Гуляев Г.М.
Современные технологии программирования

Лекция 1. Проблемы разработки ПО

Центр компетенции СПО Алтайского края

Языки программирования

- **Компьютер - это реальное или виртуальное устройство с архитектурой фон Неймана:**
  - данные и команды хранятся в общей памяти
  - работа компьютера сводится к арифметическим вычислениям
  - команды выполняются последовательно
  - выполнение команды приводит к изменению памяти
- **Сегодня существует более 2000 языков программирования, но большинство из них мертвые - реально используются только около пары сотен**
  - анализ 45 млн строк кода выявил 3 лидера: **C**, **C++** и **Java**
- **Тем не менее новые языки продолжают появляться каждый год:**
  - **Clojure** (2007), **Go** (2009), **Dart** (2011), **Ceylon** (2011), **Chapel** (2013), ...

Классификация языков программирования

	C	C++	C#	Java	Python	Delphi	Ruby	PHP	Smalltalk	Lisp
Императивные	+	+	+	+	+	+	+	+	+	+
Декларативные	-	-	-	-	+	-	+	-	+	+
Функциональные	-	-	-	-	+	-	+	-	+	+
Объектные	-	+	+	+	+	+	+	+	+	+
Логические	-	-	-	-	-	-	-	-	+	+
Распределенные	-	-	-	-	-	-	-	-	+	+

- **Группа языка определяет парадигму программирования (стиль, способ). Многие современные языки (python, ruby, ...) позволяют использовать различные парадигмы**

Типизация в языках программирования

- **Статическая типизация** (Pascal, C, C++, C#, Java,...)
- **Динамическая типизация** (Python, PHP, Ruby,...)
- **Строгая типизация** (Pascal, Java, Python, C#,...)
- **Слабая типизация** (C, C++, PHP, Perl,...)

- Язык формирует способ мышления. Поэтому, существует мнение, что для первоначального обучения следует избирать языки со строгой статической типизацией данных: Pascal, Java, C#, …
- Принцип разумной свободы. Пример языка PL/1 от фирмы IBM, который предоставлял слишком много свободы, был не понят и забыт

Проблема сложности кода

- **Для больших систем тексты программ могут содержать суммарно сотни тысяч или даже миллионы строк кода**
  - **ядро линукс 3.3 > 15 млн строк на языках C,C++**
- **Человек не может воспринимать и контролировать много параметров одновременно**
- **Решения:**
  - **код пишется не столько для компилятора, сколько для человека, который будет поддерживать программу**
  - **принцип «разделяй и властвуй» (структурирование, ООП)**
  - **единица кода должна выполнять только одну задачу**
  - **не допускать дублирования частей кода**
  - **поддержка разработчиками единого стиля оформления кода**

Требования к оформлению кода на java

- Отступы для блоков кода (рекомендуется 4 пробела)
- Имена пакетов только из строчных букв
- Имена классов начинать с большой буквы (MyClass)
- Имена полей и методов с маленькой (getNewWord())
- В составных именах каждое новое подслово с заглавной
- Имена констант состоят из больших букв и _ (ARRAY_MIN_VALUE)
- В именах использовать правильные английские слова
- Имя должно отражать смысл действия или состояния
- Слишком большой класс или метод - ошибка проектирования
- Слишком много классов в пакете - ошибка проектирования
- Слишком много классов в одном файле - ошибка проектирования (как правило, один класс = один файл)

Философия Unix

- **Дуг Макилрой:**
  - **программа должна делать что-то одно и делать это хорошо**
  - **программы должны уметь работать вместе**
  - **программы должны поддерживать текстовые потоки**
- **Эрик Рэймонд:**
  - **правило модульности**
  - **правило ясности**
  - **правило композиции**
  - **правило разделения**
  - **правило простоты**
  - **правило экономности**
  - **правило прозрачности**
  - **правило надёжности**
  - **...**

