|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»**  **Тема: «Строки в Java. Применение регулярных выражений для поиска заданных слов.»** | |
|  | |
| Выполнил студент группы ИКБО-09-18 | Налюшный А.Е. |
| Принял преподаватель | Баранова И.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись руководителя)* |

Москва 2018

Задание 1. Дан текст, который содержит информацию о продажах в магазине и о каждой продаже известно: название товара (одно слово строчными буквами) и сумма. Продаж может быть несколько и информация по одной продаже завершается символом ;(точка с запятой).

**Разработка первой задачи**

1. Условие задачи

Найти сумму всех продаж

2. Постановка задачи

Дано: массив строк

Ограничения на данные: массив не пуст.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x[][] – двумерный массив строк основанный на входной строке
* out – локальная целочисленная переменная
* В цикле проходим по всем элементам массива
* На каждой итерации цикла прибавляем к out значение x[номер итерации][1] в которой хранится число в строковом формате

4. Определение входных и выходных данных

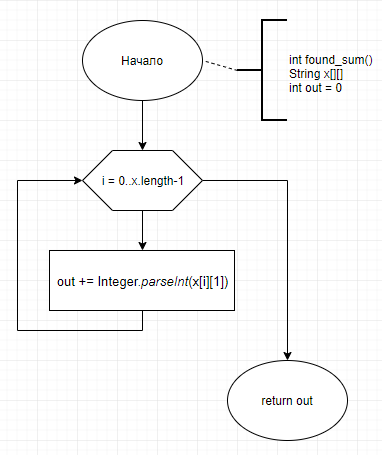
Входные данные – переменная класса.

Результат – целочисленное число.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| out | 0..231 - 1 | int | Переменная |
| x[][] | строка | String | Массив |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | out | out |
| 1. | p1 32;p2 8;p3 3 | 3 | 3 | пройден |
| 2. | CPU 33;RAM 999 | 1032 | 1032 | пройден |
| 3. | l 0;o 0;l 0 | 0 | 0 | пройден |
| 4. | h 1;e 2;l 3;l 4;0 5 | 15 | 15 | пройден |
| 5. | w 4;h 2;y 0 | 6 | 6 | пройден |
| 6. | h 6;e 6;l 6;l 6 | 24 | 24 | пройден |
| 7. | b 322;i 13;g 85 | 420 | 420 | пройден |
| 8. | p 77;l 88;a 33;y 20 | 218 | 218 | пройден |
| 9. | buy 444;me 543 | 987 | 987 | пройден |
| 10. | final 0;test 1 | 1 | 1 | пройден |

8. Исходный код программы

public int found\_sum()  
{  
 if (x == null)  
 return 0;  
 int out = 0;  
 for (int i = 0; i < x.length; i++)  
 out += Integer.*parseInt*(x[i][1]);  
 return out;  
}

**Разработка второй задачи**

1. Условие задачи

Отредактировать текст: название товара должно начинаться с прописной буквы.

2. Постановка задачи

Дано: массив строк

Ограничения на данные: массив не пуст.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x[][] – двумерный массив строк основанный на входной строке
* В цикле проходим по всем элементам массива
* На каждой итерации цикла заменяем первый символ названия продукта на её заглавный эквивалент

4. Определение входных и выходных данных

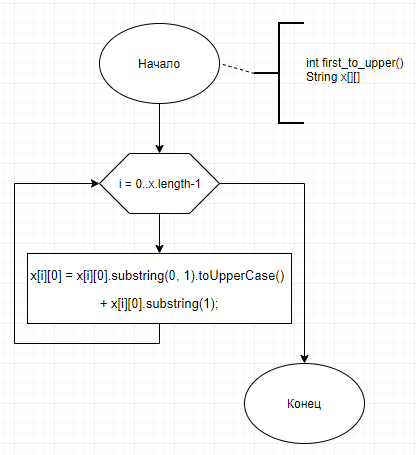
Входные данные – переменная класса.

Результат – изменение объекта.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x[][] | строка | String | Массив |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | x.toString() | x.toString() |
| 1. | product\_test 333 | Product\_test 333 | Product\_test 333 | пройден |
| 2. | two 0;products 1 | Two 0;Products 1 | Two 0;Products 1 | пройден |
| 3. | o 1;n 2;e 3 | O 1;N 2;E 3 | O 1;N 2;E 3 | пройден |
| 4. | wh 111;at 0 | Wh 111;At 0 | Wh 111;At 0 | пройден |
| 5. | to 4;test 9;there 9 | To 4;Test 9;There 9 | To 4;Test 9;There 9 | пройден |
| 6. | rero 1;rero 2;rero 3 | Rero 1;Rero 2;Rero 3 | Rero 1;Rero 2;Rero 3 | пройден |
| 7. | middle 13;line 31 | Middle 13;Line 31 | Middle 13;Line 31 | пройден |
| 8. | teremok 7;xd 0 | Teremok 7;Xd 0 | Teremok 7;Xd 0 | пройден |
| 9. | ko 33;no 99 | Ko 33;No 99 | Ko 33;No 99 | пройден |
| 10. | su 65;ba 60 | Su 65;Ba 60 | Su 65;Ba 60 | пройден |

8. Исходный код программы

public void first\_to\_upper()  
{  
 if (x == null)  
 return;  
 for (int i = 0; i < x.length; i++)  
 x[i][0] = x[i][0].substring(0, 1).toUpperCase() + x[i][0].substring(1);  
}

**Разработка третьей задачи**

1. Условие задачи

Сформировать рейтинг товаров, учтенных в продажах (по количеству продаж).

2. Постановка задачи

Дано: массив строк

Ограничения на данные: массив не пуст.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x[][] – двумерный массив строк основанный на входной строке
* y[][] – локальная копия x[][]
* out – локальная строковая переменная
* Сортируем y[][] по количеству продаж
* В цикле проходим по всем элементам копии
* На каждой итерации цикла добавляем название продукта в коней out

4. Определение входных и выходных данных

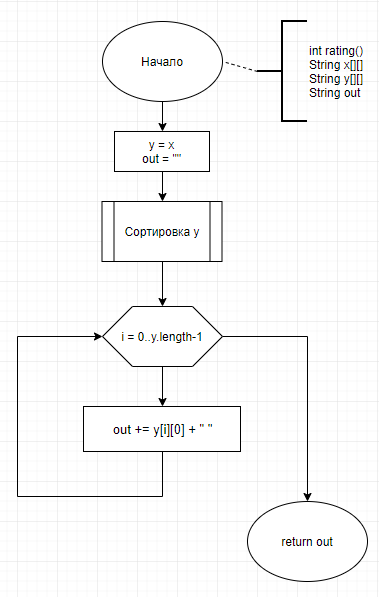
Входные данные – переменная класса.

Результат – строка.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x[][] | строка | String | Массив |
| y[][] | строка | String | Массив |
| out | строка | String | Локальная переменная |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | out | out |
| 1. | p1 11;p2 19;p3 0 | p2 p1 p3 | p2 p1 p3 | пройден |
| 2. | p1 0;p2 0;p3 0 | p1 p2 p3 | p1 p2 p3 | пройден |
| 3. | p1 9;p2 11;p3 13 | p3 p2 p1 | p3 p2 p1 | пройден |
| 4. | su 44;te 65;to 1 | te su to | te su to | пройден |
| 5. | q 11;w 22;e 1;r 5 | w q r e | w q r e | пройден |
| 6. | a 5;s 5;d 5;f 5 | a s d f | a s d f | пройден |
| 7. | te 88;re 234;bi 11 | re te bi | re te bi | пройден |
| 8. | lo 4;le 55;sptr 44 | le sprt lo | le sprt lo | пройден |
| 9. | int 66;byte 1;char 2 | int char byte | int char byte | пройден |
| 10. | n3 5;n2 10;n1 15 | n1 n2 n3 | n1 n2 n3 | пройден |

8. Исходный код программы

public String rating()  
{  
 if (x == null)  
 return null;  
 String out = "";  
 String y[][] = x;  
 Arrays.*sort*(y, new Comparator<String[]>() {  
 @Override  
 public int compare(final String[] entry1, final String[] entry2) {  
  
 Integer itemIdOne = Integer.*parseInt*(entry1[1]);  
 Integer itemIdTwo = Integer.*parseInt*(entry2[1]);  
 return itemIdTwo.compareTo(itemIdOne);  
 }  
 });  
 for (int i = 0; i < y.length; i++)  
 out += y[i][0] + " ";  
 return out;  
}

**Разработка четвертой задачи**

1. Условие задачи

Показать информацию по заданной по номеру продаже

2. Постановка задачи

Дано: массив строк, число

Ограничения на данные: массив не пуст, число меньше размера массива.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x[][] – двумерный массив строк основанный на входной строке
* y[] – локальный строковый массив
* n – целочисленная переменная
* y[] приравниваем к результату функции rating(), использовав на результат split(“ ”)
* возвращаем y[n-1]

4. Определение входных и выходных данных

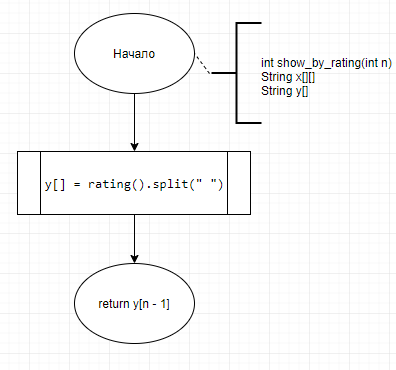
Входные данные – переменная класса, число.

Результат – строка.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x[][] | строка | String | Массив |
| y[] | строка | String | Массив |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | n | out | out |
| 1. | p1 33;p2 34;p3 35 | 1 | p3 | p3 | пройден |
| 2. | s 5;a 5;m 5;e 5 | 3 | m | m | пройден |
| 3. | z 0;e 0;r 0;o 0 | 2 | e | e | пройден |
| 4. | m 1;o 3;r 5;e 7 | 2 | r | r | пройден |
| 5. | l 14;e 12;s 10;s 8 | 3 | s | s | пройден |
| 6. | why 55;bcs 33 | 2 | bcs | bcs | пройден |
| 7. | fo 44;ur 3;mo 0;re 22 | 2 | re | re | пройден |
| 8. | three 9;now 81 | 1 | now | now | пройден |
| 9. | on 55;ly 19;two 88 | 3 | ly | ly | пройден |
| 10. | la 33;su 44;to 1 | 1 | su | su | пройден |

8. Исходный код программы

public String show\_by\_rating(int n)  
{  
 if (x == null || n > x.length || n <= 0)  
 return null;  
 String y[] = rating().split(" ");  
 return y[n-1];  
}

Задание 2. Дан текст, который содержит информацию о продажах в магазине и о каждой продаже известно: название товара (одно слово строчными буквами) и сумма. Продаж может быть несколько и информация по одной продаже завершается символом ;(точка с запятой).

**Разработка первой задачи**

1. Условие задачи

Удалить из текста сведения о продажах с суммой равной 0.

2. Постановка задачи

Дано: переменная типа StringBuffer

Ограничения на данные: переменная типа StringBuffer не пуста.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x – переменная типа StringBuffer
* y[] – локальный массив
* replacing – локальная строка
* инициализируем y[] = x.toString().split(“;”)
* В цикле проходим по всем элементам массива y[]
* На каждой итерации цикла приравниваем replacing к элементу массива если на конце стоит 0, а перед ним пробел
* Удаляем найденную продажу

4. Определение входных и выходных данных

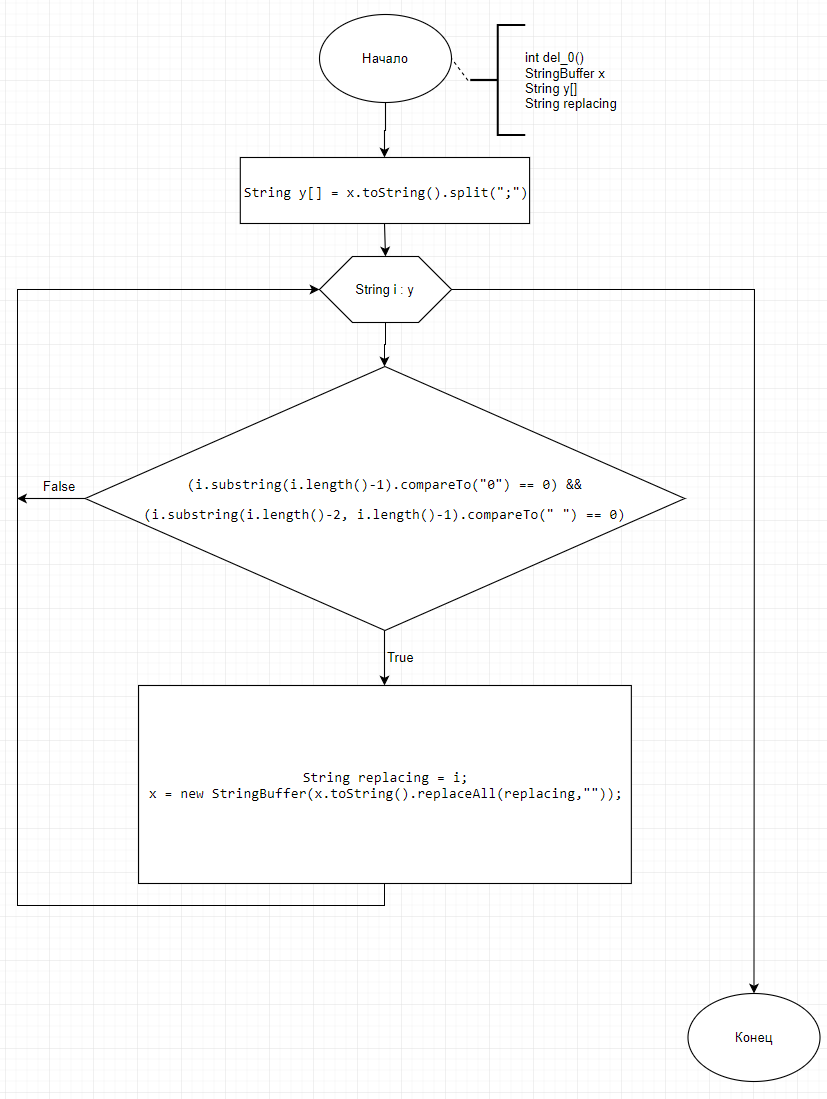
Входные данные – переменная класса.

Результат – измененная переменная класса.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x | строка | StringBuffer | строка |
| y[] | строка | String | Массив |
| replacing | строка | String | строка |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | x.toString() | x.toString() |
| 1. | another 12;test 0 | another 12 | another 12 | пройден |
| 2. | without 2;zero 9 | without 2;zero 9 | without 2;zero 9 | пройден |
| 3. | every 0;zero 0 |  |  | пройден |
| 4. | l 1;e 0;o 4;h 8 | l 1;o 4;h 8 | l 1;o 4;h 8 | пройден |
| 5. | p1 33;p2 0;p3 12 | p1 33;p3 12 | p1 33;p3 12 | пройден |
| 6. | ff 66;gg 77;wp 88 | ff 66;gg 77;wp 88 | ff 66;gg 77;wp 88 | пройден |
| 7. | onlyone 0 |  |  | пройден |
| 8. | onlyone\_2 33 | onlyone\_2 33 | onlyone\_2 33 | пройден |
| 9. | need 3;two 0;more 1 | need 3;more 1 | need 3;more 1 | пройден |
| 10. | last 0;bb 11 | bb 11 | bb 11 | пройден |

8. Исходный код программы

public void del\_0(){  
 if (x.toString().compareTo("") == 0)  
 return;  
 String y[] = x.toString().split(";");  
 for (String i : y){  
 if ((i.substring(i.length()-1).compareTo("0") == 0) && (i.substring(i.length()-2, i.length()-1).compareTo(" ") == 0)){  
 String replacing = i;  
 x = new StringBuffer(x.toString().replaceAll(replacing,""));  
 }  
 x = new StringBuffer(x.toString().replaceAll(";;",";"));  
 if (!x.toString().equals("")) {  
 if (x.substring(x.length() - 1).compareTo(";") == 0)  
 x = new StringBuffer(x.toString().replaceAll(";", ""));  
 if (x.substring(0, 1).compareTo(";") == 0)  
 x = new StringBuffer(x.toString().replaceAll(";",""));  
 }  
 }  
}

**Разработка второй задачи**

1. Условие задачи

Найти информацию о продаже, имеющей наименьшую длину. Заменить эту продажу на продажу наибольшего размера.

2. Постановка задачи

Дано: переменная типа StringBuffer

Ограничения на данные: переменная типа StringBuffer не пуста.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x – переменная типа StringBuffer
* y[] – локальный массив
* shortest, longest – локальные строки
* shortest\_first, shortest\_last – локальный целочисленные переменные
* инициализируем y[] = x.toString().split(“;”)
* В цикле проходим по всем элементам массива y[]
* На каждой итерации цикла сравниваем элемент с shortest и longest ,находя самую длинную и короткую строку
* Приравниваем к shortest\_first и shortest\_last значения индексов первого и последнего символов самого короткого элемента
* Заменяем самую короткую строку на самую длинную методом replace

4. Определение входных и выходных данных

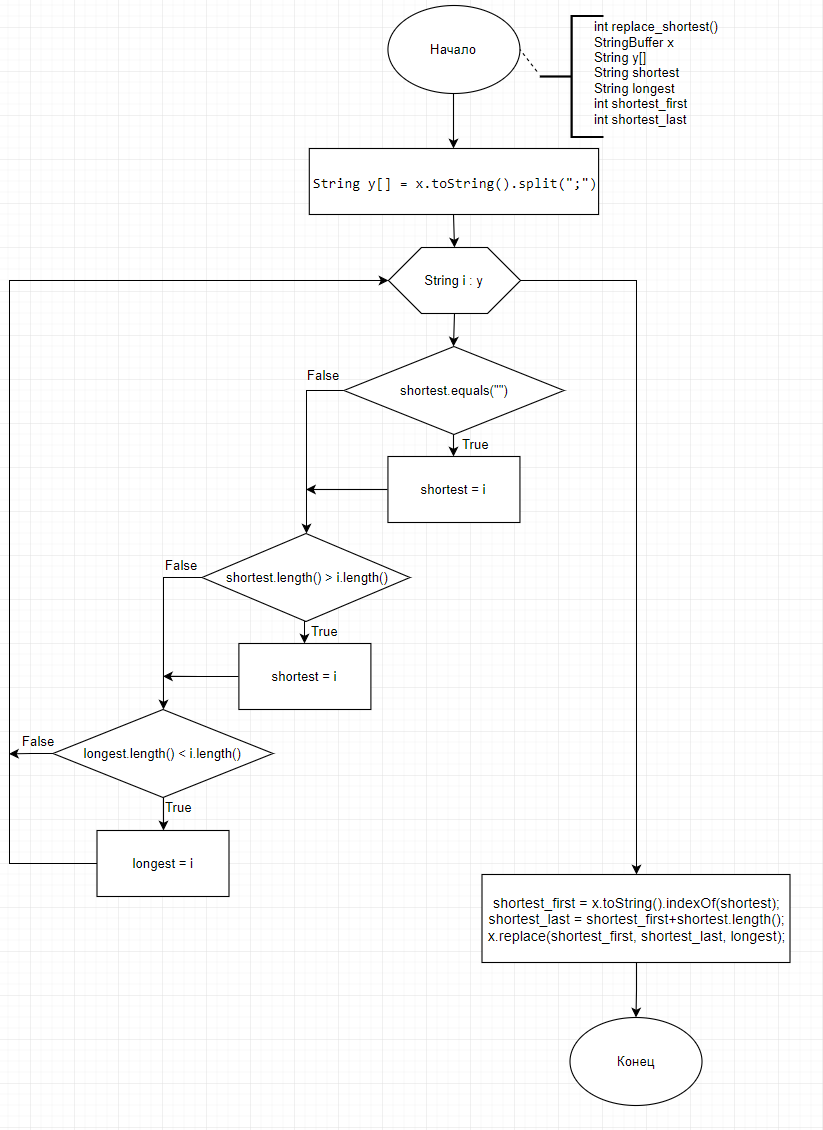
Входные данные – переменная класса.

Результат – изменение объекта.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x | строка | StringBuffer | строка |
| y[] | строка | String | Массив |
| shortest | строка | String | строка |
| longest | строка | String | строка |
| shortest\_first | 0..231 - 1 | int | Переменная |
| shortest\_last | 0..231 - 1 | int | Переменная |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | x.toString() | x.toString() |
| 1. | test 11;start 333 | start 333;start 333 | start 333;start 333 | пройден |
| 2. | same 1;size 0 | same 1;size 0 | same 1;size 0 | пройден |
| 3. | onlyone 4 | onlyone 4 | onlyone 4 | пройден |
| 4. | ff 3;f 1;fff 1 | ff 3;fff 1;fff 1 | ff 3;fff 1;fff 1 | пройден |
| 5. | a 1;b 11;c 111 | c 111;b 11;c 111 | c 111;b 11;c 111 | пройден |
| 6. | q 111;w 111;e 1 | q 111;w 111;q 111 | q 111;w 111;q 111 | пройден |
| 7. | p1 1;p2 2;p3 3;p4 4 | p1 1;p2 2;p3 3;p4 4 | p1 1;p2 2;p3 3;p4 4 | пройден |
| 8. | tr 3;nd 2;fst 0 | fst 0;nd 2;fst 0 | fst 0;nd 2;fst 0 | пройден |
| 9. | wrk 3;prfctl 2 | prfctl 2;prfctl 2 | prfctl 2;prfctl 2 | пройден |
| 10. | bye 1;bye 2 | bye 1;bye 2 | bye 1;bye 2 | пройден |

8. Исходный код программы

public void replace\_shortest(){  
 String y[] = x.toString().split(";");  
 String shortest = "", longest = "";  
 for (String i : y){  
 if (shortest.equals(""))  
 shortest = i;  
 if (shortest.length() > i.length())  
 shortest = i;  
 if (longest.length() < i.length())  
 longest = i;  
 }  
 if (shortest.length() == longest.length()){  
 System.*out*.println("Длина всех элементов одинакова");  
 return;  
 }  
 int shortest\_first, shortest\_last;  
 shortest\_first = x.toString().indexOf(shortest);  
 shortest\_last = shortest\_first+shortest.length();  
 x.replace(shortest\_first, shortest\_last, longest);  
 x = new StringBuffer(x.toString().replaceAll(" "," "));  
}

**Разработка третьей задачи**

1. Условие задачи

Добавить новую продажу в текст

2. Постановка задачи

Дано: переменная типа StringBuffer

Ограничения на данные: нет.

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* x – переменная типа StringBuffer
* text – входная строка
* добавляем text в конец x методом append

4. Определение входных и выходных данных

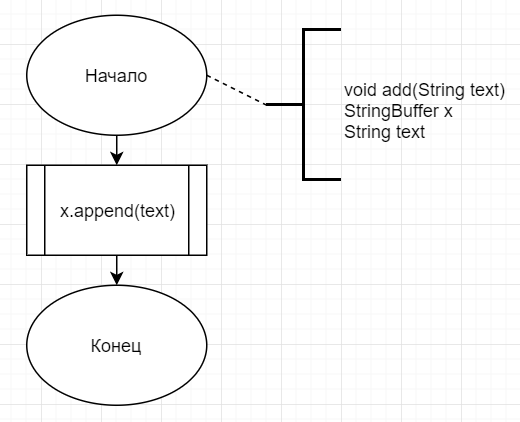
Входные данные – переменная класса.

Результат – измененная переменная класса.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x[][] | строка | String | Массив |
| text | строка | String | Входная переменная |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x.toString() | text | x.toString() | x.toString() |
| 1. | p1 3 | p2 0 | p1 3;p2 0 | p1 3;p2 0 | пройден |
| 2. |  | mo 4 | mo 4 | mo 4 | пройден |
| 3. | text 0 |  | text 0 | text 0 | пройден |
| 4. | “” | “” | “” | “” | пройден |
| 5. | a 66 | b 11 | a 66;b 11 | a 66;b 11 | пройден |
| 6. | q 1;w 2 | e 3 | q 1;w 2;e 3 | q 1;w 2;e 3 | пройден |
| 7. | wa 1 | wa 2 | wa 1;wa 2 | wa 1;wa 2 | пройден |
| 8. | cmng 2 | tgthr 1 | cmng 2;tgthr 1 | cmng 2;tgthr 1 | пройден |
| 9. | pre 7 | last 0 | pre 7;last 0 | pre 7;last 0 | пройден |
| 10. | m 1;o 3;r 5;e 7 | ! 9 | m 1;o 3;r 5;e 7;! 9 | m 1;o 3;r 5;e 7;! 9 | пройден |

8. Исходный код программы

public void add(String text){  
 x.append(text);  
}

Задание 3. Регулярные выражения

**Разработка первой задачи**

1. Условие задачи

Определите, что переданная строка является корректным временем вида '12:59', '23:41', '00:12', '00:00', '09:15'

2. Постановка задачи

Дано: строка

Ограничения на данные: нет

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* p – переменная типа Pattern
* m – переменная типа Matcher
* x – строка
* инициализируем p = Pattern.compile("^(2[0-3]|[0-1][0-9])\\:[0-5][0-9]$")
* инициализируем p = m = p.matcher(x)
* Возвращаем результат функции matches

4. Определение входных и выходных данных

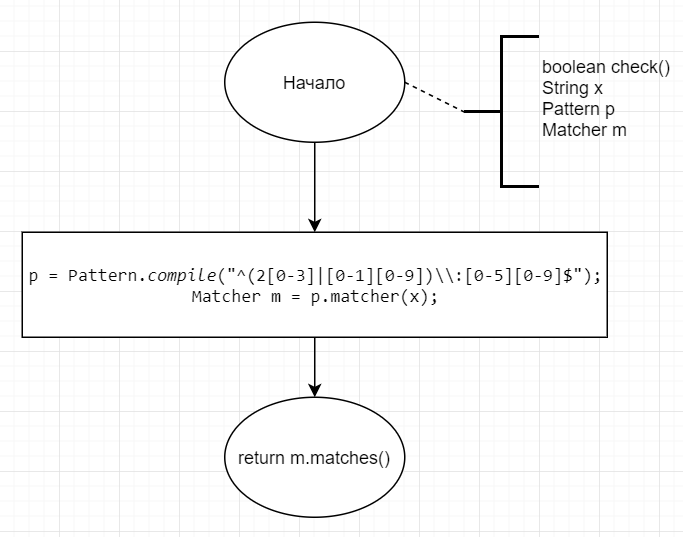
Входные данные – переменная класса.

Результат – true/false.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x | строка | String | строка |
| p | строка | Pattern | строка |
| m | Pattern | Matcher | строка |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x | m.matches() | m.matches() |
| 1. | 12:59 | true | true | пройден |
| 2. | 12:99 | false | false | пройден |
| 3. | 25:11 | false | false | пройден |
| 4. | 11;11 | false | false | пройден |
| 5. | 22:09 | true | true | пройден |
| 6. | 00:55 | true | true | пройден |
| 7. | 24:00 | false | false | пройден |
| 8. | 00:00 | true | true | пройден |
| 9. | hh:ss | false | false | пройден |
| 10. | 12:60 | false | false | пройден |

8. Исходный код программы

public boolean check(){  
 Pattern p = Pattern.*compile*("^(2[0-3]|[0-1][0-9])\\:[0-5][0-9]$");  
 Matcher m = p.matcher(x);  
 return m.matches();  
}

**Разработка второй задачи**

1. Условие задачи

Удалить из текста все значения времени из промежутка 00:00 до 02:00

2. Постановка задачи

Дано: строка

Ограничения на данные: нет

Ограничения на решаемую задачу: нет

3. Математическая модель задачи

* p – переменная типа Pattern
* m – переменная типа Matcher
* x – строка
* y[] – массив строк
* инициализируем p = Pattern.compile("^0[0-2]\\:[0-5][0-9]$")
* инициализируем m = p.matcher(x)
* инициализируем y[] = x.split(“ ”)
* В цикле проходим по всем элементам массива y[]
* На каждой итерации проверяем соответствует ли строка переменной m
* Если да то удаляем этот элемент

4. Определение входных и выходных данных

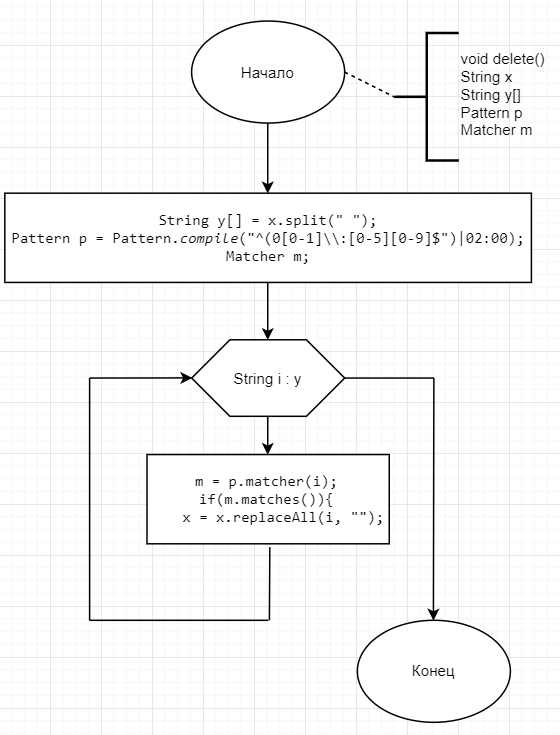
Входные данные – переменная класса.

Результат – изменение объекта.

5. Таблица имен объектов программы

|  |  |  |  |
| --- | --- | --- | --- |
| *Имя объекта* | *Диапазон допустимых значений* | *Тип данных* | *Семантика* |
| x | строка | String | строка |
| y[] | строка | String | Массив |
| p | строка | Pattern | строка |
| m | Pattern | Matcher | строка |

6. Разработка алгоритма программы



7. Разработка тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Исходные данные* | *Эталон результата* | *Результат программы* | *Отметка о правильном прохождении теста* |
| x | x | x |
| 1. | 11:11 12:12 | 11:11 12:12 | 11:11 12:12 | пройден |
| 2. | 01:11 21:22 | 21:22 | 21:22 | пройден |
| 3. | 11:11 00:00 | 11:11 | 11:11 | пройден |
| 4. | 01:01 00:59 | “” | “” | пройден |
| 5. | 21:00 01:53 03:02 | 1:00 03:02 | 1:00 03:02 | пройден |
| 6. | 22:11 19:53 08:22 | 22:11 19:53 08:22 | 22:11 19:53 08:22 | пройден |
| 7. | 02:00 14:55 | 14:55 | 14:55 | пройден |
| 8. | 11:21 01:59 | 11:21 | 11:21 | пройден |
| 9. | 02:00 02:02 | 02:02 | 02:02 | пройден |
| 10. | 21:21 02:59 | 21:21 02:59 | 21:21 02:59 | пройден |

8. Исходный код программы

public void delete(){  
 String y[] = x.split(" ");  
 Pattern p = Pattern.*compile*("^(0[0-1]\\:[0-5][0-9])|02:00$");  
 Matcher m;  
 for (String i : y){  
 m = p.matcher(i);  
 if(m.matches()){  
 x = x.replaceAll(i + " ", "");  
 }  
 } m = p.matcher(y[y.length - 1]);  
 if (m.matches())  
 x = x.replaceAll(y[y.length - 1], "");  
}

