

Модуль 00 - Бассейн Java

Структуры управления и массивы

Описание: Сегодня вы познакомитесь с основами решения как тривиальных, так и более сложных бизнес-задач с использованием базовых конструкций языка Java.

Содержание

I Предисловие

II Общие правила

III Правила дня

IV Упражнение 00. Сумма цифр

V Упражнение 01. Действительно простое число

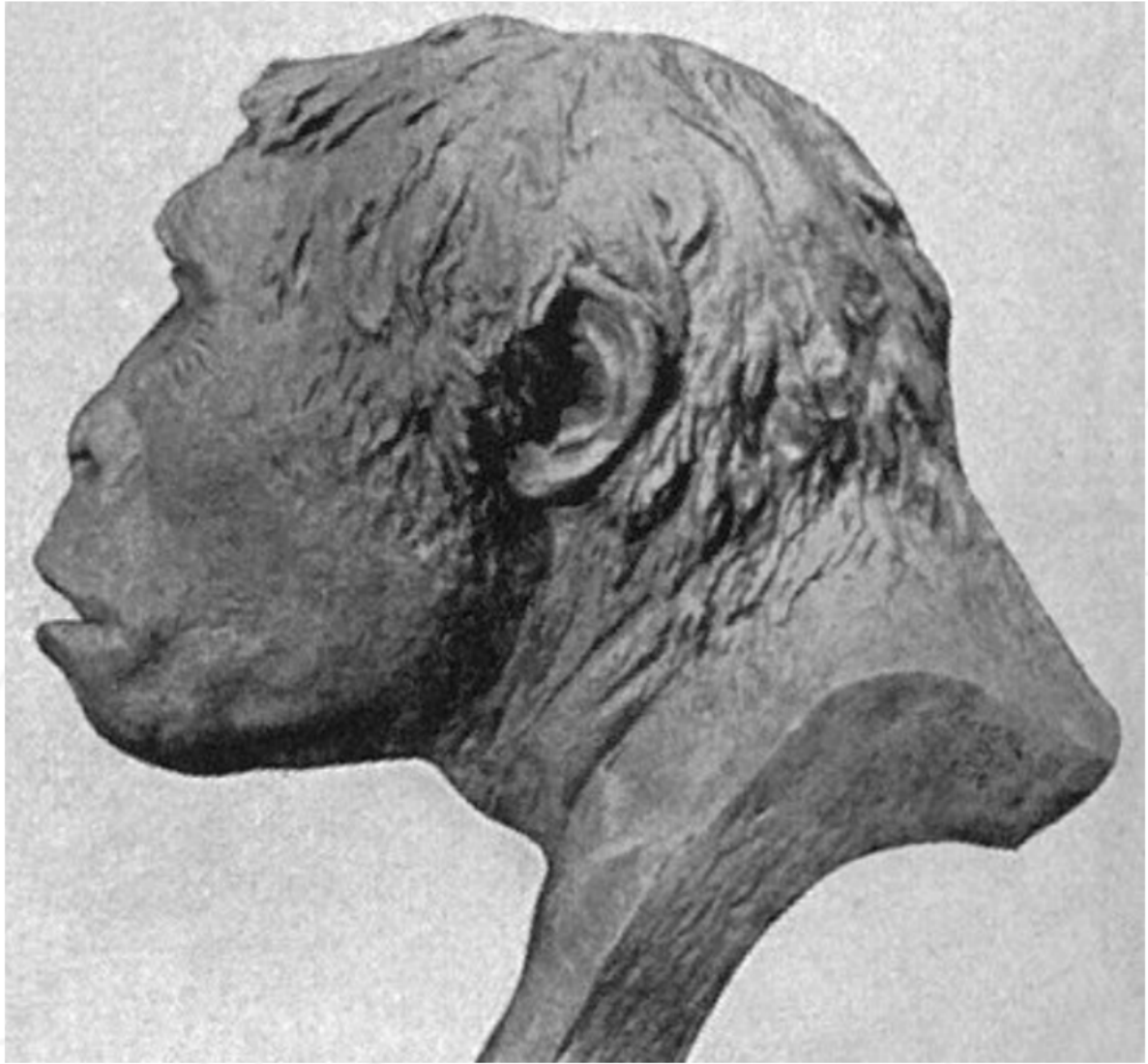
VI Упражнение 02: Бесконечная последовательность (или нет?)

