**Соритровки**

**Пузырьковая сортировка**

будем обходить весь массива, запихивая найденные максимумы в конец.

public static void bubbleSort(int[] arr){

for(int i = arr.length-1 ; i > 0 ; i--){

for(int j = 0 ; j < i ; j++){

if( arr[j] > arr[j+1] ){

int tmp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = tmp;

}

}

}

}

**Шейкерная сортировка**

Шейкерная сортировка работает немного быстрее чем пузырьковая, поскольку по массиву в нужных направлениях попеременно мигрируют и максимумы и минимумы.

public static void shakerSort(int array[]) {

int buff;

int left=0;

int right=array.length-1;

do {

for (int i=left; i<right;i++) {

if (array[i]>array[i+1]) {

buff = array[i];

array[i] = array[i + 1];

array[i + 1] = buff;

}

}

right--;

for (int i=right; i>left; i--) {

if (array[i]<array[i-1]) {

buff = array[i];

array[i] = array[i - 1];

array[i - 1] = buff;

}

}

left++;

} while (left <right);

}

**Сортировка вставками**

public static void insertIntoSort(int[] arr) {

int temp, j;

for(int i = 0; i < arr.length - 1; i++){

if (arr[i] > arr[i + 1]) {

temp = arr[i + 1];

arr[i + 1] = arr[i];

j = i;

while (j > 0 && temp < arr[j - 1]) {

arr[j] = arr[j - 1];

j--;

}

arr[j] = temp;

}

}

}

**Быстрая сортировка**

Его среднее время выполнения - O(n log n). Quick Sort реализуется с помощью рекурсии.

public static void quickSort() {

int startIndex = 0;

int endIndex = ARRAY\_LENGTH - 1;

doSort(startIndex, endIndex);

}

private static void doSort(int start, int end) {

if (start >= end)

return;

int i = start, j = end;

int cur = i - (i - j) / 2;

while (i < j) {

while (i < cur && (array[i] <= array[cur])) {

i++;

}

while (j > cur && (array[cur] <= array[j])) {

j--;

}

if (i < j) {

int temp = array[i];

array[i] = array[j];

array[j] = temp;

if (i == cur)

cur = j;

else if (j == cur)

cur = i;

}

}

doSort(start, cur);

doSort(cur+1, end);

}

**Пирамидальная сортировка**

O(n log n) - Массив можно отсортировать, если на его основе строить и перестраивать сортирующее дерево

class HeapSort {

private static int heapSize;

public static void sort(int[] a) {

buildHeap(a);

while (heapSize > 1) {

swap(a, 0, heapSize - 1);

heapSize--;

heapify(a, 0);

}

}

private static void buildHeap(int[] a) {

heapSize = a.length;

for (int i = a.length / 2; i >= 0; i--) {

heapify(a, i);

}

}

\* Переупорядочивает поддерево кучи начиная с узла i так, чтобы выполнялось

\* основное свойство кучи - a[parent] >= a[child].

private static void heapify(int[] a, int i) {

int l = left(i);

int r = right(i);

int largest = i;

if (l < heapSize && a[i] < a[l]) {

largest = l;

}

if (r < heapSize && a[largest] < a[r]) {

largest = r;

}

if (i != largest) {

swap(a, i, largest);

heapify(a, largest);

}

}

\* Возвращает индекс правого потомка текущего узла

private static int right(int i) {

return 2 \* i + 1;

}

\* Возвращает индекс левого потомка текущего узла

private static int left(int i) {

return 2 \* i + 2;

}

\* Меняет местами два элемента в массиве

private static void swap(int[] a, int i, int j) {

int temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

**Рекурсия**

**Пример 1 – факториал числа**

class Factorial {

int fact(int n) {

int result;

if(n == 1) return 1;

result = fact(n - 1) \* n;

return result;

}

}

class Recursion {

public static void main(String args[]) {

Factorial fact = new Factorial();

System.out.println("Факториал 5 равен " + fact.fact(5));

}

}

**Пример 2 – вывод первых i элементов массива**

class RecursionExample {

int values[];

RecursionExample(int i) {

values = new int[i];

}

// вывести рекурсивно элементы массива

void printArray(int i) {

if(i == 0) {

return;

} else {

printArray(i - 1);

System.out.println("[" + (i - 1) + "] " + values[i - 1]);

}

}

}

class RecursionMain {

private static final int num = 23;

public static void main(String[] args) {

RecursionExample rec = new RecursionExample(num);

int j;

for(j = 0; j < num; j++) {

rec.values[j] = j;

}

rec.printArray(num);

}

}

**Итерация**: for (int i = 1; i <= 10; i++) System.out.println(i);

**Рекурсия:**

public static void main(String[] args) {

recursion(1,10);

}

public static void recursion(int i, int limit) {

System.out.println(i);

if (i<limit) recursion(i+1, limit);

