

Алгоритмизация и программирование

8. Файловые операции

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Способы взаимодействия с пользователем

- ▶ Консольные программы (git)
 - Консоль
 - Вывод (println) – довольно часто

Способы взаимодействия с пользователем

- ▶ Консольные программы (git)
 - Консоль
 - Вывод (println) – довольно часто
 - Ввод (readLine) – КРАЙНЕ редко

Способы взаимодействия с пользователем

- ▶ Консольные программы (git)
 - Консоль
 - Вывод (println) – довольно часто
 - Ввод (readLine) – КРАЙНЕ редко
 - Аргументы командной строки
 - `fun main(args: Array<String>) { ... }`

Способы взаимодействия с пользователем

- ▶ Консольные программы (git)
 - Консоль
 - Вывод (println) – довольно часто
 - Ввод (readLine) – КРАЙНЕ редко
 - Аргументы командной строки
 - `fun main(args: Array<String>) { ... }`
 - Внешние файлы
 - Файлы настроек
 - Файлы с входными данными
 - Файлы с выходными данными
 - Временные файлы
 - Редактируемые файлы ...

Виды файлов (по содержанию)

- ▶ Текстовые
 - Просто текст
 - Упорядоченный текст
 - XML
 - ...

Виды файлов (по содержанию)

- ▶ Текстовые
 - Просто текст
 - Упорядоченный текст
 - XML
 - ...
- ▶ Бинарные
 - PDF
 - JPEG
 - ...

Пример работы с файлами

```
fun alignFile(inputName: String, lineLength: Int, outputName: String) {  
    val writer = File(outputName).bufferedWriter()  
    var currentLineLength = 0  
    fun append(word: String) { ... }  
    for (line in File(inputName).readLines()) {  
        if (line.isEmpty()) {  
            writer.newLine()  
            if (currentLineLength > 0) {  
                writer.newLine()  
                currentLineLength = 0  
            }  
        }  
        for (word in line.split(Regex("\\s+"))) {  
            append(word)  
        }  
    }  
    writer.close()  
}
```


Пример работы с файлами

```
fun alignFile(inputName: String, lineLength: Int, outputName: String) {  
    val writer = File(outputName).bufferedWriter()  
    var currentLineLength = 0  
    fun append(word: String) {  
        if (currentLineLength > 0) {  
            if (word.length + currentLineLength >= lineLength) {  
                writer.newLine()  
                currentLineLength = 0  
            } else {  
                writer.write(" ")  
                currentLineLength++  
            }  
        }  
        writer.write(word)  
        currentLineLength += word.length  
    }  
    // ...  
}
```

File

- ▶ Объект этого типа соответствует файлу
- ▶ = `java.io.File`
- ▶ Функции
 - `readLines(): List<String>`
 - `bufferedWriter(): BufferedWriter`
 - `bufferedReader(): BufferedReader`
 - ...

BufferedWriter

- ▶ Объект этого типа может записывать (буферизованную) информацию в файл или...
- ▶ = `java.io.BufferedWriter`
- ▶ Функции
 - `newLine()`
 - `write(s: String)`
 - `close()`
 - ...

java.io

- ▶ Три уровня работы с файлами
 - Файл = поток байт: `InputStream` / `OutputStream`

java.io

- ▶ Три уровня работы с файлами
 - Файл = поток байт: `InputStream` / `OutputStream`
 - Файл = поток символов:
`InputStreamReader` / `OutputStreamWriter`

java.io

- ▶ Три уровня работы с файлами
 - Файл = поток байт: `InputStream` / `OutputStream`
 - Файл = поток символов:
`InputStreamReader` / `OutputStreamWriter`
 - NB: знает про кодировку!

java.io

- ▶ Три уровня работы с файлами
 - Файл = поток байт: `InputStream` / `OutputStream`
 - Файл = поток символов:
`InputStreamReader` / `OutputStreamWriter`
 - NB: знает про кодировку!
 - Файл состоит из строк:
`BufferedReader` / `BufferedWriter`

java.io

▶ File

- InputStream → `file.getInputStream()`
 - InputStreamReader → `file.reader()`
 - BufferedReader → `file.bufferedReader()`

Кто что умеет?

► Closable: writer.close()

```
fun foo(outputName: String) {  
    val writer = File(outputName).bufferedWriter()  
    try {  
        writer.write(...)  
    }  
    finally {  
        writer.close()  
    }  
}
```

Кто что умеет?

- ▶ Closable: `writer.close()` / `writer.use !`

```
fun foo(outputName: String) {  
    val writer = File(outputName).bufferedWriter()  
    writer.use {  
        it.write(...)  
    }  
}
```

Кто как читает?

- ▶ InputStream
 - inputStream.read(): Int
 - inputStream.read(arr: ByteArray): Int

Кто как читает?

- ▶ **InputStream**
 - `inputStream.read(): Int`
 - `inputStream.read(arr: ByteArray): Int`
- ▶ **InputStreamReader**
 - `reader.read(): Int`
 - `reader.read(arr: CharArray): Int`

Кто как читает?

- ▶ **InputStream**
 - `inputStream.read(): Int`
 - `inputStream.read(arr: ByteArray): Int`
- ▶ **InputStreamReader**
 - `reader.read(): Int`
 - `reader.read(arr: CharArray): Int`
- ▶ **BufferedReader**
 - `bufReader.readLine(): String`
 - `bufReader.readLine(): List<String>`

Кто как пишет?

- ▶ **OutputStream**
 - `outputStream.write(b: Int)`
 - `outputStream.write(arr: ByteArray)`
- ▶ **OutputStreamWriter**
 - `writer.write(c: Int):`
 - `writer.write(arr: CharArray)`
- ▶ **BufferedWriter**
 - `bufWriter.write(s: String)`
 - `bufWriter.newLine() ...`

Кто как пишет?

- ▶ **OutputStream**
 - `outputStream.write(b: Int)`
 - `outputStream.write(arr: ByteArray)`
- ▶ **OutputStreamWriter**
 - `writer.write(c: Int):`
 - `writer.write(arr: CharArray)`
- ▶ **BufferedWriter**
 - `bufWriter.write(s: String)`
 - `bufWriter.newLine() ...`
- ▶ **PrintStream (!) ← `file.printStream()`**
 - `OutputStream + encoding / print / println / format`

Упражнения к лекции

- ▶ См. lesson8/task1 в обучающем проекте
- ▶ Решите хотя бы одну из задач
- ▶ Протестируйте решения с помощью готовых тестов
- ▶ Добавьте ещё хотя бы два тестовых случая
- ▶ Добавьте коммит в свой репозиторий
- ▶ Создайте Pull Request и убедитесь в правильности решения