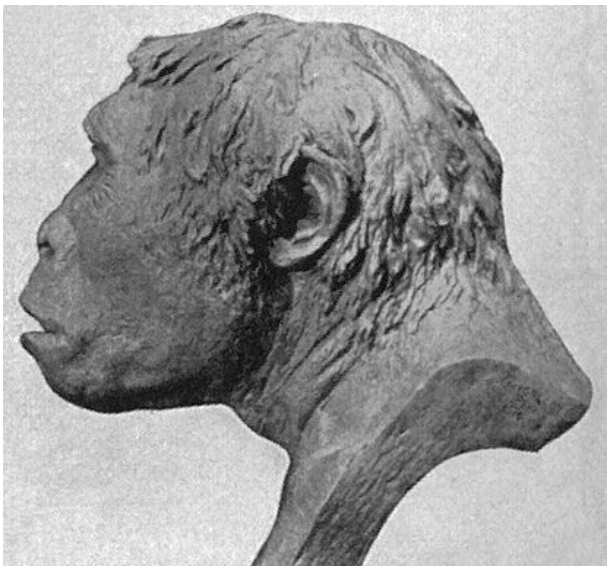
Day 00 - Piscine Java

Структуры управления и массивы

Резюме: Сегодня вы узнаете принципы решения как тривиальных, так и серьезных бизнес-задач с использованием базовых конструкций Java

Contents

Chapter I
Preamble



Java Man, or *Homo erectus erectus*

Chapter II
General Rules

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- Сейчас для вас существует только одна версия Java - 1.8. Убедитесь, что на вашем компьютере установлен компилятор и интерпретатор данной версии.
- Не запрещено использовать IDE для написания исходного кода и его отладки.
- Код чаще читается, чем пишется. Внимательно изучите представленный документ с правилами оформления кода. В каждом задании обязательно придерживайтесь общепринятых стандартов Oracle - https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html
- Комментарии в исходном коде вашего решения запрещены. Они мешают восприятию.
- Pay attention to the permissions of your files and directories.
- To be assessed your solution must be in your GIT repository.
- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.
- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google. И еще, для любых ваших вопросов существует ответ на Stackoverflow. Научитесь правильно их задавать.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the Force be with you!
- Не откладывайте на завтра то, что можно было сделать вчера ;)

Chapter III
Rules of the day

- Во всех заданиях дня запрещено использование пользовательских методов и классов, за исключением пользовательских статических функций и процедур в основном файле-классе решения.
- Во всех заданиях приведен список РАЗРЕШЕННЫХ конструкций языка для данной задачи.
- Во всех заданиях разрешено использование System::exit
- Во всех заданиях приведен пример работы приложения. Реализованное решение должно полностью повторять указанный пример вывода для текущих входных данных.
- Для наглядности в примерах заданий вводимые пользователем данные начинаются со стрелки ->. Не учитывать данные символы при реализации решения!

PS. Некоторые задания требуют нестандартного подхода ввиду указанных ограничений. Данные ограничения научат вас находить решения для автоматизации реальных бизнес-процессов.

Chapter IV
Exercise 00 - Sum of Digits

Exercise 00: Sum of Digits	
Turn-in directory	ex00
Files to turn-in	Program.java
Разрешения	
Типы	Примитивные типы
Операторы	Стандартные операции примитивных типов

Java - язык программирования со строгой типизацией. Фундаментальные типы данных (логический, символьный, целочисленный, вещественный) представлены в Java восемью примитивными типами - `boolean, char, byte, short, int, long, float, double.`

Реализуйте работу с целочисленным типом.

Для шестизначного числа типа int (значение числа задается непосредственно в коде явной инициализацией переменной number), посчитать сумму цифр этого числа.

Пример работы программы для числа `479598`:

```
$ java Program
  42
```

Chapter V
Exercise 01 - Really Prime Number

Exercise 01: Really Prime Number	
Turn-in directory	ex01
Files to turn-in	Program.java
Разрешено	
Ввод/Вывод	System.out, System.err, Scanner(System.in)
Типы	Примитивные типы
Операторы	Стандартные операции примитивных типов, условия, циклы

Согласно теореме Бёма-Якопини, любой алгоритм можно представить в виде трех структур управления - последовательной, условной и циклической.

Используя эти структуры управления в Java, вам необходимо определить - является ли входное число простым. Простым называется число, которое не имеет делителей, отличных от самого числа и единицы.

