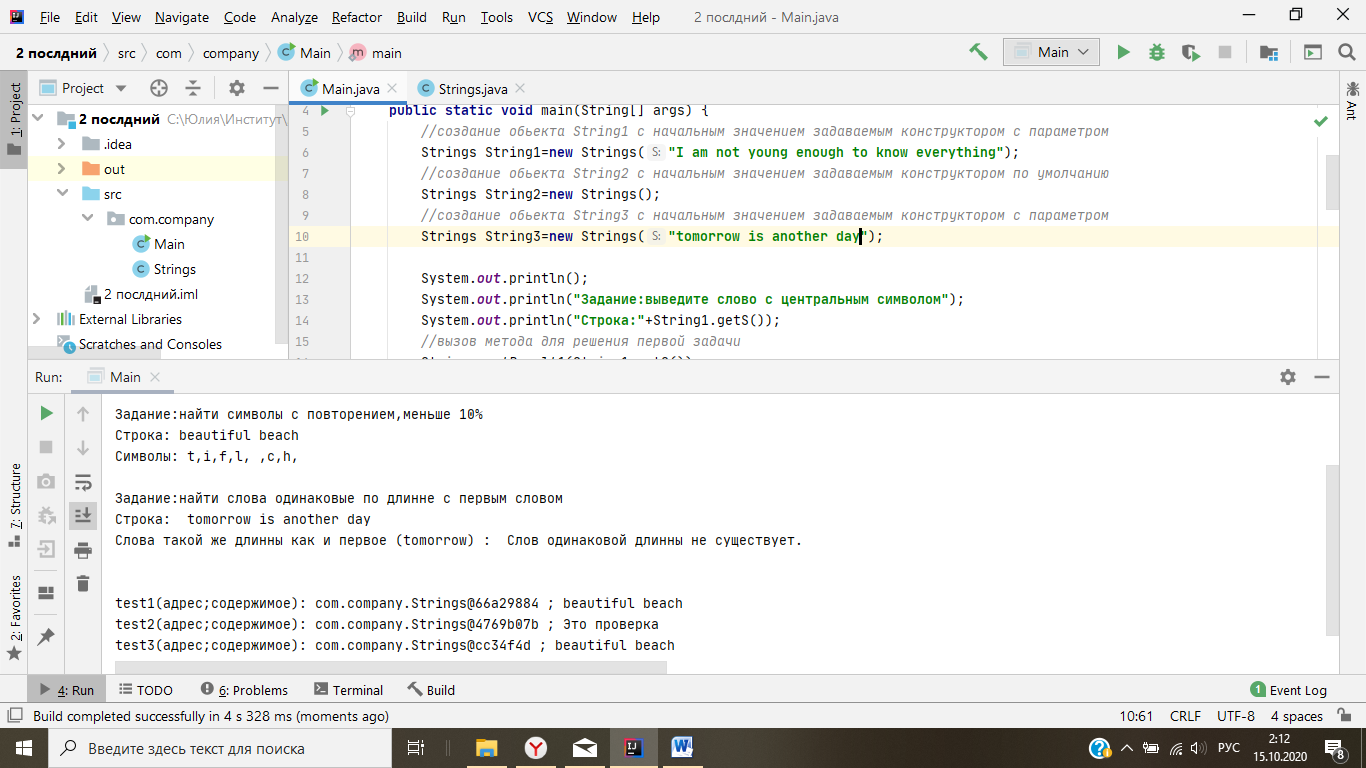


Рисунок 8 — Результат выполнения теста 2 задачи 3

1. Листинг исходного кода

**Класс Main**

**package** com.company;  
  
**public class** Main {  
 **public static void** main(String[] args) {  
 *//создание обьекта String1 с начальным значением задаваемым конструктором с параметром* Strings String1=**new** Strings(**"I am not young enough to know everything"**);  
 *//создание обьекта String2 с начальным значением задаваемым конструктором по умолчанию* Strings String2=**new** Strings();  
 *//создание обьекта String3 с начальным значением задаваемым конструктором с параметром* Strings String3=**new** Strings(**"Life is like riding a bicycle . To keep your balance you must keep moving "**);  
  
 System.***out***.println();  
 System.***out***.println(**"Задание:выведите слово с центральным символом"**);  
 System.***out***.println(**"Строка:"**+String1.getS());  
 *//вызов метода для решения первой задачи* System.***out***.println(Strings.*getResult1*(String1.getS()));  
  