Проблема библиотек и интерфейсов

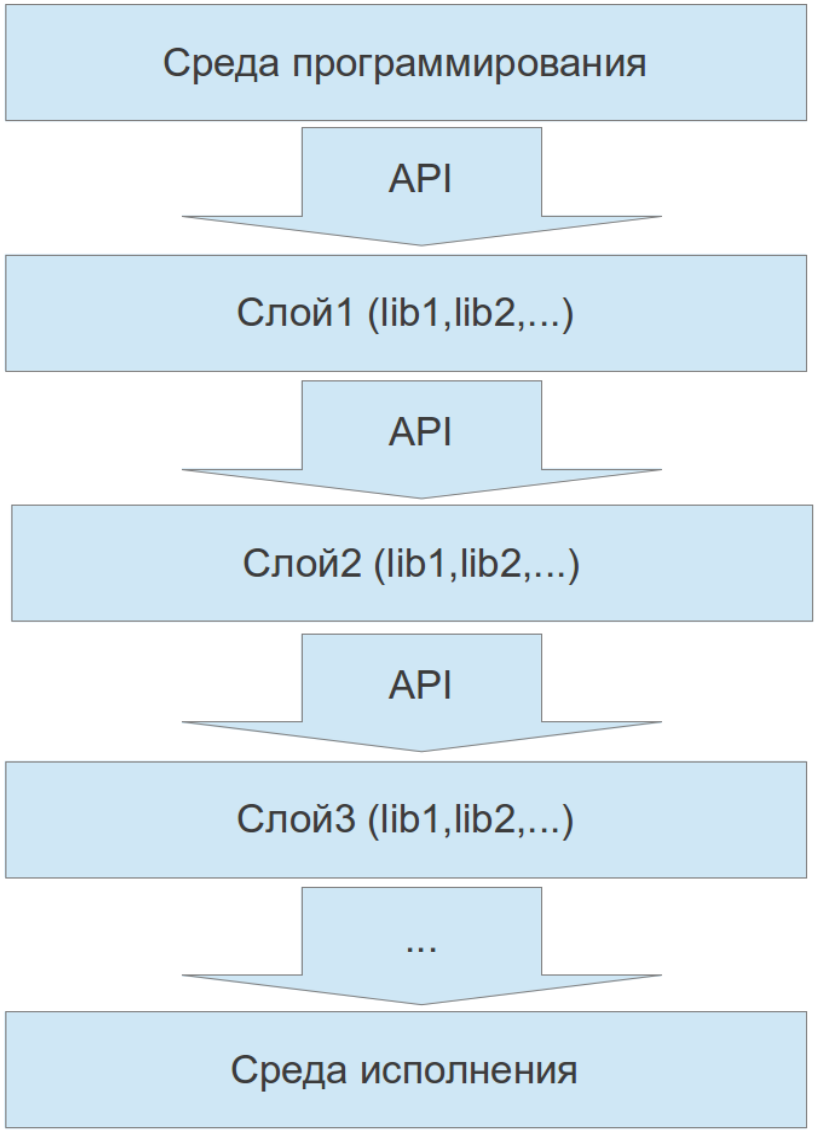
- **Потеря свободы**
- **Потеря контроля**
- **Порождение лишних сущностей (знаний)**
- **Процесс программирования превращается в гадание**

- **Решения:**

- стараться не использовать лишних слоев

- использовать библиотеки с логичным интерфейсом

- использовать библиотеки, дающие больше свободы



Попытки решения проблем (тенденции)

- Понятие «хороший код», «чистый код»
- Рефакторинг - изменения кода без изменения функциональности
- Паттерны (шаблоны) проектирования
- Разработка через тестирование (**TDD**)
- Новые технологии коллективной разработки (**agile**, **scrum**,...)
- Внедрение в языки программирования элементов для функционального стиля программирования
- Создание новых языков программирования

Языки программирования для jvm

- **Для виртуальной машины java (jvm) сегодня можно писать программы не только на языке java:**
  - **Groovy** (ООП, типизация: динамическая, возможен синтаксис java)
  - **JRuby** (ООП, типизация: динамическая, ruby для jvm)
  - **Jython** (ООП+функц, типизация: динамическая, python для jvm)
  - **Clojure** (Функц, типизация: динамическая, диалект Lisp)
  - **Scala** (ООП+функц+конкурент, тип: статическая, возможна java)
  - **Kotlin** (универс, тип: статическая, компил-ся в jvm и JavaScript)
  - **Rhino**, **Ceylon**, **Fantom** - другие языки
- **Для больших проектов: статическая типизация, ООП, функц. программирование, интеграция с java**
- **Scala - компилятор для jvm разработан в 2003 году группой под руководством Мартина Одерски**
- **Объектно-функциональная модель (scala) - это интеграция функционального и объектного подхода**

