

Программирование на Java

1. Вводная лекция

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Организация занятий

- ▶ Лекции, 2 часа в неделю
 - Отчётность – экзамен в конце семестра
 - Экзамен – собеседование по теории, с оценкой
- ▶ Практика, 1 (?) час в неделю
 - Три задания, за каждое ставится оценка
 - Отчётность – курсовой проект с оценкой
 - Оценка за курсовой проект =
Сумма оценок за три задания / 3
 - «5» за курсовой проект до конца зачётной недели
→ +1 балл к экзаменационной оценке
- ▶ Самостоятельная работа студентов
 - Не менее 2–3 часов в неделю

Практика

1. Проектирование класса (5 недель)
 - Язык = Java или Kotlin
 - Аттестация после 9 марта
2. Консольное приложение (5.5 недель)
 - Язык = Java или Kotlin
 - Аттестация после 15 апреля
3. GUI-приложение (5.5 недель)
 - Язык = Java или Kotlin
 - Курсовой проект
 - Защита в конце семестра (скорее всего в форме беседы с преподавателем)
 - Отчёт по курсовому проекту

1. Проектирование класса – два пути

- ▶ Если на Котлине
 - Или делаем один класс `FixedPointNumber / UnsignedBigInteger`
 - Или делаем два класса:
один из урока 11 и один из урока 12
 - Сдаём через систему Котоед
 - Тесты дописываем
- ▶ Если на Java
 - Делаем один класс с нуля (задания раздают преподаватели практики)
 - Сами пишем и класс, и тесты
 - Сдаём в виде репозитория на GitHub

2. Консольное приложение

3. GUI-приложение

- ▶ На выбор, Kotlin или Java
- ▶ Пишем с нуля
- ▶ Не забываем писать тесты
- ▶ Сдаём в виде репозитория на GitHub

Что требуется

- ▶ JDK 1.8 или 1.11
 - Скачать с сайта Oracle и установить
- ▶ IntelliJ IDEA Community Edition (бесплатна)
 - Скачать с сайта JetBrains и установить
 - Альтернативы: Eclipse, Netbeans
- ▶ GitHub account
 - Рецензирование кода – исключительно там!
 - Оценивается, в том числе, аккуратность ведения проекта на GitHub
- ▶ JUnit (или другая библиотека тестирования)
 - Написание тестов обязательно!

Структура лекций

- Типы, операции, конструкции
- Классы, интерфейсы, наследование
- Коллекции и утилиты
- GUI на Java: AWT, Swing, JavaFX
- Многопоточные приложения на Java

Литература

- ❑ Ильдар Хабибуллин. Java 7. СПб.: БХВ-Петербург, 2012
- ❑ Арнольд К., Гослинг Дж., Холмс Д. Язык программирования Java, 3-е издание. М.: Издательский дом «Вильямс», 2001.
- ❑ James Gosling, Bill Joy, Guy Steele, Gilad Bracha. The Java Language Specification, Second Edition.
- ❑ Joshua Bloch. Effective Java: Programming Language Guide, second edition. ISBN 978-0-321-35668-0, 2008.

Полезные ссылки

- ❑ Проект с примерами
 - ❑ <https://github.com/Kotlin-Polytech/FromKotlinToJava>
- ❑ Страничка курса
 - ❑ <http://kspt.icc.spbstu.ru/course/javaintro>
- ❑ KotlinAsFirst, chapter11, chapter12

Java vs Kotlin

Kotlin

- Уже частично изучен

- Разработка идёт быстрее

Java

- Лекции этого семестра про Java

- Более востребован

Версии Java SE

- ❑ Java SE 1.11 – 2018 год, небольшие изменения в языке / библиотеке
- ❑ Java SE 1.8 – 2014 год, довольно существенные усовершенствования
- ❑ Java SE 1.6 – 2006 год

Простейшая программа на Java (1, hello world)

```
// Hello.java
package test;
public class Hello {
    public static void main(String[] args) {
        System.out.println("Здравствуй, мир!");
    }
}
```

Простейшая программа на Java (2, square)

```
// Math.java
package test;
public class Math {
    public static int sqr(int x) {
        int y = x * x;
        return y;
    }
}
```

Классы в Java

- ❑ Класс – структурный элемент программы
- ❑ По правилам Java, **все** прочие элементы программы должны находиться внутри классов
- ❑ ООП – класс описывает какой-либо объект / понятие и объединяет в себе **данные** и **функции для работы с ними**

Классы в Java

- ❑ Класс – структурный элемент программы
- ❑ По правилам Java, **все** прочие элементы программы должны находиться внутри классов
- ❑ ООП – класс описывает какой-либо объект / понятие и объединяет в себе **данные** и **функции для работы с ними**
- ❑ Java: класс = поля + методы
- ❑ Kotlin: класс = свойства + функции

Функции vs Методы

- Приблизительно одно и то же
- Методы ~ Функции класса
- Так как в Java все функции принадлежат классам → Методы

Поля vs Свойства

- РАЗНЫЕ вещи
- Поле = Член-данное класса

Поля vs Свойства

- ❑ РАЗНЫЕ вещи
- ❑ Поле = Член-данное класса
- ❑ Свойство = что-то в классе, что можно читать (опционально также писать)
 - ❑ В том числе поле!