 System.***out***.println();  
 System.***out***.println(**"Задание:найти символы с повторением,меньше 10%"**);  
 System.***out***.println(**"Строка: "**+String2.getS());  
 *//вызов метода для решения второй задачи* System.***out***.println(Strings.*getResult2*(String2.getS()));  
  
 System.***out***.println();  
 System.***out***.println();  
 System.***out***.println(**"Задание:найти слова одинаковые по длинне с первым словом"**);  
 System.***out***.println(**"Строка: "**+String3.getS());  
 *//вызов метода для решения третьей задачи* System.***out***.println(Strings.*getResult3*(String3.getS()));  
  
 *//код для тестирования  
 /\*  
 Strings test1=new Strings(); //использование конструктора по умолчанию  
 Strings test2=new Strings("Это проверка"); //использование конструктора с параметрами  
 Strings test3=new Strings(test1); //использование конструктора копирования  
 System.out.println("test1(адрес;содержимое): "+test1+" ; "+test1.getS());  
 System.out.println("test2(адрес;содержимое): "+test2+" ; "+test2.getS());  
 System.out.println("test3(адрес;содержимое): "+test3+" ; "+test3.getS());  
 \*/* }  
}

**Класс Strings**

**package** com.company;  
  
**class** Strings {  
 **private** String **S**;  
  
 **public** Strings(String S) { *//конструктор с параметрами* **this**.**S** = S;  
 }  
 **public** Strings() { *//конструктор по умолчанию* **this**.**S** = **"beautiful beach"**;  
 }  
 **public** Strings(Strings S) { *//конструктор копирования* **this**.**S** = S.**S**;  
 }  
  
 **public** String getS() { *//Возвращение атрибута* **return S**;  
 }  
 **public void** setS(String s) { *//изменение атрибута* **this**.**S** = s;  
 }  
 **public static int** GetCenter(String S){ *//метод возвращающий центральный индекс в строке* **int** center;  
 **if**(S.length()%2==0) *//Если длинна строки четная то* center =(S.length()/2)-1; *//центральный индекс равен половине длинне строки-1* **else** *//иначе* center =(S.length()/2); *//центральный индекс равен половине длинне строки* **return** center;  
 }  
 **public static** String getResult1(String S) { *//метод выполняющий задание 1* String rez=**""**;  
 **int** center=*GetCenter*(S); *//нахождение центрального индекса  
 //нахождение слова из строки от центрального символа влево и вправо до пробела* String results = S.substring(S.lastIndexOf(**" "**, center), S.indexOf(**" "**, center));  
 **if** (results.indexOf(**" "**) == -1) { *//если центральный символ равен пробелу* rez=**"Центрального слова не существует"**;*//то центрального слова не существует* }  
 **else** { *//иначе* rez=**"Центральное слово:"** + results; *//центральное слово найдено* }  
 **return** rez;  
 }  
 **public static** String getResult2(String S){ *//метод выполняющий задание 2* String rez=**""**;  
 **char**[] mas = S.toCharArray(); *//преобразование строки в массив символов* **char**[] result = **new char**[S.length()]; *//создание массива для итоговых символов* **int** t = 0; *//переменная отвечающая за наличие повторяющихся символов* **for** (**int** i = 0; i < S.length(); i++) { *//цикл на перебор символов в строке* **double** number = 0; *//переменная количества повторяющейся буквы* **for** (**int** j = 0; j < S.length(); j++) { *//цикл на перебор символов в строке* **if** (mas[i] == mas[j]) { *//если символы равны* number++; *//то количество повторений увеличивается на 1* }  
 }  
 **if** ((100 \* number / S.length()) < 10) *//если процент повторений меньше 10* {  
 result[t] = mas[i]; *//то символ записываем в массив с результатом* t = t + 1; *//увеличиваем наличие повторяющихся символов меньше 10%* }  
 }  
 **if** (t == 0) { *//если нет повторяющихся символов меньше 10%* rez=**"Нет повторяющихся символов меньше 10%"**;*//то сообщаем об этом* }  
 **else** { *//иначе* rez=**"Символы: "**; *//вывод массива с повторяющимися символами меньше 10%* **for** (**int** i = 0; i < t; i++) {  
 rez=rez+result[i] + **","**;  
 }  
 }  
 **return** rez;  
 }  
 **public static** String getResult3(String S) { *//метод выполняющий задание 3* String rez=**""**;  
 String[] str = S.split(**" "**); *//создаем массив слов взятых из строки* **int** length = str[0].length(); *//длинна первого слова* rez=**"Слова такой же длинны как и первое ("**+str[0]+**") : "**;  
 **int** t=0; *//счетчик для наличия искомых слов* **for**(**int** i = 1; i < str.**length**; i++) *//Цикл перебора слов до окончания строки* {  
 **if** (length == str[i].length()) *//если длинна первого слова равна длине выбранного слова* {  
 rez=rez+str[i]+**", "**;  
 t=t+1; *//счетчик увеличивается* }  
 }  
 **if** (t==0) { *//если счетчик для наличия искомых слов равен 0* rez=**"Слов одинаковой длинны не существует."**; *//то слов одинаковой длинны не существует* }  
 **return** rez;  
 }  
}

