Лабораторная работа №2

РАЗРАБОТКА КОНСОЛЬНЫХ JAVA-ПРИЛОЖЕНИЙ

Цель работы:

В данной лабораторной работе разрабатывается консольное приложение для реализации простейшего приложения с использованием массивов, строк и файлов.

Указания к работе:

Консольное приложение Java представляет собой откомпилированный класс, содержащий точку входа.

Рассмотрим простой пример:

```
public class First {
    public static void main(String[] args) {
        System.out.println ("Первая программа на Java!");
        }
    }
```

Здесь класс **First** используется только для того, чтобы определить метод main(), который и является точкой входа и с которого начинается выполнения программы интерпретатором Java. Метод main() содержит аргументы-параметры командной строки String[] args в виде массива строк и является открытым (public) членом класса. Это означает, что метод main() виден и доступен любому классу. Ключевое слово *static* объявляет методы и переменные класса, используемые для работы с классом в целом, а не только с объектом класса. Символы верхнего и нижнего регистров в Java **различаются**, чем Java напоминает языки C/C++ и PHP.

Вывод строки «Первая программа на Java!» в примере осуществляет метод println() (ln – переход к новой строке после вывода) свойства *out* класса System, который доступен в программе автоматически вместе с пакетом java.lang. Приведенную программу необходимо поместить в файл, имя которого совпадает с именем класса и с расширением java. Простейший способ компиляции написанной программы – вызов строчного компилятора:

javac First.java

При успешной компиляции создается файл First.class. Этот файл можно запустить на исполнение из командной строки с помощью интерпретатора Java следующим образом:

java First

Для разработки программы возможно использование и специальных средств разработчика.

- **NetBeans IDE** – свободная интегрированная среда разработки для всех платформ Java – Java ME, Java SE и Java EE. Пропагандируется Sun Microsystems, разработчиком Java, как базовое средство для разработки ПО на языке Java.
- **Eclipse IDE** – свободная интегрированная среда разработки для Java SE, Java EE и Java ME. Пропагандируется IBM, одним из важнейших разработчиков корпоративного ПО, как базовое средство для разработки ПО на языке Java.
- **IntelliJ IDEA** – среда разработки для платформ Java SE, Java EE и Java ME. Разработчик – компания JetBrains. Распространяется в двух версиях: свободной бесплатной (Community Edition) и коммерческой проприетарной (Ultimate Edition).
- **JDeveloper** – среда разработки для платформ Java SE, Java EE и Java ME. Разработчик — компания Oracle.
- **JBuilder –** профессиональная интегрированная среда разработки (IDE) на языке Java, основанная на программной среде с открытыми исходными кодами Eclipse.
- **BlueJ** – Среда разработки программного обеспечения на языке Java, созданная в основном для использования в обучении, но также подходящая для разработки небольших программ.

Ниже рассмотрены основные классы, используемые при выполнении лабораторной работы, рассмотрен пример решения одного из заданий.

Класс **java.io.File**

Для работы с файлами в приложениях Java могут быть использованы классы из пакета java.io, одним из которых является класс File.

Класс **File** служит для хранения и обработки в качестве объектов каталогов и имен файлов. Этот класс не описывает способы работы с содержимым файла, но позволяет манипулировать такими свойствами файла, как права доступа, дата и время создания, путь в иерархии каталогов, создание, удаление, изменение имени файла и каталога и т.д.