Поля vs Свойства

- ❑ РАЗНЫЕ вещи
- ❑ Поле = Член-данное класса
- ❑ Свойство = что-то в классе, что можно читать (опционально также писать)
 - ❑ В том числе поле!
 - ❑ Не факт, что значение свойства где-то хранится!

Поля vs Свойства

- Пример – Планета
- Планета описывается массой и радиусом – это поля класса «Планета»

Поля vs Свойства

- Пример – Планета
- Планета описывается массой и радиусом – это поля класса «Планета»
- На их основе можно рассчитать ускорение свободного падения на поверхности – это свойство класса «Планета»

Видимость в Java

- `public` – класс (метод, поле) видят все
- (no modifier) – класс (метод, поле) видят все в том же пакете (`package private`)

Видимость в Java

- `public` – класс (метод, поле) видят все
- (no modifier) – класс (метод, поле) видят все в том же пакете (`package private`)
- `private` – метод / поле видят все внутри класса

Видимость в Kotlin

- (no modifier) – видят все
- private – видят все внутри класса

Статические и обычные методы класса

- ❑ Отличаются наличием (или отсутствием) модификатора **static**
- ❑ Статические методы
 - ❑ ~ замена глобальных
 - ❑ Вызов: `Math.sqrt`, где `Math` – имя класса
- ❑ Нестатические методы
 - ❑ Привязаны к конкретному объекту и вызываются через него
 - ❑ Вызов: `list.size()`

Типы в Kotlin / Java

- Kotlin
 - `val (var) name: (Type) (= ...) (;)`
- Java
 - `Type name (= ...);`
 - **JDK 10+:**
 - `var name = ...;`

Типы в Java

- Примитивные
 - Называются с маленькой буквы
 - byte, short, int, long,
float, double, char, boolean

Типы в Java

- Примитивные
 - Называются с маленькой буквы
 - byte, short, int, long,
float, double, char, boolean

- Ссылочные
 - Все остальные, называются с большой буквы

Типы в Java

- Прimitives
 - Называются с маленькой буквы
 - byte, short, int, long, float, double, char, boolean

- Ссылочные
 - Все остальные, называются с большой буквы

- Kotlin
 - Все типы ссылочные или притворяются ими

Типы-обёртки в Java

- Ссылочные аналоги примитивных типов
 - Byte, Short, **Integer**, Long, Float, Double, **Character**, Boolean

Примитивные типы

- Целые
 - byte (1 байт, -128...127)
 - short (2 байта, -32768...32767)
 - int (4 байта, $-2^{31} \dots 2^{31}-1$)
 - long (8 байт, $-2^{63} \dots 2^{63}-1$)

Примитивные типы

- Целые
 - byte (1 байт, -128...127)
 - short (2 байта, -32768...32767)
 - int (4 байта, $-2^{31} \dots 2^{31}-1$)
 - long (8 байт, $-2^{63} \dots 2^{63}-1$)
- Вещественные
 - float (4 байта)
 - double (8 байт)

Примитивные типы

- Целые
 - byte (1 байт, -128...127)
 - short (2 байта, -32768...32767)
 - int (4 байта, $-2^{31} \dots 2^{31}-1$)
 - long (8 байт, $-2^{63} \dots 2^{63}-1$)
- Вещественные
 - float (4 байта)
 - double (8 байт)
- Символьный
 - char (2 байта, Unicode, 0...65535)

Примитивные типы

- Целые
 - byte (1 байт, -128...127)
 - short (2 байта, -32768...32767)
 - int (4 байта, $-2^{31} \dots 2^{31}-1$)
 - long (8 байт, $-2^{63} \dots 2^{63}-1$)
- Вещественные
 - float (4 байта)
 - double (8 байт)
- Символьный
 - char (2 байта, Unicode, 0...65535)
- Логический
 - boolean (true или false)

Константы

□ Целые

- 57, +323, -48 (десятичная форма, 4 байта)
- 024, -0634, 0777 (восьмеричная форма)
- 0xabcd, -0x19f (шестнадцатеричная форма)
- 0b010001001 (двоичная форма, **только JDK 7+**)
- 43_934 (форма с _, **только в JDK 7+**)
- 1234567890123L, 0хabcdef1234L (8-байтные)