Особенности использования scala

Преимущества

1. Все преимущества **java** и **jvm** (статическая типизация, мультиплатформенность, быстродействие, библиотеки **java**, …)
2. Логичный синтаксис, отсутствия дублирования кода
3. Функциональное программирование (ФП) упрощает код
4. Неизменяемые объекты - более безопасный код
5. Конкурентное программирование (**actors**)
6. Интерактивный интерпретатор (**REPL**)

Недостатки

1. Много свободы (много способов решения одной и той же задачи)
2. Высокий порог вхождения (требуется понимание **java** и ФП)
3. Слабая поддержка **IDE** (медленная компиляция, нет отладки)

Решения

- Выработать свой стиль программирования
- Использовать **IDE** только для ввода кода
- Использовать для сборки другие средства (**sbt**)

Сравнение java и scala на примерах

- f(n) = сумме всех натуральных чисел меньших n, делящихся на 3 или на 5. Написать программу для вычисления n → f(n)

- Решение на java:

```
int f (int n) {
    int s = 0;
    for (int i=1; i<n; i++) {
        if ((i%3==0)||(i%5==0)) s+=i;
    }
   return s;
}
```

- Решение на scala:

```
def f(n: Int) = {
  var s = 0
   for (i <- 1 to n-1) {
     if ((i%3==0)||(i%5==0)) s+=i
   }
   s
}
```

- Решение на scala (функциональный стиль):

```
def f(n: Int) = (1 to n-1).filter(x => (x%3==0) || (x%5==0)).sum
```

Сравнение java и scala на примерах

- **Текстовый файл abit.txt содержит большой список абитуриентов 2013 года АлтГТУ в формате:**

  фамилия имя отчество математика русский физика сумма документы

  Абакумов Антон Юрьевич 77 73 92 242 копия

  ...

- **Консольная программа должна дать пользователю возможность выбора и вывести 10 абитуриентов с лучшим баллом по математике или по русскому или по физике или по сумме баллов**
- **Результаты:**

  Scala: 1 класс и 1 объект, 29 строк, 949 байт

  Java: 3 класса, 112 строк, 2876 байт

Отличия синтаксиса scala от java

- Необязательность точки с запятой в конце строк
- Имя объекта с методом **main** может не совпадать с именем файла
- Необязательность задания типа переменной, если он может быть вычислен компилятором из контекста
- Необязательность директивы **return** для возвращаемого значения
- Необязательность круглых скобок у метода без аргументов (**sum = sum()**)
- Возможность объявить не класс, а объект устраняет директивы **static**
- Типы объявляются после переменной или функции через двоеточие (**n: Int**)
- Элементы массива(списка) выбираются при помощи круглых, а не квадратных скобок (**args(0)**)

Отличия синтаксиса scala от java

- Типизация массива(списка) задается квадратными, а не угловыми скобками (**Array[String]**)
- Функции (методы) определяются директивой **def**, изменяемые переменные **var**, неизменяемые **val** (тем самым отпадает необходимость в директиве **final**)
- При задании функций между определением и реализацией используется знак **=** как и при задании переменных
- Отсутствуют примитивные типы, например, числа являются объектами
- Сравнение **==** объектов осуществляется по их внутреннему содержанию (равносильно **equals** в java)
- Операторы являются методами (**1.+(2)** равносильно **1 + 2**)
- Имя пакета не обязательно совпадает с именем каталога

Ресурсы для самостоятельного изучения

- **Почему Скала:**
  http://fprog.ru/2010/issue6/vlad-patryshev-why-scala/

- **Java 8 vs. Scala: сравнение возможностей:**
  http://blogerator.ru/page/java-8-vs-scala-sravnenie

- **Scala как расширенная Java или Java++:**
  http://habrahabr.ru/post/152815/

- **Есть ли жизнь после Java:**
  http://www.youtube.com/watch?v=28DDcwW_W8s

Спасибо за внимание !

www.altailand.ru