

# Программирование на языке Java

**Тема 41. Работа с файлами  
(ввод и вывод)**

# Класс Scanner

---

**Scanner** – класс, который читает форматный ввод и преобразует его в бинарную форму.

**Scanner** позволяет читать данные с клавиатуры, из файла на диске, из строки.

# Класс Scanner. Чтение с клавиатуры

---

## Общий вид:

```
Scanner in1 = new Scanner(System.in);  
Scanner in2 = new Scanner(System.in, "cp1251");
```

# Класс Scanner. Чтение из строки

---

## Общий вид:

```
Scanner in =  
    new Scanner("10 99,88 сканирование это  
просто");  
int a = in.nextInt();           // 10  
double b = in.nextDouble();    // 99.88  
String s = in.next();          // "сканирование"
```

# Класс Scanner. Чтение из файла

---

## Общий вид:

```
File file = new File ("in.txt");  
Scanner in1 = new Scanner(file);  
Scanner in2 = new Scanner(file, "cp1251");
```

**Внимание!** Для того, чтобы работать с классом `File`, нужно подключить пакет `java.io` с помощью команды

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

**Внимание!** В методе `main` нужно указать исключение ввода-вывода, которое может генерироваться этим методом

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

# Основы сканирования

---

**Scanner** читает **лексемы** из некоторого источника (с клавиатуры, из строки, из файла), который указан при создании объекта **Scanner**.

**Лексема** – порция ввода, отделенная набором разделителей, которыми по умолчанию являются пробелы.

# Процедура сканирования

---

1. Определите, доступен ли специфический тип ввода вызовом одного из методов класса **Scanner** `hasNextX()`, где **X** – нужный тип данных.
2. Если ввод доступен, читайте его одним из методов класса **Scanner** `nextX()`.
3. Повторяйте процесс до завершения ввода.

# Пример. Чтение целых чисел с клавиатуры

---

```
Scanner in = new Scanner (System.in);  
    int i;  
    while (in.hasNextInt()) {  
        i = in.nextInt();  
        // ...  
    }
```

Цикл `while` остановится, как только следующая лексема окажется не целым числом.

# Некоторые методы hasNext – 1

Метод	Описание
<code>boolean hasNext()</code>	Возвращает <code>true</code> , если доступна для чтения лексема любого типа.
<code>boolean hasNextBoolean()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>boolean</code> .
<code>boolean hasNextByte()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>byte</code> .
<code>boolean hasNextDouble()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>double</code> .
<code>boolean hasNextFloat()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>float</code> .
<code>boolean hasNextInt()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>int</code> .

# Некоторые методы hasNext – 2

---

Метод	Описание
<code>boolean hasNextLine()</code>	Возвращает <code>true</code> , если доступна строка ввода.
<code>boolean hasNextLong()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>long</code> .
<code>boolean hasNextShort()</code>	Возвращает <code>true</code> , если доступно для чтения значение типа <code>short</code> .

# Некоторые методы `next` – 1

Метод	Описание
<code>String next()</code>	Возвращает следующую лексему любого типа из входного источника.
<code>boolean nextBoolean()</code>	Возвращает следующую лексему как значение типа <code>boolean</code> .
<code>byte nextByte()</code>	Возвращает следующую лексему как значение типа <code>byte</code> .
<code>double nextDouble()</code>	Возвращает следующую лексему как значение типа <code>double</code> .
<code>float nextFloat()</code>	Возвращает следующую лексему как значение типа <code>float</code> .
<code>int nextInt()</code>	Возвращает следующую лексему как значение типа <code>int</code> .

# Некоторые методы `next` – 2

---

Метод	Описание
<code>String nextLine()</code>	Возвращает следующую строку ввода.
<code>long nextLong()</code>	Возвращает следующую лексему как значение типа <code>long</code> .
<code>short nextShort()</code>	Возвращает следующую лексему как значение типа <code>short</code> .