VII Упражнение 03. Немного статистики

VIII Упражнение 04. Еще немного статистики

IX Упражнение 05: Расписание

Глава I
Предисловие



Java Man, or *Homo erectus erectus*

Глава II

Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить свой раствор.
- Теперь у вас есть только одна версия Java, 1.8. Убедитесь, что на вашем компьютере установлены компилятор и интерпретатор этой версии.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код читается чаще, чем пишется. Внимательно прочитайте документ, в котором приведены правила форматирования кода. При выполнении каждой задачи убедитесь, что вы следуете общепринятым стандартам Oracle.
- Комментарии в исходном коде вашего решения запрещены. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Для оценки ваше решение должно находиться в вашем репозитории GIT.
- Ваши решения будут оценены вашими товарищами.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить ваш .gitignore, чтобы избежать несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещается отображать предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- Есть вопрос? Спросите у соседа справа. В противном случае попробуйте с соседом слева.
- Ваш справочник: друзья/интернет/гугл. И еще кое-что. На Stackoverflow есть ответ на любой вопрос, который у вас может возникнуть. Научитесь правильно задавать вопросы.
- Внимательно прочитайте примеры. Они могут требовать вещи, которые иначе не указаны в теме.
- Используйте "System.out" для вывода

Глава III

Правила дня


- Определенные пользователем методы и классы запрещены для всех задач дня, кроме определяемых пользователем статических функций и процедур в основном файле класса решения.
- Все задачи содержат список РАЗРЕШЕННЫХ языковых конструкций для конкретной задачи.
- `System::exit` можно использовать для всех задач.
- Все задания содержат пример работы приложения. Реализованное решение должно быть идентично заданному выходному примеру для текущих входных данных.
- Для наглядности данные, введенные пользователем в примерах задач, отмечены стрелкой (->). Не учитывайте эти стрелки при реализации решения!

P.S. Некоторые задачи требуют нетривиального подхода из-за вышеупомянутых ограничений.

Эти ограничения научат вас находить решения для автоматизации реальных бизнес-процессов.

Глава IV

Упражнение 00: Сумма цифр

	Exercise 00
	Sum of Digits
	Turn-in directory : <i>ex00/</i>
	Files to turn in : Program.java
	Allowed functions : Input/Output : System.out Types : Primitive types Operators : Standard operations of primitive types

Java — строго типизированный язык программирования. Основные типы данных (логическое, символьное, целое число, число с плавающей запятой) представлены в Java восемью примитивными типами: логическое, символьное, байтовое, короткое, целое, длинное, плавающее, двойное.

Работа с целочисленным типом.


- Вычислить сумму цифр шестизначного числа `int` (значение числа задается непосредственно в коде путем явной инициализации переменной `number`).

Пример работы программы для номера 479598:

```
$ java Program
42
```

Глава V

Упражнение 01: Действительно простое число

	Exercise 01
Really Prime Number	
Turn-in directory : <i>ex01/</i>	
Files to turn in : Program.java	
Allowed functions : Input/Output : System.out, System.err, Scanner(System.in) Types : Primitive types, Operators : Standard operations of primitive types, conditions, loops	

Согласно теореме Бёма-Якопини любой алгоритм можно записать с помощью трех утверждений:

последовательность, выбор и итерация.