На вход программа принимает число, введенное в процессе работы программы с клавиатуры, и выводит на экран результат проверки на “простоту”. Также программа должна выводить количество шагов (итераций), за которое ей удалось выполнить эту проверку. В данной задаче итерация - одна операция сравнения.

Для отрицательных чисел, нуля и единицы следует вывести сообщение `IllegalArgument` и завершить выполнение программы с кодом **-1**.

Примеры работы программы:

```
$ java Program
-> 169
   false 12

$ java Program
-> 113
   true 10

$ java Program
-> 42
   false 1

$ java Program
-> -100
   Illegal Argument
```

Chapter VI
Exercise 02 - Endless Sequence (or not?)

Exercise 02: Endless Sequence (or not?)	
Turn-in directory	ex02
Files to turn-in	Program.java
Разрешено	
Ввод/Вывод	System.out, System.err, Scanner(System.in)
Типы	Примитивные типы
Операторы	Стандартные операции примитивных типов, условия, циклы

Сегодня вы - Google.
Ваша задача - посчитать, сколько запросов, касающихся приготовления кофе, к нашей поисковой системе делают пользователи в произвольный момент времени. Понятно, что последовательность запросов бесконечна. Хранить такие запросы, а потом считать количество - невозможно.

Но есть решение - обрабатывать данные “в потоке”. Зачем тратить ресурсы на все запросы, если нас интересует только определенная характеристика этой последовательности запросов? Пусть каждый запрос - это любое отличное от нуля и единицы натуральное число. Запрос относится к приготовлению кофе тогда, когда сумма цифр этого числа-запроса - простое число.

Таким образом, необходимо реализовать программу, которая для заданного набора чисел посчитает количество элементов, у которых сумма цифр - простое число.
Для простоты будем считать, что у этой потенциально бесконечной последовательности запросов все же есть предел - последним элементом последовательности станет число **42**.

В этом задании гарантируется полная корректность входных данных.

Пример работы программы:

```
$ java Program
-> 198131
-> 12901212
-> 11122
-> 42
   Count of coffee-request - 2
```

Chapter VII
Exercise 03 - A little bit of statistics

Exercise 03: A little bit of statistics	
Turn-in directory	ex03
Files to turn-in	Program.java
Разрешено	
Ввод/Вывод	System.out, System.err, Scanner(System.in)
Типы	Примитивные типы, String
Операторы	Стандартные операции примитивных типов, условия, циклы
Методы	String::equals

При разработке корпоративных систем часто возникают задачи по сбору различной статистики. При этом заказчик всегда хочет, чтобы такая аналитика была наглядной. Кому интересны голые цифры?

Образовательные организации и онлайн-школы часто выступают такими заказчиками. И сейчас вам необходимо реализовать функционал, демонстрирующий прогресс учеников. Заказчик хочет видеть график, в котором мы покажем, как меняется успеваемость ученика в течение нескольких недель.

Эту успеваемость заказчик оценивает как минимальный балл, полученный за 5 контрольных в течение каждой недели. За любую контрольную можно получить оценку от 1 до 9 баллов.

Максимально возможное количество недель, по которым следует проводить анализ - 18. После того, как программа получила информацию по каждой из недель, она выведет в консоль график, отражающий значения минимального балла за конкретную неделю.

Придерживаемся традиции, 42 - признак окончания ввода.

Гарантируется точное количество контрольных в неделю - 5.

Не гарантируется правильный порядок ввода недель, таким образом, Week 1 может быть введен позже Week 2. В случае нарушения порядка необходимо вывести сообщение `IllegalArgument` и завершить выполнение программы с кодом **-1.**

Примечание:

1. Существует большое количество вариантов хранения информации, массивы - не единственный из них. Примените другой метод для хранения данных о контрольных без использования массивов.
2. Конкатенация строк часто приводит к непредвиденным последствиям в работе программы. В случае, когда в цикле происходит многократное повторение операции конкатенации для одной переменной, может существенно замедлиться работа приложения. Следовательно, использовать конкатенацию строк внутри цикла для формирования результата - недопустимо.

Пример работы программы:

```
$ java Program
-> Week 1
-> 4 5 2 4 2
-> Week 2
-> 7 7 7 7 6
-> Week 3
-> 4 3 4 9 8
-> Week 4
-> 9 9 4 6 7
-> 42
Week 1 ==>
Week 2 ======>
Week 3 ===>
Week 4 ====>
```

Chapter VIII
Exercise 04 - A bit more of statistics

Exercise 04: A bit more of statistics	
Turn-in directory	ex04
Files to turn-in	Program.java
Разрешено	
Ввод/Вывод	System.out, System.err, Scanner(System.in)
Типы	Примитивные типы, String, массивы
Операторы	Стандартные операции примитивных типов, условия, циклы
Методы	String::equals, String::toCharArray

Вы знали, что с помощью частотного анализа можно дешифровать “плохо зашифрованные” тексты?