## Пример. Чтение с клавиатуры

---

```
Scanner in = new Scanner (System.in);
int count = 0; double sum = 0;
while (in.hasNext()) {
    if (in.hasNextDouble()) {
        sum += in.nextDouble();
        count++; }
    else
        break;
}
System.out.printf("Среднее = %f", sum / count);
```

# Пример. Чтение из файла – 1

---

Рассмотрим ту же самую задачу, но с чтением данных из файла. Пусть имеется файл `in.txt`, который находится в папке проекта.

`in.txt`

```
2 3,4 5 6 7,4 9,1 10,5
```

# Пример. Чтение из файла – 2

---

Подгружаем пакет для работы с классом  
`File`

Метод `main` может генерировать  
исключения ввода-вывода

```
import java.util.*;
import java.io.*;
public class Main {
public static void main(String[] args)
    throws IOException {
File file = new File("in.txt");
Scanner in = new Scanner (file, "cp1251");

int count = 0;
double sum = 0;
```

Обращение к файлу `in.txt`

## Пример. Чтение из файла – 3

```
while (in.hasNext()) {  
    if (in.hasNextDouble()) {  
        sum += in.nextDouble();  
        count++; }  
    else  
        break;  
}  
System.out.printf("Среднее = %f", sum / count);  
}}
```

Дальнейший код  
повторяет пример  
с чтением с  
клавиатуры

# Класс `PrintWriter`

---

`PrintWriter` – класс, который применяется для записи файла.

**Внимание!** Для того, чтобы работать с классом `PrintWriter`, нужно подключить пакет `java.io` с помощью команды

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

**Внимание!** В методе `main` нужно указать исключения ввода-вывода, которые могут генерироваться этим методом

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

# Конструкторы класса `PrintWriter`

---

Общий вид:

```
PrintWriter out = new PrintWriter(<имя файла>);
```

Каждый раз при новой записи предыдущие данные будут стираться.

# Некоторые методы `PrintWriter`

Метод	Описание
<code>void close()</code>	Закрывает поток. Последующие попытки записи генерируют исключения <code>IOException</code> .
<code>void print(тип x)</code>	Записывает значение <code>x</code> в выходной поток
<code>void println(тип x)</code>	Записывает значение <code>x</code> и перевод строки в выходной поток
<code>void printf("форматная строка", список_аргументов)</code>	Записывает в выходной поток отформатированную строку

## Пример. Запись в файл

---

```
import java.io.*;
public class Main {
public static void main(String[] args)
    throws IOException {
    PrintWriter out = new PrintWriter("out.txt");
    out.print ("2 3,4 5 6 7,4 9,1 10,5");
    out.close();
}
}
```

Файл `out.txt` будет помещен в папке проекта.

# Пример. Чтение и запись – 1

---

student.in

Дмитрий 24

Петр 23

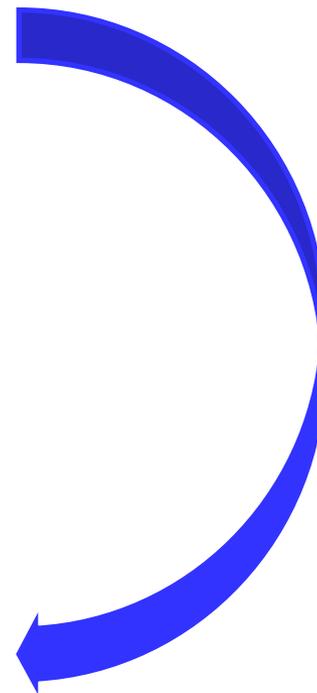
Ольга 22

student.out

Имя: Дмитрий ; возраст: 24

Имя: Петр; возраст: 23

Имя: Ольга; возраст: 22



## Пример. Чтение и запись – 2

---

```
import java.util.*;
import java.io.*;
public class Main {
    public static void main(String[] args)
        throws IOException {
        File file = new File("student.in");
        Scanner in = new Scanner(file);
        PrintWriter out =
            new PrintWriter("student.out");
```

## Пример. Чтение и запись – 3

---

```
while (in.hasNext()) {
    String name = in.next();
    int age = in.nextInt();
    out.printf("Имя: %s; возраст: %d\n", name,
age);
}
in.close();
out.close();
}
```

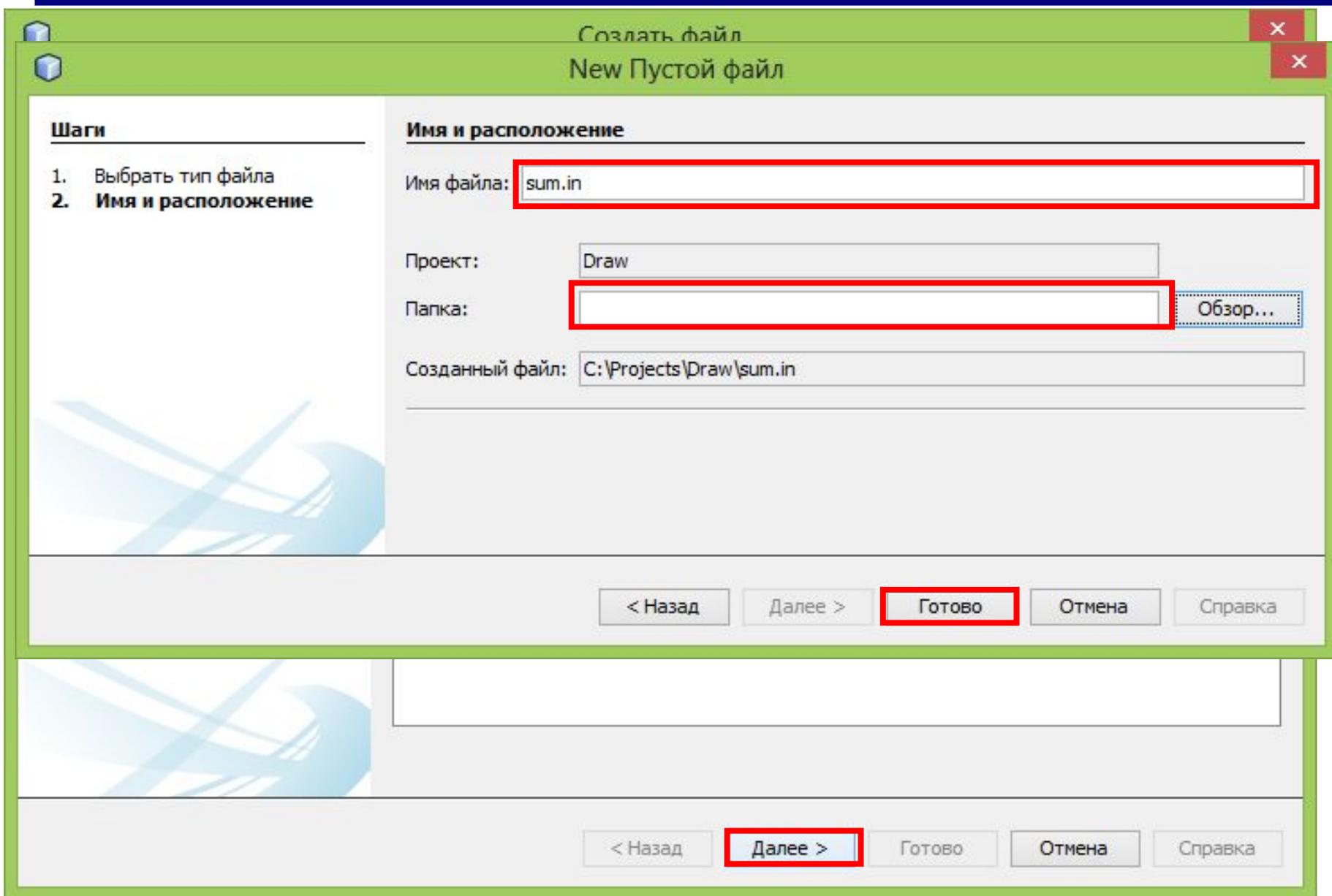
# Создание текстовых файлов в NetBeans

---

Для создания текстовых файлов

1. Вызовите меню «Файл» - «Создать файл»,
2. Выберите категорию «Прочее», тип файла «Пустой файл».
3. Задайте имя файла с расширением, например `sum.in`
4. В поле «Папка» должно быть пусто.
5. Нажмите кнопку «Готово».
6. В открывшемся окне отредактируйте входные данные для программы и сохраните их.