**Исходный код**

1. **Класс Task1**

package com.code;  
import java.util.Arrays;  
import java.util.Comparator;  
  
public class Task1 {  
  
 private String x[][];  
  
 public Task1()  
 {  
 x = null;  
 }  
  
 public Task1(String text) throws Exception  
 {  
 String y[] = text.split(";");  
 x = new String[y.length][2];  
 for (int i = 0; i < y.length; i++) {  
 String z[] = y[i].split(" ");  
 if (z.length > 2)  
 throw new Exception();  
 x[i][0] = z[0];  
 x[i][1] = z[1];  
 int test = Integer.*parseInt*(x[i][1]);  
 if (test < 0){  
 throw new Exception();  
 }  
 }  
 }  
  
 public String toString()  
 {  
 if (x == null)  
 return null;  
 String out = "";  
 for (int i = 0; i < x.length; i++)  
 out += x[i][0] + " " + x[i][1] + ";";  
 return out.substring(0, out.length()-1);  
 }  
  
 public int found\_sum()  
 {  
 if (x == null)  
 return 0;  
 int out = 0;  
 for (int i = 0; i < x.length; i++)  
 out += Integer.*parseInt*(x[i][1]);  
 return out;  
 }  
  
 public void first\_to\_upper()  
 {  
 if (x == null)  
 return;  
 for (int i = 0; i < x.length; i++)  
 x[i][0] = x[i][0].substring(0, 1).toUpperCase() + x[i][0].substring(1);  
 }  
  
 public String rating()  
 {  
 if (x == null)  
 return null;  
 String out = "";  
 String y[][] = x;  
 Arrays.*sort*(y, new Comparator<String[]>() {  
 @Override  
 public int compare(final String[] entry1, final String[] entry2) {  
  
 Integer itemIdOne = Integer.*parseInt*(entry1[1]);  
 Integer itemIdTwo = Integer.*parseInt*(entry2[1]);  
 return itemIdTwo.compareTo(itemIdOne);  
 }  
 });  
 for (int i = 0; i < y.length; i++)  
 out += y[i][0] + " ";  
 return out;  
 }  
  
 public String show\_by\_rating(int n)  
 {  
 if (x == null || n > x.length || n <= 0)  
 return null;  
 String y[] = rating().split(" ");  
 return y[n-1];  
 }  
}

1. **Класс Task2**

package com.code;  
  
public class Task2 {  
  
 private StringBuffer x;  
  
 public Task2(){  
 x = new StringBuffer("");  
 }  
  
 public Task2(String text){  
 x = new StringBuffer(text);  
 }  
  
 public String toString(){  
 return x.toString();  
 }  
  
 public void del\_0(){  
 if (x.toString().compareTo("") == 0)  
 return;  
 String y[] = x.toString().split(";");  
 for (String i : y){  
 if ((i.substring(i.length()-1).compareTo("0") == 0) && (i.substring(i.length()-2, i.length()-1).compareTo(" ") == 0)){  
 String replacing = i;  
 x = new StringBuffer(x.toString().replaceAll(replacing,""));  
 }  
 x = new StringBuffer(x.toString().replaceAll(";;",";"));  
 if (!x.toString().equals("")) {  
 if (x.substring(x.length() - 1).compareTo(";") == 0)  
 x = new StringBuffer(x.toString().replaceAll(";", ""));  
 if (x.substring(0, 1).compareTo(";") == 0)  
 x = new StringBuffer(x.toString().replaceAll(";",""));  
 }  
 }  
 }  
  
 public void replace\_shortest(){  
 String y[] = x.toString().split(";");  
 String shortest = "", longest = "";  
 for (String i : y){  
 if (shortest.equals(""))  
 shortest = i;  
 if (shortest.length() > i.length())  
 shortest = i;  
 if (longest.length() < i.length())  
 longest = i;  
 }  
 if (shortest.length() == longest.length()){  
 System.*out*.println("Длина всех элементов одинакова");  
 return;  
 }  
 int shortest\_first, shortest\_last;  
 shortest\_first = x.toString().indexOf(shortest);  
 shortest\_last = shortest\_first+shortest.length();  
 x.replace(shortest\_first, shortest\_last, longest);  
 x = new StringBuffer(x.toString().replaceAll(" "," "));  
 }  
  
 public void add(String text){  
 x.append(text);  
 }  
}

**3)Класс Task3**

package com.code;  
  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Task3 {  
 private String x;  
  
 Task3(){  
 x = null;  
 }  
  
 Task3(String text){  
 x = text;  
 }  
  
 public String toString(){  
 return x;  
 }  
  
 public boolean check(){  
 Pattern p = Pattern.*compile*("^(2[0-3]|[0-1][0-9])\\:[0-5][0-9]$");  
 Matcher m = p.matcher(x);  
 return m.matches();  
 }  
  
 public void delete(){  
 String y[] = x.split(" ");  
 Pattern p = Pattern.*compile*("^(0[0-1]\\:[0-5][0-9])|02:00$");  
 Matcher m;  
 for (String i : y){  
 m = p.matcher(i);  
 if(m.matches()){  
 x = x.replaceAll(i + " ", "");  
 }  
 } m = p.matcher(y[y.length - 1]);  
 if (m.matches())  
 x = x.replaceAll(y[y.length - 1], "");  
 }  
  
