

# Java Основы и введение

## Ввод при помощи **Scanner**

Открытие терминала  
Scanner \*название\* = new Scanner()

Считывание строчки  
String \*переменная\* = \*название\*.nextLine()

Другие считывания:  
\*переменная\* = \*название\*.nextInt()

\*переменная\* = \*название\*.nextDouble()  
есть ещё куча считываний разных данных

проверка есть ли ещё ввод  
\*название\*.hasNext() - все что угодно  
\*название\*.hasNextInt() - есть ли ещё число  
\*название\*.hasNextDouble() - есть ли дробное число

Закрытие сканера: \*название\*.close()

Все остальные методы есть тут:

<https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/util/Scanner.html>

## Работа с файлом

Начинаем работать с файлом  
File file = new File("Example.txt");

Создание файла  
file.createNewFile();

### Запись

Создание объекта FileWriter

FileWriter writer = new FileWriter(file, true); -  
каждая запись - добавление в файл

FileWriter writer = new FileWriter(file, false); -  
перезапись файла

Запись содержимого в файл  
writer.write("текст");  
writer.close();

### Чтение

Создание объекта FileReader  
FileReader file = new FileReader(file);

char [] a = new char[(int) file.length()];  
Количество символов, которое будем считывать

file.read(a); - Чтение содержимого в массив

for(char c : a)  
System.out.print(c); - Вывод символов один  
за другими  
file.close(); - закрытие

## Структура файла

public class Main { --- создание класса

public static void main(String[] args) {  
} --- функция запуска вашего файла

static void ex0(){} -- функция ничего не  
возвращает  
Чтобы выводило - вместо void пишите  
свои данные  
}

## Модификаторы доступа

private - доступно только  
из класса, где создана  
функция

protected - доступно из  
вашего класса, а так же  
класса наследников

public - доступна отовсюду

## Collection

List  
Хранит упорядоченные элементы (кот. могут  
быть одинаковые). представляет  
функциональность простых списков.

ArrayList  
преимущество в навигации. Представляет  
простой список, аналогичный массиву, за тем  
исключением, что количество элементов в  
нем не фиксировано.

LinkedList  
преимущество во вставке и удалении элементов.  
Соединяет функциональность работы со списком  
и функциональность очереди. Это список, в  
котором у каждого элемента есть ссылка на  
предыдущий и следующий элементы

Queue  
для реализации в виде очереди.  
Определяет поведение класса в качестве  
однонаправленной очереди.

Deque  
определяет поведение двунаправленной  
очереди, которая работает как обычная  
однонаправленная очередь, либо как стек,  
действующий по принципу LIFO (последний  
вошел - первый вышел).

PriorityQueue  
работают по принципу FIFO (first in first out).  
Упорядоченная очередь. Этот класс может быть  
полезен, например, для нахождения п  
минимальных чисел в большом  
неупорядоченном списке

Set  
не содержит повторяющихся элементов.  
Неупорядоченное множество уникальных  
элементов

SortedSet  
предназначен для создания коллекций, который  
хранят элементы в отсортированном виде  
(сортировка по возрастанию). Хранит только  
уникальные значения.

TreeSet  
упорядочивает элементы по их значениям.  
Представляет структуру данных в виде дерева,  
в котором все объекты хранятся в  
отсортированном виде по возрастанию

HashSet  
упорядочивает элементы по их хэш ключам.  
Представляет хеш-таблицу. Хеш-таблица  
представляет такую структуру данных, в которой  
все объекты имеют уникальный ключ или хеш-  
код. Данный ключ позволяет уникально  
идентифицировать объект в таблице.

```
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.net.SocketTimeoutException;
// задание: записать слово TEST в файл 10 раз
public class Main {
    public static void main(String[] args) {
        Integer n = 10;
        String text = "TEST";
        String file_name = "1.txt";

        File file = new File(file_name);

        try{
            FileWriter writer = new FileWriter(file,false);
            for (int i = 0; i < n; i++){
                writer.write(text);
                writer.write("\n");
            }
            writer.close();
            System.out.println("Получилось!");
        }
        catch (Exception e){
            System.out.println("Что то пошло не так");
        }
    }
}
```

```
try{
    FileReader f = new FileReader(file);
    char[] a = new char[(int) file.length()];
    f.read(a);
    for(char c: a){
        System.out.print(c);
    }
    f.close();
}
catch (Exception e){
    System.out.println("Что то не так");
}
}
```