# Создание текстовых файлов в NetBeans



# Задание

---

**Задача 1.** Напишите программу, которая считывает **2 целых числа** из файла и выводит сумму этих чисел в другой файл.

**Задача 2.** Напишите программу, которая считывает **все целые числа** из файла и выводит сумму этих чисел в другой файл.

# Класс `FileWriter`

---

`FileWriter` – класс, который применяется для записи файла.

**Внимание!** Для того, чтобы работать с классом `FileWriter`, нужно подключить пакет `java.io` с помощью команды

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

**Внимание!** В методе `main` нужно указать исключение ввода-вывода, которое может генерироваться этим методом

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

# Конструкторы класса `FileWriter`

---

## Общий вид:

Каждый раз при записи предыдущие данные будут удаляться из файла.

```
FileWriter out = new FileWriter(<имя файла>);
```

Новые данные будут дописываться в конец файла.

```
FileWriter out = new FileWriter(<имя файла>,  
                                true);
```

# Некоторые методы `FileWriter`

---

Метод	Описание
<code>void close()</code>	Закрывает поток. Последующие попытки записи генерируют <code>IOException</code> .
<code>void write(String str)</code>	Записывает строку <code>str</code> в Выходной Поток

# Пример. Запись в файл

```
import java.io.*;
public class Main {
public static void main(String[] args)
    throws IOException {
    FileWriter out = new FileWriter("out.txt");
    out.write("2 3,4 5 6 7,4 9,1 10,5 end");
    out.close();
}
}
```

Запись данных в файл

Поток закрыт

Файл `out.txt` будет помещен в **папку проекта**.

# Класс `Formatter`

---

`Formatter` – класс, который предлагает преобразования формата, позволяющие отображать числа, строки в любом виде.

## Общий вид:

```
Formatter fmt = new Formatter();  
fmt.format(<форматная строка>,  
          <список аргументов>);
```

**Внимание!** Для того, чтобы работать с классом `Formatter`, нужно подключить пакет `java.util.Formatter`

# Пример использования класса `Formatter`

---

```
Formatter fmt = new Formatter();  
fmt.format("Форматировать %s очень  
просто: %d, %f",  
"с помощью Java", 10, 98.5);
```

Объект `Formatter`, содержащий строку  
«Форматировать с помощью Java очень  
просто: 10, 98,500000»

# Пример. Чтение и запись – 1

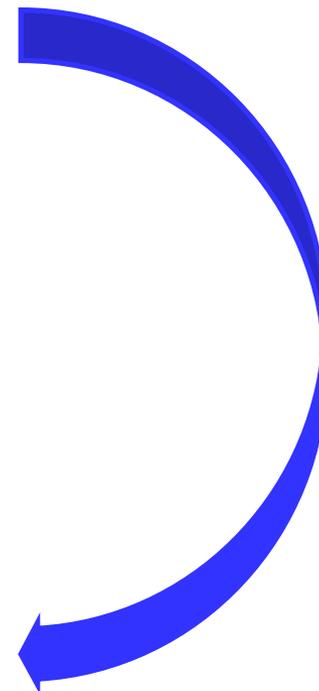
---

in.txt

Василий 24  
Петр 23  
Анна 24

out.txt

Имя: Василий; возраст: 24  
Имя: Петр; возраст: 23  
Имя: Анна; возраст: 24



## Пример. Чтение и запись – 2

---

```
import java.util.*;
import java.io.*;
public class Main {
    public static void main(String[] args)
        throws IOException {
        File file = new File("in.txt");
        Scanner in = new Scanner(file, "cp1251");
        FileWriter out = new FileWriter("out.txt");
        Formatter fmt = new Formatter();
        String s;
```

## Пример. Чтение и запись – 3

---

```
while (in.hasNext()) {
    String name = in.next();
    int age = in.nextInt();
    fmt.format("Имя: %s; возраст: %d\n", name, age);
}
in.close();
s = fmt.toString();
out.write(s);
out.close();
}
```