Алгоритмизация и программирование

5. Текстовый ввод-вывод

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Текстовый ввод-вывод

- Разбор (парсинг) строки

Текстовый ввод-вывод

- Разбор (парсинг) строки
  - Взятой из файла, пришедшей по сети, ...

Разбор строки

- “11:34:45” → число секунд

Разбор строки

- “11:34:45” → число секунд

```
fun timeStrToSeconds(str: String): Int {
    val parts = str.split(":")
    var result = 0
    for (part in parts) {
        val number = part.toInt()
        result = result * 60 + number
    }
    return result
}
```

Функция split

- Существует в нескольких вариантах
- Аргумент(ы) = разделители
  - str.split(":")
  - str.split(" ", ",")

Функция split

- Существует в нескольких вариантах
- Аргумент(ы) = разделители
  - str.split(":")
  - str.split(" ", ",")
- Результат = List<String>

Функция split

- Существует в нескольких вариантах
- Аргумент(ы) = разделители
  - str.split(":")
  - str.split(" ", ",")
- Результат = List<String>
- “11:34:45” → [“11”, “34”, “45”]

Функция split

- Существует в нескольких вариантах
- Аргумент(ы) = разделители
  - str.split(":")
  - str.split(" ", ",")
- Результат = List<String>
- “11:34:45” → [“11”, “34”, “45”]

- Также может работать с регулярными выражениями (RegExp)

Функция toInt()

- Превращение (строки) в целое число
  - “45” → 45
- Есть аналоги для других стандартных типов

Разбор строки

- “11:34:45” → число секунд
- Когда может работать неправильно?

```
fun timeStrToSeconds(str: String): Int {
    val parts = str.split(":")
    var result = 0
    for (part in parts) {
        val number = part.toInt()
        result = result * 60 + number
    }
    return result
}
```

Некорректный случай

- “Alpha”.toInt() = ???

Некорректный случай

- “Alpha”.toInt() = ???

- java.lang.NumberFormatException:
  For input string: "Alpha"

Некорректный случай

- “Alpha”.toInt() = ???

- java.lang.NumberFormatException:
  For input string: "Alpha"
- Произошло исключение (Exception)

Исключения (Java / Kotlin)

- Особый тип данных с дополнительными возможностями; описывает ошибки

Исключения (Java / Kotlin)

- Особый тип данных с дополнительными возможностями; описывает ошибки
- Можно бросать

```
fun throwExample() {
    val e = NumberFormatException("Description")
    throw e // NumberFormatException("Description")
}
```

Исключения (Java / Kotlin)

- Особый тип данных с дополнительными возможностями; описывает ошибки
- Можно бросать

```
fun throwExample() {
    val e = NumberFormatException("Description")
    throw e // NumberFormatException("Description")
}
```

- NumberFormatException = тип

Исключения (Java / Kotlin)

- Особый тип данных с дополнительными возможностями; описывает ошибки
- Можно бросать

```
fun throwExample() {
    val e = NumberFormatException("Description")
    throw e // NumberFormatException("Description")
}
```

- NumberFormatException = тип
- NumberFormatException(…) = конструктор

Стек вызовов

- Пример:

```
at java.lang.NumberFormatException.forInputString
    (NumberFormatException.java:65)
at java.lang.Integer.parseInt(Integer.java:580)
at java.lang.Integer.parseInt(Integer.java:615)
at lesson5.task1.ParseKt.timeStrToSeconds
    (Parse.kt:14)
at lesson5.task1.Tests.timeStrToSeconds
    (Tests.kt:11)
```

Некорректный случай

- “Alpha”.toInt() = ???
- java.lang.NumberFormatException:
  For input string: "Alpha"
- “AA:BB:CC”: а что, собственно, делать?

Некорректный случай

- “Alpha”.toInt() = ???
- java.lang.NumberFormatException:
  For input string: "Alpha"
- “AA:BB:CC”: а что, собственно, делать?
  - Бросить исключение

Некорректный случай

- “Alpha”.toInt() = ???
- java.lang.NumberFormatException:
  For input string: "Alpha"
- “AA:BB:CC”: а что, собственно, делать?
  - Бросить исключение
  - Вернуть невозможный результат

Некорректный случай

- “Alpha”.toInt() = ???
- java.lang.NumberFormatException:
  For input string: "Alpha"
- “AA:BB:CC”: а что, собственно, делать?
  - Бросить исключение
  - Вернуть невозможный результат (-1?)

Обработка исключения

```
fun timeStrToSeconds(str: String): Int {
    val parts = str.split(":")
    var result = 0
    try {
        for (part in parts) {
            val number = part.toInt()
            result = result * 60 + number
        }
        return result
    }
    catch (e: NumberFormatException) {
        return -1
    }
}
```

Исключения (Java / Kotlin)

- Особый тип данных с дополнительными возможностями; описывает ошибки
- Можно поймать (catch) …

Исключения (Java / Kotlin)

- Особый тип данных с дополнительными возможностями; описывает ошибки
- Можно поймать (catch) …
- … в любом месте стека вызовов

Известные исключения

- IndexOutOfBoundsException

Известные исключения

- IndexOutOfBoundsException
- UnsupportedOperationException

Известные исключения

- IndexOutOfBoundsException
- UnsupportedOperationException
- IllegalArgumentException

Известные исключения

- IndexOutOfBoundsException
- UnsupportedOperationException
- IllegalArgumentException
- IllegalStateException

Текстовый ввод-вывод

- Разбор (парсинг) строки
  - Взятой из файла, пришедшей по сети, ...
- Форматирование строки

Составление строки

- Обратная задача
  - “11:34:45” ← число секунд

Составление строки

- Обратная задача
  - “11:34:45” ← число секунд

```
fun timeSecondsToStr(seconds: Int): String {
    val hour = seconds / 3600
    val minute = (seconds % 3600) / 60
    val second = seconds % 60
    return String.format("%02d:%02d:%02d",
                         hour, minute, second)
}
```

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева
  - %2d – вывести целое число в 2 позициях

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева
  - %2d – вывести целое число в 2 позициях
  - %d – вывести целое число

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева
  - %2d – вывести целое число в 2 позициях
  - %d – вывести целое число
  - %20s – вывести строку в 20 позициях

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева
  - %2d – вывести целое число в 2 позициях
  - %d – вывести целое число
  - %20s – вывести строку в 20 позициях
  - %lf – вывести Double

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева
  - %2d – вывести целое число в 2 позициях
  - %d – вывести целое число
  - %20s – вывести строку в 20 позициях
  - %lf – вывести Double
  - %le – вывести Double в формате 1.2e+34

Модификаторы формата

- Начинаются с %
  - %02d – вывести целое число в 2 позициях, дополнив его нулями слева
  - %2d – вывести целое число в 2 позициях
  - %d – вывести целое число
  - %20s – вывести строку в 20 позициях
  - %lf – вывести Double
  - %le – вывести Double в формате 1.2e+34
  - %6.2lf – вывести Double в 6 позициях с 2 знаками после десятичной точки
  - ... (см. Wikipedia, printf_format_string)

Консольный ввод

- Используется readLine(): String?

Консольный ввод

- Используется readLine(): String?

```
fun main(args: Array<String>) {
    println("Введите время в формате ЧЧ:ММ:СС")
    val line = readLine()
    if (line != null) {
        val seconds = timeStrToSeconds(line)
        if (seconds == -1) {
            println("Строка $line не в формате ЧЧ:ММ:СС")
        }
        else {
            println("Прошло секунд с начала суток: $seconds")
        }
    }
    else {
        println("Достигнут <конец файла. Программа прервана")
    }
}
```

Что такое Type? и null

- null =
  константа: некорректная (нулевая) ссылка

Что такое Type? и null

- null =
  константа: некорректная (нулевая) ссылка
- Type? = тип: ОДЗ(Type?) = ОДЗ(Type) U {null}

Что такое Type? и null

- null =
  константа: некорректная (нулевая) ссылка
- Type? = тип: ОДЗ(Type?) = ОДЗ(Type) U {null}
- В Котлине, Type? всегда ссылочный

Что такое Type? и null

- null =
  константа: некорректная (нулевая) ссылка
- Type? = тип: ОДЗ(Type?) = ОДЗ(Type) U {null}
- В Котлине, Type? всегда ссылочный

- Type? нельзя использовать как Type

```
println("Введите время в формате ЧЧ:ММ:СС")
val line = readLine() // String?
val seconds = timeStrToSeconds(line) // Error!
```

Type? → Type

```
if (line != null) { // line: Type }
```

Type? → Type

```
if (line != null) { // line: Type }

val notNullLine = line ?: "" // : Type
```

Упражнения к лекции

- См. lesson5/task1 в обучающем проекте
- Решите хотя бы одно из заданий
- Протестируйте решение с помощью готовых тестов
- Добавьте ещё хотя бы один тестовый случай
- Напишите main с консольным вводом входных данных и выводом результатов
- Добавьте коммит в свой репозиторий
- Создайте Pull Request и убедитесь в правильности решения