Алгоритмизация и программирование

6. Простые классы

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Составные типы данных

- Включают в себя вложенные элементы
  - У каждого из них свой тип
- Списки, строки, коллекции, массивы…
  - Типы вложенных элементов одинаковы
- Классы
  - Типы вложенных элементов произвольны

Структурирование данных

```
fun distance(x1: Double, y1: Double,
             x2: Double, y2: Double) =
        sqrt(sqr(x2 - x1) + sqr(y2 - y1))
```

Структурирование данных

```
fun distance(x1: Double, y1: Double,
             x2: Double, y2: Double) =
        sqrt(sqr(x2 - x1) + sqr(y2 - y1))

// Проблема 1: много параметров
```

Структурирование данных

```
fun distance(x1: Double, y1: Double,
             x2: Double, y2: Double) =
        sqrt(sqr(x2 - x1) + sqr(y2 - y1))

// Проблема 1: много параметров
// Проблема 2: можно перепутать порядок
```

Структурирование данных

```
fun distance(x1: Double, y1: Double,
             x2: Double, y2: Double) =
        sqrt(sqr(x2 - x1) + sqr(y2 - y1))

// Проблема 1: много параметров
// Проблема 2: можно перепутать порядок

fun distance(p1: Point, p2: Point) =
        sqrt(sqr(p2.x – p1.x) + sqr(p2.y - p1.y))
```

Структурирование данных

```
fun distance(x1: Double, y1: Double,
             x2: Double, y2: Double) =
        sqrt(sqr(x2 - x1) + sqr(y2 - y1))

// Проблема 1: много параметров
// Проблема 2: можно перепутать порядок

class Point(val x: Double, val y: Double)

fun distance(p1: Point, p2: Point) =
        sqrt(sqr(p2.x – p1.x) + sqr(p2.y - p1.y))
```

Пример использования

```
fun usePoints() {
    val a = Point(0.0, 3.0)
    val b = Point(x = 4.0, y = 0.0)
    println(distance(a, b)) // 5.0
}
```

Объекты (экземпляры) класса

- a, b – объекты (экземпляры) класса Point
- У каждого из них есть свойства x и y

Свойства (классов)

- x, y – свойства Point

Свойства (классов)

- x, y – свойства Point
- size – свойство List
- length – свойство String

Свойства (классов)

- x, y – свойства Point
- size – свойство List
- length – свойство String

- Свойство – можно получить (по имени), имея объект класса
- Свойства, заданные как var, можно также изменять

Функции класса

```
class Point(val x: Double, val y: Double) {
    fun distance(other: Point): Double =
        sqrt(sqr(x - other.x) +
             sqr(y - other.y))
}
```

Функции класса

```
class Point(val x: Double, val y: Double) {
    fun distance(other: Point): Double =
        sqrt(sqr(x - other.x) +
             sqr(y - other.y))
}

fun usePoints() {
    val a = Point(0.0, 3.0)
    val b = Point(4.0, 0.0)
    println(a.distance(b)) // 5.0
}
```

Инфиксные функции

```
class Point(val x: Double, val y: Double) {
    infix fun distance(other: Point): Double =
        sqrt(sqr(x - other.x) +
             sqr(y - other.y))
}

fun usePoints() {
    val a = Point(0.0, 3.0)
    val b = Point(4.0, 0.0)
    println(a distance b) // infix call
}
```

Сравнение на равенство

```
class Square(val column: Int, val row: Int)
```

Сравнение на равенство

```
class Square(val column: Int, val row: Int)


fun main(args: Array<String>) {
    val first = Square(3, 6)
    val second = Square(3, 6)
    println(first == second)
}
```

Сравнение на равенство

```
data class Square(val column: Int, val row: Int)


fun main(args: Array<String>) {
    val first = Square(3, 6)
    val second = Square(3, 6)
    println(first == second)
}
```

Строковое представление

```
class Square(val column: Int, val row: Int) {
    override fun toString() = "$row - $column"
}
```

Строковое представление

```
class Square(val column: Int, val row: Int) {
    override fun toString() = "$row - $column"
}

fun main(args: Array<String>) {
    val first = Square(3, 6)
    println(first)
}
```

Строковое представление

```
data class Square(val column: Int,
                  val row: Int)

fun main(args: Array<String>) {
    val first = Square(3, 6)
    println(first)
}
```

Включение классов

```
data class Triangle(val a: Point,
                    val b: Point,
                    val c: Point)

data class Segment(val begin: Point,
                   val end: Point)
```

Включение классов

```
data class Triangle(val a: Point,
                    val b: Point,
                    val c: Point)

data class Segment(val begin: Point,
                   val end: Point)

fun main(args: Array<String>) {
    val t = Triangle(Point(0.0, 0.0),
                     Point(3.0, 0.0),
                     Point(0.0, 4.0))
    println(t.b.x) // 3.0
}
```

Сравнение отрезков: equals

```
data class Segment(val begin: Point, val end: Point) {

    override fun equals(other: Any?) =
        other is Segment &&
        ((begin == other.begin && end == other.end) ||
         (begin == other.end   && end == other.begin))
}
```

Any и Any?

- Any = любой тип (но не null)

Any и Any?

- Any = любой тип (но не null)
- Any? = любой тип (в том числе null)

Any и Any?