- Используя эти операторы в Java, вам нужно определить, является ли введенное число простым. Простое число — это число, у которого нет других делителей, кроме самого числа и 1.
- Программа принимает на вход число, введенное с клавиатуры, и отображает результат проверки, является ли это число простым. Кроме того, программа должна выводить количество шагов (итераций), необходимых для выполнения проверки. В этой задаче итерация — это одна операция сравнения.
- Для отрицательных чисел, 0 и 1, вывести сообщение `IllegalArgumentException` и закрыть программу с кодом -1.

Пример работы программы:

```
$ java Program
-> 169
false


$ java Program
-> 113
true

$ java Program
-> 42
false

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

Глава VI

Упражнение 02: Бесконечная последовательность (или нет?)

	Exercise 02
Endless Sequence (or not?)	
Turn-in directory : <i>ex02/</i>	
Files to turn in : Program.java	
Allowed functions : Input/Output : System.out, System.err, Scanner(System.in) Types : Primitive types, Operators : Standard operations of primitive types, conditions, loops	

Сегодня вы Google.

Вам необходимо подсчитать запросы, связанные с приготовлением кофе, которые делают пользователи нашей поисковой системы в определенный момент. Понятно, что последовательность поисковых запросов бесконечна. Невозможно сохранить эти запросы и подсчитать их позже.

Но есть решение — обрабатывать поток запросов. Почему мы должны тратить наши ресурсы на все запросы, если нас интересует только конкретная функция этой последовательности запросов? Предположим, что каждый запрос представляет собой любое натуральное число, отличное от 0 и 1. Запрос относится к приготовлению кофе только в том случае, если сумма цифр числа (запроса) является простым числом.

Итак, нам нужно реализовать программу, которая будет подсчитывать количество элементов для заданного набора чисел, сумма цифр которого является простым числом.

Для простоты предположим, что эта потенциально бесконечная последовательность запросов все еще ограничена, а последний элемент последовательности имеет номер 42.


- Эта задача гарантирует, что входные данные абсолютно корректны.

Пример работы программы:

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


Глава VII

Упражнение 03. Немного статистики

	Exercise 03
A Little Bit of Statistics	
Turn-in directory : <i>ex03/</i>	
Files to turn in : Program.java	
Allowed functions : Input/Output : System.out, System.err, Scanner(System.in) Types : Primitive types, String Operators : Standard operations of primitive types, conditions, loops Methods : String::equals	

При разработке корпоративных систем часто возникает необходимость сбора различного рода статистики.

А заказчику всегда хочется, чтобы такая аналитика была наглядна. Кому нужны холодные, сухие фигуры?

Образовательные организации и онлайн-школы часто относятся к этому типу клиентов. Теперь вам нужно реализовать функциональность для визуализации прогресса учащихся. Клиент хочет видеть диаграмму, показывающую изменения успеваемости учащегося за несколько недель. Заказчик оценивает этот прогресс как минимальную оценку за пять тестов в неделю. Каждый тест может быть оценен от 1 до 9. Максимальное количество недель для анализа — 18. После того, как программа получит информацию за каждую неделю, она отобразит на консоли график, показывающий минимальные оценки за конкретную неделю.

И мы продолжаем считать, что 42 — это предел входных данных.

Точное гарантированное количество тестов в неделю — 5.

Однако порядок еженедельного ввода данных не гарантируется, поэтому данные первой недели можно вводить после данных второй недели. При неправильном порядке ввода данных будет выведено сообщение `IllegalArgumentException` и программа будет закрыта с кодом -1.

Примечание:

- Существует множество вариантов хранения информации, и массивы — лишь один из них.

Примените другой способ хранения данных о тестах учащихся без использования массивов.

- Конкатенация строк часто приводит к неожиданному поведению программы. Если в цикле выполняется много итераций операции конкатенации для одной переменной, приложение может значительно замедлиться. Вот почему мы не должны использовать конкатенацию строк внутри цикла для получения результата.

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 ====>
```

Глава VIII

Упражнение 04. Еще немного статистики

	Exercise 04
A Bit More of Statistics	
Turn-in directory : <i>ex04/</i>	
Files to turn in : Program.java	
Allowed functions : Input/Output : System.out, System.err, Scanner(System.in) Types : Primitive types, String, arrays Operators : Standard operations of primitive types, conditions, loops Methods : String::equals, String::toCharArray, String::length	

Знаете ли вы, что можно использовать частотный анализ для расшифровки плохо зашифрованных текстов?

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

Почувствуйте себя хакером и напишите программу для подсчета вхождения символов в текст.

Нам нравится визуальная ясность. Вот почему программа будет отображать результаты в виде гистограммы.

Эта диаграмма покажет 10 наиболее часто встречающихся символов в порядке убывания.

- Если символы встречаются одинаковое количество раз, они должны быть отсортированы в лексикографическом порядке.
- Каждый символ может встречаться в тексте большое количество раз. По этой причине диаграмма должна быть масштабируемой. Максимальная высота отображаемой диаграммы равна 10, а минимальная — 0.
- Входными данными для программы является строка с одним символом "\n" в конце (таким образом, в качестве входных данных может использоваться одна длинная строка).
- Предполагается, что каждый вводимый символ может содержаться в переменной char (Unicode BMP; например, код буквы «S» — 0053, максимальное значение кода — 65535).
- Максимальное количество вхождений символов — 999.

Примечание: эту проблему необходимо решать без многократных итераций по исходному тексту (сортировки и удаления повторов), т. к. эти способы значительно замедлят работу приложения. Используйте другие методы обработки информации.

Пример работы программы:

```
$ java Program
->
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASSSSSSSSSSSSSSSSSSSSSSDDDDDD
DDDDDDDDDD
DDDDDDDDDDDDDDDDDDWEWWKFKKDKDSKAKLSLDKSKALLLLLLLLLLLRTTETWTWW
WWWWW
```


```

WWWOOOOOOO42
36
# 35
# #
# # 27
# # #
# # #
# # #
# # # 14 12
# # # # 9
# # # # # 7 4
# # # # # # 2 2
DASWLKOTER

```

Глава IX

Упражнение 05: Расписание

	Exercise 05
Schedule	
Turn-in directory : <i>ex05/</i>	
Files to turn in : Program.Java	
Allowed functions : Input/Output : System.out, System.err, Scanner(System.in) Types : Primitive types, String, arrays Operators : Standard operations of primitive types, conditions, loops Methods : String::equals, String::toCharArray, String::length	

Вы только что стали отличным хакером, но ваш клиент вернулся к вам с другим заданием. На этот раз им нужно иметь возможность вести расписание занятий в своем учебном заведении. Заказчик открывает школу в сентябре 2020 года. Значит, вам нужно реализовать MVP-версию проекта только в этом месяце.

Вы должны иметь возможность создать список учеников и указать время и дни недели для занятий.

Занятия могут проходить в любой день недели с 13:00 до 18:00. В один день можно проводить несколько занятий. Однако общее количество занятий в неделю не может превышать 10.

Максимальное количество студентов в расписании также 10. Максимальная длина имени студента 10 (без пробелов).

Вы также должны предоставить возможность записывать посещаемость студента. Для этого рядом с именем каждого студента необходимо указать время и дату занятий, а также статус посещаемости (ЗДЕСЬ, НЕ_ЗДЕСЬ). Вам не нужно записывать посещаемость всех занятий за месяц.

Таким образом, жизненный цикл приложения выглядит следующим образом:

- Создание списка студентов
- Заполнение расписания — каждое занятие (время, день недели) вводится в отдельной строке.
- Запись посещаемости

Отображение расписания в виде таблицы со статусами посещаемости. Каждый этап работы приложения разделен знаком "." (период). Абсолютная корректность данных гарантируется, за исключением последовательного порядка занятий при заполнении расписания.

Пример работы программы:

```

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		