}

**Log4j**

Logger установлен для **отладки** приоритета. - log4j.rootLogger=DEBUG, stdout

**import org.apache.log4j.Logger;**

public class HelloExample{

final static Logger logger = Logger.getLogger(HelloExample.class);

public static void main(String[] args) {

HelloExample obj = new HelloExample();

obj.runMe("mkyong");

}

private void runMe(String parameter){

if(logger.isDebugEnabled()) logger.debug("This is debug : " + parameter);

if(logger.isInfoEnabled()) logger.info("This is info : " + parameter);

logger.warn("This is warn : " + parameter);

logger.error("This is error : " + parameter);

logger.fatal("This is fatal : " + parameter);

}

}

**Вывод:**

2014-07-02 20:52:39 DEBUG HelloExample:19 - This is debug : mkyong

2014-07-02 20:52:39 INFO HelloExample:23 - This is info : mkyong

2014-07-02 20:52:39 WARN HelloExample:26 - This is warn : mkyong

2014-07-02 20:52:39 ERROR HelloExample:27 - This is error : mkyong

2014-07-02 20:52:39 FATAL HelloExample:28 - This is fatal : mkyong

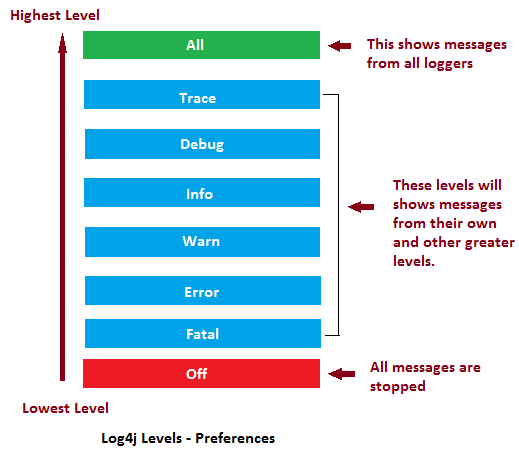
Logger установлен на приоритет ошибки - log4j.rootLogger=error, stdout

**Вывод:**

2014-07-02 20:56:02 ERROR HelloExample:27 - This is error : mkyong

2014-07-02 20:56:02 FATAL HelloExample:28 - This is fatal : mkyong

**Приоритет:**



***Пример, показывающий, как использовать log4j для регистрации исключения.***

import org.apache.log4j.Logger;

public class HelloExample2{

final static Logger logger = Logger.getLogger(HelloExample2.class);

public static void main(String[] args) {

HelloExample2 obj = new HelloExample2();

try{

obj.divide();

}catch(ArithmeticException ex){

logger.error("Sorry, something wrong!", ex);

}

}

private void divide(){

int i = 10 /0;

}

}

**Вывод:**

2014-07-02 21:03:10 ERROR HelloExample2:16 - Sorry, something wrong!

java.lang.ArithmeticException: / by zero

at com.mkyong.HelloExample2.divide(HelloExample2.java:24)

at com.mkyong.HelloExample2.main(HelloExample2.java:14)

**Потоки – IO**

**Байтовый**

import java.io.\*;

public class FileCopy {

public static void main(String args[]) throws IOException {

FileInputStream fileIn = null;

FileOutputStream fileOut = null;

try {

fileIn = new FileInputStream("file.txt");

fileOut = new FileOutputStream("copied\_file.txt");

int a;

// Копирование содержимого файла file.txt

while ((a = fileIn.read()) != -1) {

fileOut.write(a); // Чтение содержимого файла file.txt и запись в файл copied\_file.txt

}

}finally {

if (fileIn != null) {

fileIn.close();

}

if (fileOut != null) {

fileOut.close();

}

}

}

}

**Символьный**

import java.io.\*;

public class FileCopy {

public static void main(String args[]) throws IOException {

FileReader fileIn = null;

FileWriter fileOut = null;

try {

fileIn = new FileReader("file.txt");

fileOut = new FileWriter("copied\_file.txt");

int a;

while ((a = fileIn.read()) != -1) {

fileOut.write(a);

}

} finally {

if (fileIn != null) {

fileIn.close();

}

if (fileOut != null) {

fileOut.close();

}

}

}

}

\*\*\*\*\*\*\*\*

char c[] = {'a','b','c'};

OutputStream output = new FileOutputStream("file.txt"); // Создание текстового файла

for(int i = 0; i < c.length; i++) {

output.write(c[i]); // Запись каждого символа в текстовый файл

}

output.close();

InputStream input = new FileInputStream("file.txt");

int size = input.available(); // сколько байт можно прочесть

for(int j = 0; j < size; j++) {

System.out.print((char)input.read() + " "); // Чтение текстового файла посимвольно

}

input.close();

}catch(IOException e) {

System.out.print("Exception");

}

}

}