- Any = любой тип (но не null)
- Any? = любой тип (в том числе null)

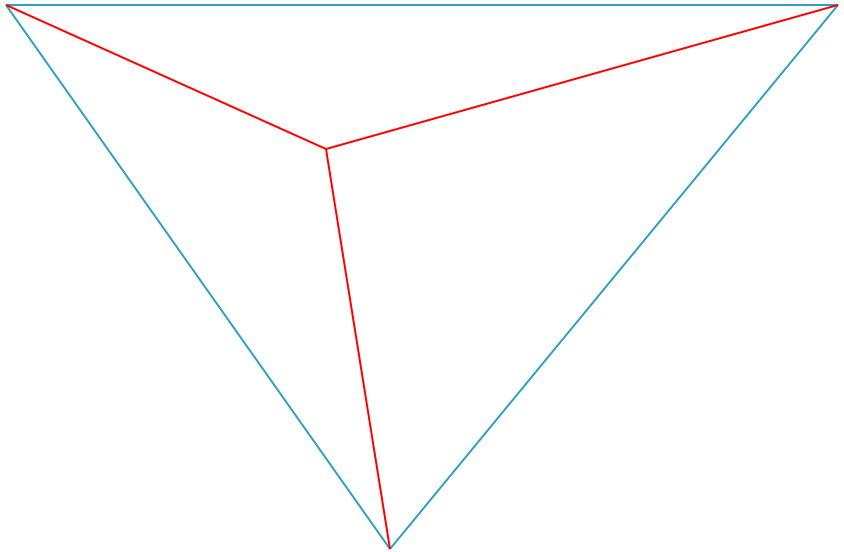
- Имея Any, можно делать **equals**

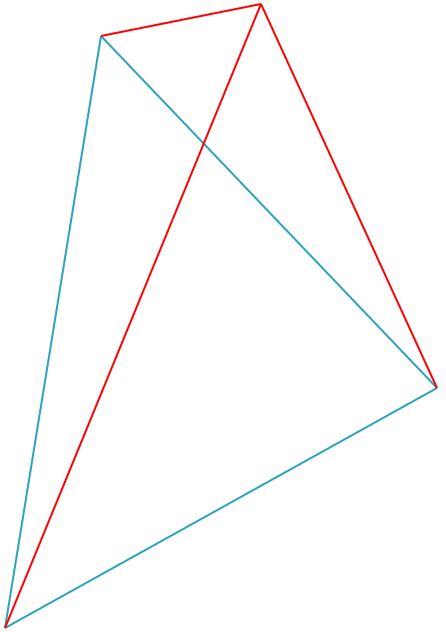
Any и Any?

- Any = любой тип (но не null)
- Any? = любой тип (в том числе null)

- Имея Any, можно делать **equals**
- Имея Any?, можно делать **toString**

Задача: верно ли, что точка внутри треугольника?





Задача: верно ли, что точка внутри треугольника?

```
data class Triangle(val a: Point,
                    val b: Point,
                    val c: Point) {
    fun halfPerimeter() = 0.5 *
        (a.distance(b) + b.distance(c) + c.distance(a))

    fun area(): Double {
        val p = halfPerimeter()
        return sqrt(p * (p - a.distance(b)) *
            (p - b.distance(c)) * (p - c.distance(a)))
    }
}
```

Задача: верно ли, что точка внутри треугольника?

```
data class Triangle(val a: Point,
                    val b: Point,
                    val c: Point) {

    fun contains(p: Point): Boolean {
        val abp = Triangle(a, b, p)
        val bcp = Triangle(b, c, p)
        val cap = Triangle(c, a, p)
        return abp.area() + bcp.area() +
               cap.area() <= area()
    }
}
```

Готовые классы: Pair / Triple

- ```
  data class Pair<A, B>
  (val first: A, val second: B)
  ```
- ```
  data class Triple<A, B, C>
  (val first: A, val second: B, val third: C)
  ```

- A, B, C – произвольные типы

Пример: Triple

- “11:34:45” → число секунд

```
fun timeStrToSeconds(
    str: String
): Triple<Int, Int, Int> {
    val parts = str.split(":").map {
        it.toInt()
    }
    return Triple(parts[0], parts[1], parts[2])
}
```

Пример: Triple

- “11:34:45” → число секунд

```
fun useTimeStrToSeconds() {
    val triple = timeStrToSeconds("11:34:45")
    val hh = triple.first
    val mm = triple.second
    val ss = triple.third
    // или: деструктурирование
    val (hours, minutes, seconds) =
            timeStrToSeconds("11:34:45")
}
```

Деструктурирование: list.withIndex()

```
fun test() {
    val list = listOf("abc", "def")
    for ((index, elem) in list.withIndex()) {
        println("#$index: $elem")
    }
}
```

Деструктурирование: list.withIndex()

```
fun test() {
    val list = listOf("abc", "def")
    for ((index, elem) in list.withIndex()) {
        println("#$index: $elem")
    }
}
// list.withIndex()
// возвращает список пар «индекс, элемент»
// в данном случае [(0, "abc"), (1, "def")]
```

Упражнения к лекции

- См. lesson6/task1 и tasks2 в обучающем проекте
  - Task1 = геометрические задачи
  - Task2 = шахматные задачи
  - Можно выбрать что-то одно или решить по задаче из обеих групп
- Решите хотя бы одно из заданий
- Протестируйте решение с помощью готовых тестов
- Добавьте ещё хотя бы один тестовый случай
- Добавьте коммит в свой репозиторий
- Создайте Pull Request и убедитесь в правильности решения