}

**4)Основная программа**

package com.code;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args)  
 {  
 String text\_for\_1\_2 = "tomato 11;meat 55;salad 6;fish 43;candy 50;eggplant 0;apple 49";  
 String text\_for\_3 = "22:33 00:01 21:40 12:40 02:00 11:00 01:55 16:47 00:00 21:21";  
 Scanner in = new Scanner(System.*in*);  
 boolean c = true;  
 int check;  
 System.*out*.println("1:Задание 1\n2:Задание 2\n3:Задание 3");  
 check = in.nextInt();  
 switch (check){  
 case 1:  
 System.*out*.println("1:Ввод\n2:Вывод\n3:Найти сумму всех продаж\n" +  
 "4:Отредактировать текст: название товара должно начинаться с прописной буквы\n" +  
 "5:Сформировать рейтинг товаров, учтенных в продажах\n" +  
 "6:Показать информацию по заданной по номеру продаже\n" + "7:Выход");  
 Task1 text1 = new Task1();  
 while (c){  
 System.*out*.println("Выберите действие\n");  
 check = in.nextInt();  
  
 switch (check){  
 case 1:  
 System.*out*.println("1:Готовый текст\n2:Самостоятельно");  
 check = in.nextInt();  
 switch (check){  
 case 1:  
 try {  
 text1 = new Task1(text\_for\_1\_2);  
 }catch (Exception e) {  
 text1 = new Task1();  
 System.*out*.println("Ошибка ввода");  
 }  
 break;  
 case 2:  
 System.*out*.println("Формат: <название> <сумма>;(...);<название> <сумма>\n");  
 String text\_for\_1\_user = in.nextLine();  
 text\_for\_1\_user = in.nextLine();  
 try {  
 text1 = new Task1(text\_for\_1\_user);  
 }catch (Exception e) {  
 text1 = new Task1();  
 System.*out*.println("Ошибка ввода");  
 }  
 break;  
 }  
 break;  
 case 2:  
 System.*out*.println(text1);  
 break;  
 case 3:  
 System.*out*.println(text1.found\_sum());  
 break;  
 case 4:  
 text1.first\_to\_upper();  
 System.*out*.println(text1);  
 break;  
 case 5:  
 System.*out*.println(text1.rating());  
 break;  
 case 6:  
 System.*out*.println("Под каким номером в рейтинге вывести элемент?");  
 check = in.nextInt();  
 System.*out*.println(text1.show\_by\_rating(check));  
 break;  
 case 7:  
 c = false;  
 break;  
 }  
 }  
 break;  
 case 2:  
 System.*out*.println("1:Ввод\n2:Вывод\n3:Удалить из текста сведения о продажах с суммой равной 0\n" +  
 "4:Найти информацию о продаже, имеющей наименьшую длину. " +  
 "Заменить эту продажу на продажу наибольшего размера\n" +  
 "5:Добавить новую продажу в текст\n6:Выход");  
 Task2 text2 = new Task2();  
 while(c){  
 System.*out*.println("Выберите действие\n");  
 check = in.nextInt();  
  
 switch (check){  
 case 1:  
 System.*out*.println("1:Готовый текст\n2:Самостоятельно");  
 check = in.nextInt();  
 switch (check){  
 case 1:  
 text2 = new Task2(text\_for\_1\_2);  
 break;  
 case 2:  
 System.*out*.println("Формат: <название> <сумма>;(...);<название> <сумма>\n");  
 String text\_for\_2\_user = in.nextLine();  
 text\_for\_2\_user = in.nextLine();  
 text2 = new Task2(text\_for\_2\_user);  
 break;  
 }  
 break;  
 case 2:  
 System.*out*.println(text2);  
 break;  
 case 3:  
 text2.del\_0();  
 break;  
 case 4:  
 text2.replace\_shortest();  
 break;  
 case 5:  
 System.*out*.println("Введите новую продажу");  
 System.*out*.println("Формат: <название> <сумма>\n");  
 String to\_end = in.nextLine();  
 to\_end = in.nextLine();  
 if (text2.toString().compareTo("") != 0 && to\_end.compareTo("") != 0)  
 text2.add(";");  
 text2.add(to\_end);  
 break;  
 case 6:  
 c = false;  
 break;  
 }  
 }  
 break;  
 case 3:  
 System.*out*.println("1:Определите, что переданная строка является корректным временем\n" +  
 "2:Удалите из текста все значения времени из промежутка 00:00 до 02:00\n3:Выход");  
 Task3 text3 = new Task3();  
 while (c){  
 System.*out*.println("Выберите действие\n");  
 check = in.nextInt();  
  
 switch (check) {  
 case 1:  
 System.*out*.println("Введите значение\n");  
 String entered\_time = in.nextLine();  
 entered\_time = in.nextLine();  
 text3 = new Task3(entered\_time);  
 System.*out*.println(text3.check());  
 break;  
 case 2:  
 System.*out*.println("1:Готовый текст\n2:Самостоятельно");  
 check = in.nextInt();  
 switch (check) {  
 case 1:  
 text3 = new Task3(text\_for\_3);  
 text3.delete();  
 System.*out*.println(text3);  
 break;  
 case 2:  
 System.*out*.println("Введите текст\n");  
 System.*out*.println("Формат: <Время1> <Время2> (...) <ВремяN>\n");  
 String entered\_text = in.nextLine();  
 entered\_text = in.nextLine();  
 text3 = new Task3(entered\_text);  
 text3.delete();  
 System.*out*.println(text3);  
 break;  
 }  
 break;  
 case 3:  
 c = false;  
 break;  
 }  
 }  
 break;  
 }  
  
 }  
}