Университет ИТМО

Кафедра вычислительной техники

**Отчёт по лабораторной работе №1  
по предмету «Программирование интернет-приложений»**

Выполнил:  
Студент гр. P3201  
Петров Д. С.

Принял:  
Гаврилов А. В.

**Номер варианта**

Отсутствует

**Цель работы**

Целью данной работы является практическое освоение принципов ООП на примере языка Java, ознакомление с базовыми утилитами входящих в состав JDK таких, как java, javac, javadoc и jar, изучение особенностей языка Java, знакомство с понятиями переменного окружения, classpath.

**Задание**

На языке Java написать простейшую консольную программу, выводящую в поток System.out список переданных ей аргументов командной строки. Дополнить эту программу использованием любого из классов стандартной библиотеки или любой синтаксической конструкции языка Java.

Обязательно использовать:

1. Аргументы командной строки (массив args)
2. Методы класса Math
3. Циклы (конструкции for, while)
4. Обработку исключений (класс Exception)

**Исходный код получившейся программы**

import java.lang.\*;

import java.lang.Math.\*;

public class Main

{

public static void main(String[] args)

{

try

{

if(args.length != 2) throw new Exception("Not good");

String s1 = args[0];

String s2 = args[1];

char[] chArray1 = s1.toCharArray();

char[] chArray2 = s2.toCharArray();

int n1, n2;

n1 = chArray1.length;

n2 = chArray2.length;

//Динамическое программирование:

int[][] matrix;

matrix = new int[n1 + 1][n2 + 1]; //Массив инициализируется нулями

for(int i = 1; i <= n1; i++)

{

for(int j = 1; j <= n2; j++)

{

if(chArray1[i - 1] == chArray2[j - 1])

{

matrix[i][j] = matrix[i - 1][j - 1] + 1;

}

else

{

matrix[i][j] = matrix[i - 1][j] > matrix[i][j - 1] ? matrix[i - 1][j] : matrix[i][j - 1];

}

}

}

//Восстановление ответа:

int leng = matrix[n1][n2];

char[] ans = new char[leng];

int i = n1, j = n2;

int k = leng - 1;

while(i > 0 && j > 0)

{

while (i > 0 && matrix[i][j] == matrix[i - 1][j]) i--;

while (j > 0 && matrix[i][j] == matrix[i][j - 1]) j--;

if(i > 0 && j > 0) //&& matrix[i][j] == matrix[i - 1][j - 1] + 1)

{

ans[k--] = chArray1[i - 1];

i--;

j--;

}

}

String ans\_string = new String(ans); //Собираем строку из массива char через конструкто строки

//Распознана строка:

System.out.println("length of number = " + ans\_string.length());

System.out.println(ans\_string); //if empty - all ok

try

{

int number = new Integer(ans\_string);

System.out.println((int)Math.pow((double)number, 2));

}

catch(NumberFormatException ex\_number)

{

System.out.println("Sorry, but your argument isn't numeric!");

}

}

catch(Exception e)

{

System.out.println("Sorry, but your argument is not correct!");

}

}

}

**Описание базовых утилит в составе JDK**

Переменная окружения CLASSPATH определяет дополнительные пути поиска классов. Путь по умолчанию указывает на jar-архивы с классами Java API, входящими в состав JDK, которые находятся в каталогах lib и jre/lib. В переменной CLASSPATH через символ : (двоеточие) перечисляются директории, zip- и jar-архивы, содержащие классы, необходимые для выполнения программ.

**Java. Интерпретатор байт-кода.**

*java [параметры] имя\_класса [аргументы]*

Программа, которую необходимо выполнить — это байт-код, находящийся в классе с именем *имя\_класса*.

Любые аргументы, следующие за именем класса в командной строке Java, помещаются в массив args[] и передаются методу main() при запуске Java. Интерпретатор автоматически загружает все дополнительные классы, необходимые для выполнения программы.

Параметры:

* *- classpath путь* — переопределяет путь поиска классов, заданный переменной classpath
* *- cs* – указывает на необходимость сравнения времени компиляции файла касса и времени последней модификации исходного текста программмы и при необходимости перекомпиляции программы.
* *-D имя = значение* — присваивает системному свойству с заданным именем указанное значение.
* -*noverify* — отключает режим проверки байт-кода
* *-verify* — включает режим проверки байт-кода для всех классов(по умолчанию проверяются только классы загруженные по сети).
* *-v(-verbose)* — выводит на экран сообщение когда Java загружает класс.