Константы

□ Целые

- 57, +323, -48 (десятичная форма, 4 байта)
- 024, -0634, 0777 (восьмеричная форма)
- 0xabcd, -0x19f (шестнадцатеричная форма)
- 0b010001001 (двоичная форма, **только JDK 7+**)
- 43_934 (форма с `_`, **только в JDK 7+**)
- 1234567890123L, 0xabcdef1234L (8-байтные)

□ Вещественные

- 37.29, -19.41 (обычная форма, 8 байт)
- 3e+12, -1.1e-7 (экспоненциальная форма)
- 3.6F, -1.0e-1F (4-байтные)

Константы

□ Символьные

- 'a', '?', ' ', '\n', '\t', '\\' (обычный вариант)
- '\40', '\62' – восьмеричный код
- '\u0053' – юникод

Константы

□ Символьные

- 'a', '?', ' ', '\n', '\t', '\\' (обычный вариант)
- '\40', '\62' – восьмеричный код
- '\u0053' – юникод

□ Строковые

- "Hello, world\n"
- "Сложение " + "строк"
- Нет строковых шаблонов: ~~"Hello \$name"~~

Операции

□ Арифметические: + - * / % ++ --

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !
- Сравнения: > < >= <= == !=

Операции сравнения: Java vs Kotlin

- Kotlin
 - `a == b` сравнение значений
 - `a === b` сравнение ссылок
- Java
 - `a.equals(b)` сравнение значений
 - `a == b` сравнение ссылок

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !
- Сравнения: > < >= <= == !=
- Побитовые: ~ & | ^

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !
- Сравнения: > < >= <= == !=
- Побитовые: ~ & | ^
- Сдвиговые: << >> >>>

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !
- Сравнения: > < >= <= == !=
- Побитовые: ~ & | ^
- Сдвиговые: << >> >>>
- Присваивания: = += -= *= /= %= &= |=
^= <<= >>= >>>=

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !
- Сравнения: > < >= <= == !=
- Побитовые: ~ & | ^
- Сдвиговые: << >> >>>
- Присваивания: = += -= *= /= %= &= |=
^= <<= >>= >>>=
- Условная: **a > b ? a : b**

Операции

- Арифметические: + - * / % ++ --
- Логические: & && | || ^ !
- Сравнения: > < >= <= == !=
- Побитовые: ~ & | ^
- Сдвиговые: << >> >>>
- Присваивания: = += -= *= /= %= &= |=
^= <<= >>= >>>=
- Условная: **a > b ? a : b**
- Приведения типа: **int a = (int)2.5**

Ветвления

- `if ... else ...` -- как в Kotlin
- НО! Нельзя использовать как выражение
 - вместо этого есть `a ? b : c`

Ветвления

- switch ... case ...
- Как when в Kotlin, но меньше возможностей

```
switch (someInt) {  
    case 1:  
        ...  
        break;  
    default:  
        ...  
        break;  
}
```

Ключи в switch

- Только
 - Целые числа
 - Символы
 - Элементы перечисления (enum)
 - Строки (только в JDK 7+)

Циклы

- `while (...) { ... }`
 - как в Котлине

Циклы

- `while (...) { ... }`
 - как в Котлине
- `do { ... } while (...)`
 - ~ как в Котлине

Циклы

- `while (...) { ... }`
 - как в Котлине
- `do { ... } while (...)`
 - ~ как в Котлине
- `for (int i = 0; i < 10; i++) { ... }`

Циклы

- `while (...) { ... }`
 - как в Котлине
- `do { ... } while (...)`
 - ~ как в Котлине
- `for (int i = 0; i < 10; i++) { ... }`
- `for (String element: listOfStrings) { ... }`

Циклы

- `while (...) { ... }`
 - как в Kotlinе
- `do { ... } while (...)`
 - ~ как в Kotlinе
- `for (int i = 0; i < 10; i++) { ... }`
- `for (String element: listOfStrings) { ... }`
- `break, continue`

Строки

- ❑ Тип String
- ❑ В основном как в Котлине
- ❑ Сложение +
- ❑ Сравнение equals (!)

Массивы

- ❑ Тип массива в Java обозначается как T[], где T – базовый тип
- ❑ например, int[], float[], double[], String[]
- ❑ Создается массив следующим образом:

```
int[] arr = new int[10]; // с нулями  
int[] arr = null; // нулевая ссылка  
// начальные значения заданы  
int[] arr2 = new int[] { 2, 3, 4 };
```

Итоги

- Рассмотрена
 - простейшая программа
 - примитивные типы
 - константы и операции
 - основные конструкции
- Далее
 - проектирование классов