См. https://en.wikipedia.org/wiki/Frequency_analysis

Почувствуйте себя хакерами и реализуйте программу для определения частоты появления того или иного символа в тексте.

Мы любим наглядность, поэтому результатом работы программы станет график-гистограмма. В данном графике вы покажете 10 наиболее часто встречаемых символов, от меньшего к большему.

В случае, если буквы встречаются одинаковое количество раз, необходимо упорядочить их в лексикографическом порядке.

Каждый символ может присутствовать в тексте огромное количество раз, поэтому график необходимо масштабировать. Так, максимальная высота столбца выводимого графика - 10, минимальная - 0.

На вход программе подается строка, содержащая только один символ `\n` в конце (следовательно, на вход может быть подана одна длинная строка).

Предполагается, что любой входной символ может быть помещен в переменной типа char (Unicode BMP, например, буква `Ы` имеет код 1067, максимальное значение кода - 65,535).

Частота появления одного символа не превышает 999.

Примечание: данная задача должна быть решена без многократного прохождения по исходному тексту (сортировка, удаление повторений). Поскольку применение данных методов существенно замедлит работу приложения. Примените другие методы обработки информации.

Пример работы программы:

```
 $ java Program

->
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASSSSSSSSSSSSSSSSSSSSSSSDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDWEWWKFKKDKKDSKAKLSLDKSKALLLLLLLLLLRTRTETWTWWWWWWWWWWOOOOOOO42

  36
   #  35
   #   #
   #   #  27
   #   #   #
   #   #   #
   #   #   #
   #   #   #  14  12
   #   #   #   #   #   9
   #   #   #   #   #   #   7   4
   #   #   #   #   #   #   #   #   2   2
   D   A   S   W   L   K   O   T   E   R
```

Chapter IX
Exercise 05 - Schedule

Exercise 05: Schedule	
Turn-in directory	ex05
Files to turn-in	Program.java
Разрешено	
Ввод/Вывод	System.out, System.err, Scanner(System.in)
Типы	Примитивные типы, String, массивы
Операторы	Стандартные операции примитивных типов, условия, циклы
Методы	String::equals, String::toCharArray

Не успели вы стать успешным хакером, как к вам вернулся заказчик. На этот раз ему необходимо получить возможность вести расписание занятий в своем образовательном учреждении. Заказчик открывает свою школу в сентябре 2020-го. Поэтому, вы должны реализовать MVP-версию проекта только для этого месяца:)

Вам следует обеспечить возможность формировать список учеников, а также указывать, в какое время и в какие дни недели проводятся занятия. Занятия могут проводится в любой из семи дней недели в период с 01 pm до 06 pm. В одном дне может быть более одного занятия, тем не менее, общее количество занятий в неделю не может превышать 10.

Также 10 - максимальное количество учеников в расписании. Максимальная длина имени ученика равна ... 10 (не содержит пробелов).

Также следует предусмотреть возможность отмечать посещаемость ученика. Для этого напротив каждого имени ученика следует указать время и дату занятия, а также статус посещения (HERE, NOT_HERE). Не обязательно отмечать посещения для всех уроков месяца.

Таким образом, приложение имеет следующий жизненный цикл:

1. Формирование списка учеников
2. Внесение расписания - каждый урок (время, день недели) вносится в отдельной строке
3. Внесение посещений
4. Вывод расписания в табличном виде со статусами посещения.

Каждый этап работы приложения разделяется символом “.” Гарантируется полная корректность входных данных, но не гарантируется последовательный порядок уроков при внесении расписания.

Пример работы приложения:

```
John
Mike
.
2 MO
4 WE
.
Mike 2 28 NOT_HERE
John 4 9 HERE
Mike 4 9 HERE
.
          4:00 WE  2|2:00 MO  7|4:00 WE  9|2:00 MO 14|4:00 WE 16|2:00 MO 21|4:00 WE 23|2:00 MO 28|4:00 WE 30|
      John          |          |         1|          |          |          |          |          |          |
      Mike          |          |         1|          |          |          |          |         -1|          |
```

CHECKLIST:
https://docs.google.com/document/d/1luj5mePaPlYGgrkrRGkTza7mwmcbhvM9uYfifhcjBIE/edit?usp=sharing