Задание №3

**ДЛЯ ВСЕХ НАПРАВЛЕНИЙ**

Для тех, кто уже прислал №1 и №2.

На языке со статической типизацией (на выбор **Java** **или** **C#**) реализовать скрипт (в виде jar-файла или исполняемой сборки), который реализует обобщенную игру камень-ножницы-бумага (любое число произвольных комбинаций).

При запуске **параметрами командной строки** (аргументы метода main или Main) передаётся нечётное число >=3 неповторяющихся строк (при неправильно заданных аргументах вывести аккуратное сообщение об ошибке — что неверно, пример как правильно). Эти строки — это ходы (например, Камень Ножницы Бумага или Камень Ножницы Бумага Ящерица Спок или 1 2 3 4 5 6 7 8 9).

Победа определяется так — половина следующих по кругу выигрывает, половина предыдущих по кругу проигрывает (семантика строк не важна, в какой последовательности что пользователь ввел, в том и играет).

Скрипт генерирует случайный ключ (SecureRandom или RandomNumberGenerator — **обязательно**!) длиной 128 бит, делает свой ход (опять же безопасным рандомом), вычисляет HMAC (на базе SHA2 или SHA3) от хода со сгенерированным ключом, показывает пользователя HMAC. После этого пользователь получает "меню" 1 - Камень, 2 - Ножницы, ...., 0 - Exit. Пользователь делает свой выбор (при некорректном вводе опять отображается "меню"). Скрипт показывает кто победил, ход компьютера и исходный ключ.

Таким образом, пользователь может проверить, что компьютер играет честно (не поменял свой ход после хода пользователя).

При выборе опции "help" в терминале нужно отобразить таблицу, определяющую какой ход выигрывает.

Генерация таблицы должна быть вынесена в отдельный класс, определение "правил" кто победил должно быть в отдельном классе, функции генерации ключа и HMAC должны быть в отдельном классе (всего 4 класса). По максимуму следует использовать базовые библиотеки классов и сторонние библиотеки, а не изобретать велосипед.

ЧИСЛО ХОДОВ МОЖЕТ БЫТЬ ЛЮБЫМ (нечетным > 1, зависит от переданных параметров), не зашито в коде.

Пример:

>java -jar game.jar rock paper scissors lizard Spock

HMAC: FAAC40C71B4B12BF0EF5556EEB7C06925D5AE405D447E006BB8A06565338D411

Available moves:

1 - rock

2 - paper

3 - scissors

4 - lizard

5 - Spock

0 - exit

? - help

Enter your move: 2

Your move: paper

Computer move: rock

You win!

HMAC key: BD9B5544739FCE7359C267E734E380A2

Для сдачи нужно прислать на p.lebedev@itransition.com:

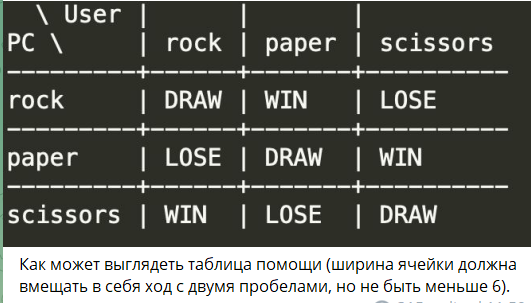
1) ссылку на видео, демонстрирующее запуск с 3 и 7 параметрами, запуск с неправильными параметрами (повторение или чётное число, один или отсутствие), генерацию помощи (на 5 параметрах), выбор хода, вывод результатов;

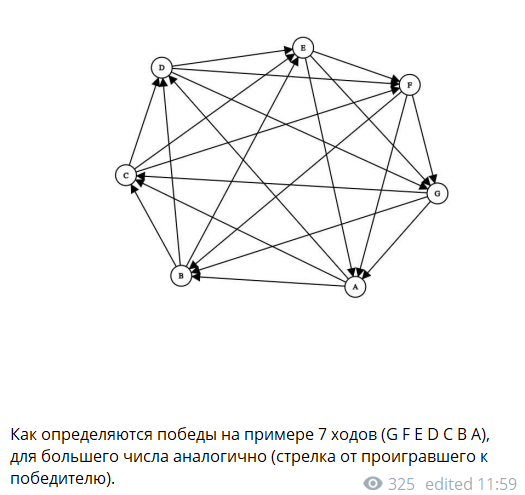
2) исходный код ссылкой на гитхаб.

И как пояснение: при вычислении HMAC ключ — это тот самый секретный ключ, который вы сгенерировали. А сообщение — это ход (прямо вот строка хода). После хода пользователя у него будет ключ, ход компьютера, пользователь сможет вычислить HMAC и сравнить с HMAC-ом, который был показан до хода пользователя. Это не очень сложно :))

Частая ошибка — попытка изобрести свой "HMAC" как хэш от случайного "ключа". Так не пойдёт. Если вы показываете одинаковые строки до хода и после хода, пользователь после хода не получает новой информации и, соответсвенно, вы ничего ему не доказываете. Нужно сгенерировать ключ (безопасным генератором), сделать ход компа, вычислить HMAC (стандартным алгоритмом) от хода компа (сообщение) и ключа (ключ), показать HMAC, получить ход пользователя, показать ключ. Перечитывать этот абзац до полного вкуривания.

"Правильный" порядок (хотя пользователь может использовать другой порядок и играть в игру, в которой ножницы побеждают камень; или играть в игру ХОД1 ХОД2 ХОД3): КАМЕНЬ БУМАГА НОЖНИЦЫ или КАМЕНЬ СПОК БУМАГА ЯЩЕРИЦА НОЖНИЦЫ.





В №3 "правильный" порядок (хотя пользователь может использовать другой порядок и играть в игру, в которой ножницы побеждают камень; или играть в игру ХОД1 ХОД2 ХОД3): КАМЕНЬ БУМАГА НОЖНИЦЫ или КАМЕНЬ СПОК БУМАГА ЯЩЕРИЦА НОЖНИЦЫ.

Коллеги, ТЕКСТ в таблицах должен быть выровнен по ЛЕВОМУ краю (для left-to-right систем записи). ЧИСЛА выравниваются по правому краю или же, если не целые, по десятичной запятой (точке).

Коллеги, я не просто так прошу вас переделать №3 на использование аргументов командной строки, не потому, что хочется помучать. 1) Это требование ТЗ, нельзя просто но настроению решить поменять формат входных данных. 2) Так принято передавать настройки в консольные приложения. 3) Это избавляет от кучи сложного недостаточно гибкого кода по разбору значений. Например, код, разбивающий данные по пробелам не позволяет использовать пробел в имени хода, например "Большой Камень" — а почему такое ограничение, с чего было оно вообще было? Аргументы командной строки не имеют такого ограничения (помимо этого там можно использовать переменные, например). Идея всё такая же — не нужно изобретать велосипед.

Изображение выглядит как текст

Автоматически созданное описание

Ссылки:

<https://github.com/freva/ascii-table> - вывод теста таблицей

<https://www.geeksforgeeks.org/random-vs-secure-random-numbers-java/> - рандом