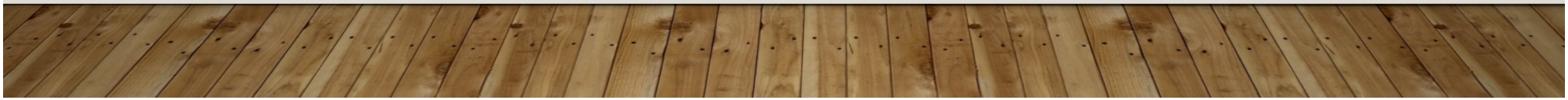
# ЛЕКЦИЯ 14

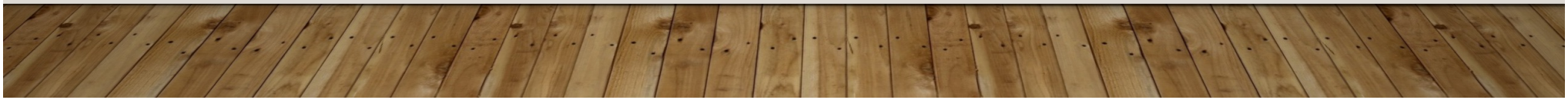
## ТРЕТЬЯ ВЕРСИЯ ВИСЕЛИЦЫ



# ПЛАН ЗАНЯТИЯ

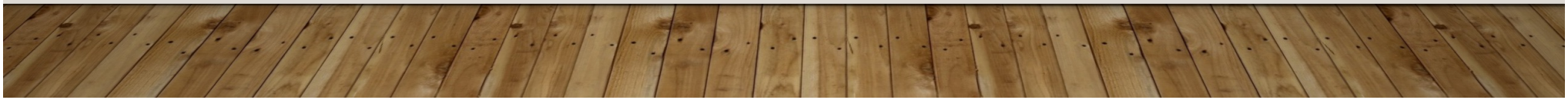
1. Используем файлы в «Виселице» — храним данные
2. Почему плохо хранить строковые данные в основном тексте программы

- Мы улучшим нашу игру Виселица так, чтобы в коде программы не было строковых констант. Научимся подгружать псевдографику из отдельных текстовых файлов в папке image и загадаем слово, выбирая его произвольно из отдельного файла words.txt.
- Заодно вспомним, что такое поля (переменные) класса в Ruby и как ими пользоваться.



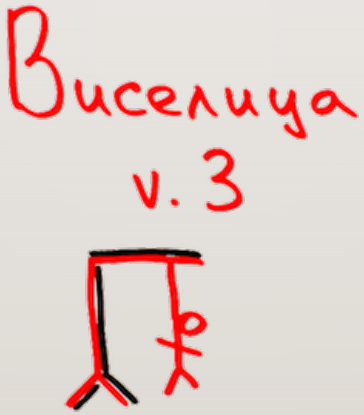
# УЧИМ ВИСЕЛИЦУ ЗАГАДЫВАТЬ СЛОВО САМОСТОЯТЕЛЬНО

- В обеих версиях нашей Виселицы (без классов и с классами) для игры нужен второй человек, который загадает игроку слово.
- Мы рады, если вам удалось найти коллегу и вы проходите наши уроки вдвоём, но что делать тем, кому не так повезло? :)
- Давайте сделаем так, чтобы программа загадывала одно из слов из подготовленного заранее небольшого словарика.



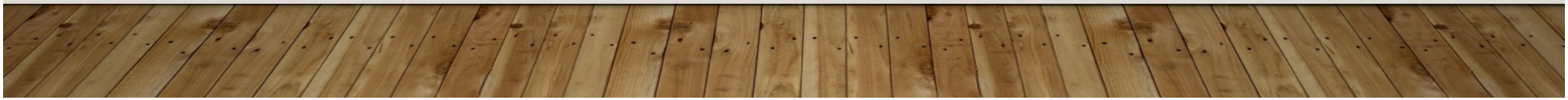
# УЧИМ ВИСЕЛИЦУ ЗАГАДЫВАТЬ СЛОВО САМОСТОЯТЕЛЬНО

- Мы будем улучшать код нашей старой программы, поэтому скопируйте три файла (game.rb, result_printer.rb и viselitsa.rb) из Виселицы v.2 в новую папку для текущего урока.



# СОЗДАЕМ СПИСОК СЛОВ ДЛЯ ЗАГАДЫВАНИЯ

- Не следует хранить данные, которые использует программа (ещё их часто называют «ресурсами») в той же папке, что и код программы.
- Поэтому снова создаём подпапку data и уже в ней создаём наш словарик words.txt. Придумайте свои слова ;-)
- Внимание! Пользователи Windows: не забывайте все файлы (как данные, так и код программ) сохранять в кодировке UTF-8.



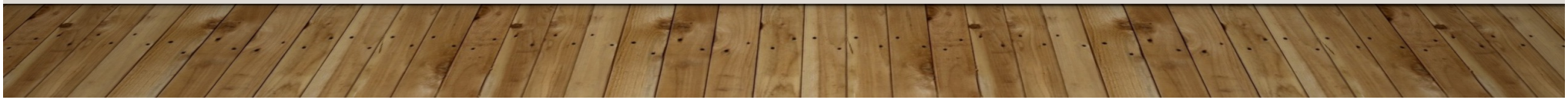
# КЛАСС WORDREADER

- Давайте придумаем, что в нашей программе файл со словами будет читать отдельный целый класс. Не будем сейчас рассуждать, насколько это целесообразно — выделять для такой простой функции целый класс, но наша задача научиться пользоваться классами и привыкнуть к процессу.
- Итак, как обычно, для нового класса — новый файл! Создаём файл word_reader.rb и пишем в нём наш новый класс:

```
class WordReader
  # тут будет описание класса
end
```

# МЕТОД READ_FROM_FILE

- Наш класс будет открывать файл с помощью единственного метода read_from_file, который на вход принимает один параметр: имя файла для чтения.
- Потом он будет из этого файла читать построчно все слова и возвращать один случайный элемент из массива строк (мы считаем, что в нашем файле каждое слово находится на отдельной строке).
- Напомним, что с тем, как читать из файла и возвращать произвольную строчку, мы разбирались в прошлом уроке.



# ИСПОЛЬЗУЕМ КЛАСС WORDREADER В ПРОГРАММЕ

```
require_relative 'word_reader.rb'
```

- Теперь нам надо создать экземпляр нашего нового класса WordReader:

```
reader = WordReader.new
```

- