Задачи

1. Транспонировать матрицу относительно главной диагонали.
2. Транспонировать матрицу относительно побочной диагонали
3. Задана матрица (двумерный массив) A размером N x M , состоящая из действительных элементов. Получить новую матрицу путем умножения всех элементов данной матрицы на ее наибольший элемент. Сформировать вектор из элементов главной диагонали и отсортировать его по возрастанию.
4. Дана матрица нулевых элементов. В любом месте матрицы ставится одно значение 1. Нужно посчитать за сколько ходов единица «захватит мир», если каждый ход область ее владений расширяется на соседние элементы.
5. В предыдущей задаче поменять условие: элементов ставится несколько и их значения указывают на количество расширений владений за 1 ход.

Строки

Строки тоже являются переменными ссылочного типа, а точнее — ссылками на объекты одного из нескольких строковых классов Java. Мы рассмотрим класс String.

Самый распространенный способ создать строку — это организовать ссылку типа String на строку-константу:
String s = "Это строка"

Можно просто сначала объявить переменную, а потом заставить ее ссылаться на строку-константу, другую строку или воспользоваться командой new, чтобы явным образом выделить память для строки:
String s1, s2, s3; // Объявление трех переменных, которые пока не связаны ни с какой строкой
s1 = "Да здравствует день танкиста"; // Переменная s1 теперь ссылается на область памяти, в которой хранится строка "Да здравствует день танкиста"
s2 = s1; // Теперь обе переменные s1 и s2 ссылаются на одно и то же место памяти
s3 = new String(); // s3 ссылается на место в памяти, где хранится пустая строка

Операции со строками

Объединение (сцепление) строк

Сцепление строк производится командой +. Ей соответствует оператор +=, который часто бывает очень удобен:

```
String S = "Привет";
String S1 = "мир"; S += ", " + S1 + "!"; // Теперь S ссылается на строку “Привет, мир!”
```

Длина строки

Определить длину строки можно методом length():

```
int x = S.length(); // Переменная x получит значение 12
```

Обратите внимание, String является классом, а length() - его методом, и поэтому указывается через точку после имени переменной. Аналогично записываются и другие методы класса String.

Операции со строками

Получение отдельных символов строки

Метод charAt(int i) возвращает символ строки с индексом i. Индекс первого символа строки — 0 (т.е. символы строки индексируются (нумеруются) аналогично элементам массива. Например:

```
char ch = S.charAt(2); // Переменная ch будет иметь значение 'и'
```

Метод toCharArray() преобразует строку в массив символов:

```
char[] ch = S.toCharArray(); // ch будет иметь представлять собой массив {'П','р','и','в','е','т',' ',',',' ','м','и','р','!'}
```

Замена отдельного символа

Метод replace(int old, int new) возвращает новую строку, в которой все вхождения символа old заменены на символ new.

```
String SS = S.replace('и', 'ы'); // SS = "Прывет, мыр!"
```

Операции со строками

Получение подстроки

Метод substring(int begin, int end) возвращает фрагмент исходной строки от символа с индексом begin до символа с индексом end-1 включительно. Если не указывать end, будет возвращен фрагмент исходной строки, начиная с символа с индексом begin и до конца:

String S2 = S.substring(4,7); // S2 = "ет,"

S2 = S.substring(4); // S2 = "ет, мир!"

Разбиение строки на подстроки

Метод split(String regExp) разбивает строку на фрагменты, используя в качестве разделителей символы, входящие в параметр regExp, и возвращает ссылку на массив, составленный из этих фрагментов. Сами разделители ни в одну подстроку не входят.

String parts[] = S.split(" "); // Разбили строку S на отдельные слова, используя пробел в качестве разделителя, в результате получили массив parts, где parts[0] = "Привет,", а parts[1] = "мир!"

String parts[] = S.split(" и"); // Разбили строку S на отдельные слова, используя в качестве разделителя пробел и букву и, в результате parts[0] = "Пр", parts[1] = "вет,", parts[2] = "м", parts[3] = "р!"

Операции со строками

Сравнение строк

Если сравнивать строки, используя логическую операцию ==, то ее результатом будет true только в том случае, если строковые переменные указывают (ссылаются) на один и тот же объект в памяти.

Если же необходимо проверить две строки на совпадение, следует использовать стандартный метод equals(Object obj). Он возвращает true, если две строки являются полностью идентичными вплоть до регистра букв, и false в противном случае. Его следует использовать следующим образом:

```
S1.equals(S2); // Вернет true, если строки S1 и S2 идентичны
S2.equals(S1); // Абсолютно то же самое
boolean b = S.equals("Привет, мир!"); // b = true
```

Метод equalsIgnoreCase(Object obj) работает аналогично, но строки, записанные в разных регистрах, считает совпадающими.

Операции со строками

Поиск подстроки

Метод indexOf(int ch) возвращает индекс первого вхождения символа ch в исходную строку. Если задействовать этот метод в форме indexOf(int ch, int i), то есть указать два параметра при вызове, то поиск вхождения начнется с символа с индексом i. Если такого символа в строке нет, результатом будет -1.

```
int pos = S.indexOf('в'); // pos = 3
pos = "Корова".indexOf('о', 2); // pos = 3
pos = "Корова".indexOf('К', 2); // pos = -1, поиск ведется с учетом регистра
```

Последнее вхождение символа можно найти с помощью метода lastIndexOf(int ch) или lastIndexOf(int ch, int i), который работает аналогично, но просматривает строку с конца.

У всех перечисленных методов есть одноименные варианты, которые принимают в качестве параметра строку вместо символа и проверяют, содержится ли эта строка в исходной строке.

```
pos = "Корова".indexOf("ор"); // pos = 1
```

Изменение регистра символов в строке

Метод toLowerCase() возвращает новую строку, в которой все буквы сделаны строчными. Метод toUpperCase() возвращает новую строку, в которой все буквы сделаны прописными.

```
S = S.toUpperCase(); // S = "ПРИВЕТ, МИР!"
```

Задачи

1. С клавиатуры задается строка. Заменить в заданной строке все пробелы знаками подчеркивания.
2. С клавиатуры задается строка. Определить длину строки в символах и в словах (разделителем между словами считать знак пробела). Вывести заданную строку в обратном порядке по символам и по словам. + учесть множественные пробелы
3. Дан массив текстовых значений. Найти самый длинный элемент массива. Создать предложение из входящих в массив строк, самый длинный элемент массива разместить в начале предложения.
4. Поменять все буквы на противоположный регистр.
5. Поставить слова в предложении в алфавитном порядке по первой букве (с учетом множественности пробела).
6. Поменять начало слов на буквы противоположного регистра.
7. В задании 5 учитывать все буквы слов.
8. Сделать транслитератор (если вводится текст на английском, тогда транслитерировать его на русский и наоборот)