**Javac. Компилятор Java.**

*javac [параметры] файлы*

Программа javac компилирует исходные тексты Java в байт-код Java.

Компилятор javac может передать в командной строке любое кол-во файлов исходных текстов java, чьи имена должны иметь расширение java

Когда файл исходного текста ссылается на класс, не определенный в командной строке какого-либо другого файла исходного текста, javac ищет этот класс, используя параметр -classpath

Параметры:

* *-classpath* путь — определяет пусть, используемый javac для поиска классов, на которые имеются ссылки в исходных текста.
* *-d каталог* — определеяет каталог, в котором должны храниться файлы классов.
* *-depend* —javac о необходимости перекомпилировать любой встретившийся ему устаревший файл.
* *-nowarn* — отключает вывод предупреждений. Сообщения об ошибках выводятся.
* *-vebrose* — дает компилятору указание выдавать сообщение о том, что он делает.

**Javadoc. Генератор документаци.**

*javadoc [параметры] имя пакета  
javadoc [параметры] имена файлов*

Программа javadoc создает документацию в формате HTML для указанных пакетов или отдельных файлов исходных текстов Java, заданных в командной строке.

Файлы документации, создаваемые javadoc, описывают как иерархию наследования классов, так и индекс и каждый член класса, объявленный как public или prоtected. Созданный файл содержит комментарии, которые связаны с классами, методами, конструкторами и переменными. Комментарии для документаций начинаются символами /\*\* и оканчиваются символами \*/. Они могут включать в себя HTML-теги вместе с их значениями.

Параметры:

* *-author путь* — выводит информацию о авторе, заданной с помощью тега @author. Эта информация по умолчанию не выводится.
* *-classpath путь* — определяет путь, Который javadoc использует для поиска, как файлов классов, так и фалов с исходными текстами для указанного пакета.
* *-d каталог* — каталог, в который javadoc должен записывать создаваемые им в HTML-файлы.
* *-decoding название\_кодировки*Задает кодировку символов, которая будет использоваться в документации, создаваемой с помощью javadoc.
* *-encoding название\_кодировки*Задает кодировку символов, которая будет использоваться при чтении файлов с исходными текстами и комментариями, содержащимися в них
* *-vebrose* — дает компилятору указание выдавать сообщение о том, что он делает.

**Jar. Утилита создающая архивные файлы.**

*jar c|t|x[f][m][v] [jar-файл] [файл описания} [файлы}*

Программа jar используется для создания архивных файлов Java (JAR) и работы с ними. JAR-файл представляет собой сжатый ZIP-файл с дополнительным файлом описания.

Параметры командной строки jar задаются в виде блока ключей – записанных слитно букв, которые передаются в виде одного аргумента, а не через отдельные аргументы командной строки. Первая ключ такого аргумента задает необходимое действие, которое должна выполнить программа jar. Остальные ключи в этом аргументе являются необязательными. Различные дополнительные аргументы зависят от того, какие ключи были заданы.

Параметры:

Первым аргументом командной строки jar является набор ключей, задающих операцию, которая должна быть выполнена. Первый ключ определяет основную операцию и является обязательным.

* *с* – создать новый JAR-архив.
* *t* – вывести список файлов, содержащихся в JAR-архиве
* *x* – извлечь содержимое JAR-архива

За ключом, определяющим выполняемое действие, могут следовать необязательные ключи:

* *f* – указывает на то, что имя JAR-файла, над которым нужно произвести указанные действия задается в командной строке.
* *m* – используется только вместе с ключом ‘с’ и указывает на то, что утилита jar должна использовать файл описания, указанный в командной строке в качестве основы для создания описания, включаемого в JAR-файл.
* *v* – при использовании в сочетании с ключом ‘с’ выводится имя каждого добавляемого в архив файла со статистикой его сжатия.

**Вывод**

В ходе выполнения лабораторной работы было установлено, что массивы в java «знают о своей длине», а также их длину можно устанавливать в ходе выполнения программы, в отличие от массивов, предусмотренных языком Си.

На языке java просто и понятно реализуются простые алгоритмы, такие как «наибольшая общая подпоследовательность» и другие.

В ходе выполнения лабораторной работы мы ознакомились с синтаксическими особенностями языка Java, а также узнали об основных средствах JDK и их основных параметрах.