Основные *методы класса File* и способы их применения рассмотрены в следующем примере:

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws IOException
    /*отказ от обработки исключения в main()*/{
```

```
        //с объектом типа File ассоциируется файл на диске. Способы создания объекта
(убрать "//" по очереди)
        //File fp = new File( "com\\learn\\FileTest.java" );
        //File fp = new File("\\com\\learn", "FileTest.java");
        //File fp = new File("d:\\temp\\demo.txt");
        File fp = new File("demo.txt");

        if(fp.isFile()){//если объект дисковый файл
            System.out.println("Имя файла:\t" + fp.getName());
            System.out.println("Путь к файлу:\t" + fp.getPath());
            System.out.println("Абсолютный путь:\t" + fp.getAbsolutePath());
            System.out.println("Канонический путь:\t" + fp.getCanonicalPath());
            System.out.println("Размер файла:\t" + fp.length());
            System.out.println("Последняя модификация файла:\t" + fp.lastModified());
            System.out.println("Файл доступен для чтения:\t" + fp.canRead());
            System.out.println("Файл доступен для записи:\t" + fp.canWrite());
            System.out.println("Файл удален:\t" + fp.delete());
            }

        if(fp.createNewFile()){
            System.out.println("Файл " + fp.getName() + " создан");
            }

        if(fp.exists()){
            System.out.println("temp файл "+ fp.getName() + " существует");
            }

        else
            System.out.println("temp файл " + fp.getName() + " не существует");
        //в объект типа File помещается каталог\директория
        File dir = new File( "com\\learn" );

        if (dir.isDirectory())/*если объект объявлен как каталог на диске*/
            System.out.println("Директория!");

        if(dir.exists()){//если каталог существует
            System.out.println( "Dir " + dir.getName() + " существует " );
            File [] files = dir.listFiles();
            System.out.println("");

            for(int i=0; i < files.length; i++){
                Date date = new Date(files[i].lastModified());
                System.out.println(files[i].getPath() + " \t| " + files[i].length() +
"\t| " + date.toString());
                //toLocaleString());//toGMTString())
                }
            }
        }
    }
```

У каталога (директории) как объекта класса File есть дополнительное свойство – просмотр списка имен файлов с помощью методов list(), listFiles(), listRoots().

Класс **System**

Класс System содержит набор полезных статических методов и полей системного уровня. Экземпляр этого класса не может быть создан или получен.

Наиболее широко используемой возможностью, предоставляемой System, является стандартный вывод, доступный через переменную System.out. Стандартный вывод можно перенаправить в другой поток (файл, массив байт и т.д., главное, чтобы это был объект PrintStream, смотри документацию JSDK: http://docs.oracle.com/javase/6/docs/api/):

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintStream;

public class Second {
    public static void main(String[] args) {
        System.out.println("Study Java");

    try {
        PrintStream print = new PrintStream(new FileOutputStream("d:\\file2.txt"));
        //место на диске, куда сохраняеться файл
        System.setOut(print);
        System.out.println("Study well");
        }

    catch(FileNotFoundException e) {
        e.printStackTrace();
        }
    }
}
```

При запуске этого кода на экран будет выведено только:

```
Study Java.
```

И в файл "d:\file2.txt" будет записано:

```
Study well.
```

Аналогично могут быть перенаправлен стандартный ввод System.in – вызовом System.setIn(InputStream) и поток вывода сообщений об ошибках System.err – вызовом System.setErr (по умолчанию все потоки – in, out, err – работают с консолью приложения).

Класс **String**

Класс String содержит основные методы для работы со строками:

- concat(String s) или + – слияние строк;
- equals(Object ob), equalsIgnoreCase(String s) – сравнение строк с учетом и без учета регистра;
- compareTo(String s), compareToIgnoreCase (String s) – лексикографическое сравнение строк с учетом и без учета регистра;
- contentEquals(StringBuffer ob) – сравнение строки и содержимого объекта типа StringBuffer;
- charAt(int n)– извлечение из строки символа с указанным номером (нумерация начинается с нуля);
- substring(int n, int m)- извлечение из строки подстроки длины m-n, начиная с позиции n;
- length() – определение длины строки;
- valueOf(объект) – преобразование примитивного объекта к строке;

- toUpperCase() / toLowerCase() – преобразование всех символов вызывающей строки в верхний/нижний регистр;
- replace(char c1, char c2) – замена в строке всех вхождений первого символа вторым символом;
- getBytes(параметры), getChars(параметры) – извлечение символов строки в виде массива байт или символов.