Заключение

В процессе выполнения лабораторной работы 2: “ Разработка классов и их использование” была проведена декомпозиция задачи из индивидуального задания и разработан класс, содержащий соответствующие поля для хранения необходимых данных и методы, обеспечивающие достаточную для решения задачи функциональность класса. Было сформировано 9 тестов, покрывающих все ситуации. Составленная на языке Java программа, реализующая разработанные алгоритмы, прошла все тесты успешно.

В ходе лабораторной работы осуществлялось применение элементов объектного подхода на практике: выполнение объектной декомпозиции, инкапсуляция механизмов реализации объектов

Мы получили практический опыт описания и реализации классов, создание объектов, вызова их методов, работы с модификаторами доступа.

Все цели практики достигнуты.

Ответы на индивидуальные вопросы:

1. Как выполняли объектную декомпозицию?

Ответ: Разделила задачу на несколько менее сложных подзадач. Они должны быть максимально простой структуры и независимы. Что бы можно было отдельно работать с каждой подзадачей в будущем. Всё необходимое

1. Что является объектом класса Strings? Значениями каких полей определяется состояние объектов указанного класса?

Ответ: Обьект класса Strings-это Strings. Поле S (строка) определяет состояние объектов класса Strings.

1. Так как объектная декомпозиция нужна для сложных программ, в которых методы используются каким-либо образом, то нужно, чтобы методы возвращали конкретное значение, а не вывод на консоль строки.

Исправила

1. Что будет при создании объекта, если разработчик класса не создать ни одного конструктора?

Ответ: Компилятор Java сам создаст конструктор по умолчанию (default constructor), который будет пустым и не делает ничего, кроме вызова конструктора суперкласса.

В случае если явно класса предка не указано, а по умолчанию все классы Java наследуют класс Object поэтому вызывается конструктор класса Object.

1. Ответьте на теоретические вопросы:

- Модификаторы доступа.

Все члены класса в языке Java - поля и методы - имеют модификаторы доступа. Модификаторы доступа позволяют задать допустимую область видимости для членов класса, то есть контекст, в котором можно употреблять данную переменную или метод.

В Java используются следующие модификаторы доступа:

public: публичный, общедоступный класс или член класса. Поля и методы, объявленные с модификатором public, видны другим классам из текущего пакета и из внешних пакетов.

private: закрытый класс или член класса, противоположность модификатору public. Закрытый класс или член класса доступен только из кода в том же классе.

protected: такой класс или член класса доступен из любого места в текущем классе или пакете или в производных классах, даже если они находятся в других пакетах

Модификатор по умолчанию. Отсутствие модификатора у поля или метода класса предполагает применение к нему модификатора по умолчанию. Такие поля или методы видны всем классам в текущем пакете.

- Сигнатура метода.

Весь код, который описывает метод, называется объявлением метода. Сигнатура метода включает название метода и типы параметров в определенном порядке:

*<* *модификатор доступа >,<тип возвращаемого значения> ,* *<идентификатор метода>*

*( <тип данных аргумента> <имя аргумента> ) { // тело метода }*

1. Для чего нужны геттеры и сеттеры.

Ответ: Если переменная имеет уровень доступа private(применяется при инкапсуляции), к ней невозможно обратиться извне класса, в котором она объявлена. Но все равно необходим способ обращения к private переменным из другого класса, иначе такие изолированные переменные не будут иметь смысла. Это достигается с помощью объявления специальных public методов. Методы, которые возвращают значение переменных, называются геттеры. Методы, которые изменяют значение свойств, называются сеттеры. С помощью геттеров и сеттеров можно обращаться к переменной, но не изменять ее.

Все нумеруемые заголовки выравниваются по ширине с отступом первой строки 1,25.

Список литературы

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

2. API Specification for the Java 7 SE. [официальный сайт] URL: [http://docs.oracle.com/javase/7/docs/api/](http://docs.oracle.com/javase/7/docs/api/%20)

3. The Java Tutorials. SE [электронный ресурс], URL: <http://docs.oracle.com/javase/tutorial/index.html>

4. 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.

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