В следующем примере массив символов и целое число преобразуются в объекты типа String с использованием методов этого класса.

```
public class DemoString {
    public static void main(String[] args) {
        char s[] = { 'J', 'a', 'v', 'a' };
        int i = 2;

        // комментарий содержит результат выполнения кода
        String str = new String(s); // str ="Java"
        i = str.length(); // i=4
        String num = String.valueOf(2); // num="2"
        str = str.toUpperCase(); // str="Java"
        num = str.concat(num); // num="Java2"
        str = str + "C";// str="JavaC";
        char ch = str.charAt(2); // ch='V'
        i = str.lastIndexOf( 'A' ); // i=3 (-1 если отсутствует )
        num = num.replace('2', 'H'); // num="JavaH"
        i = num.compareTo(str); // i=5 ( между символами 'H' и ' C ')
        str.substring(0, 3).toLowerCase(); // java
        }
    }
```

Пример консольного java-приложения:

Задание: Ввести n строк с консоли. Вывести на консоль строки и их длины, упорядоченные по возрастанию.

Решение:

```
import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        int n = 0;
        while ( true ) // ввод числа строк
            {
            System.out.println("Введите число строк");
            Scanner sc1 = new Scanner(System. in );
            try {
                n = sc1.nextInt();
                break;
                }
            catch(InputMismatchException fg){
                // если введенное значение не является числом
                System.out.print("Вы ввели не число. " );
                }
            }
        // создание массива строк
        String[] str = new String[n];
        Scanner sc2 = new Scanner(System.in);
```

```
        for (int i = 0; i < n; i++){
            System. out.println( " Введите строку №" + (i+1));
            str[i] = sc2.nextLine();
            }
        //сортировка массива строк по длине
        for ( int i = 0; i < str. length -1; i++){

            for (int j = i+1; j < str.length; j++){

                if (str[i].length()<str[j].length()){
                    String temp = str[i];
                    str[i] = str[j];
                    str[j] = temp;
                    }
                }
            }

        int maxlength = str[0].length();
        System.out.println("Строки в порядке убывания длины:");
        for ( int i = 0; i < str. length ; i++){
            System.out.print(str[i]);
            for (int j = 0; j < maxlength - str[i].length(); j++)
                System.out.print(" ");
                System.out.println( " её длина = " + str[i].length());
                }
        }
    }
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Ввести n строк с консоли, найти самую короткую строку. Вывести эту строку и ее длину.

2. Ввести n строк с консоли. Упорядочить и вывести строки в порядке возрастания их длин, а также (второй приоритет) значений этих их длин.

3. Ввести n строк с консоли. Вывести на консоль те строки, длина которых меньше средней, также их длины.

4. В каждом слове текста k-ю букву заменить заданным символом. Если k больше длины слова, корректировку не выполнять.

5. В русском тексте каждую букву заменить ее номером в алфавите. В одной строке печатать текст с двумя пробелами между буквами, в следующей строке внизу под каждой буквой печатать ее номер.

6. Из небольшого текста удалить все символы, кроме пробелов, не являющиеся буквами. Между последовательностями подряд идущих букв оставить хотя бы один пробел.

7. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.

8. В тексте найти все пары слов, из которых одно является обращением другого.

9. Найти и напечатать, сколько раз повторяется в тексте каждое слово.

10. Найти, каких букв, гласных или согласных, больше в каждом предложении текста.

11. Выбрать три разные точки заданного на плоскости множества точек, составляющие треугольник наибольшего периметра.

12. Найти такую точку заданного на плоскости множества точек, сумма расстояний от которой до остальных минимальна.

13. Выпуклый многоугольник задан на плоскости перечислением координат вершин в порядке обхода его границы. Определить площадь многоугольника.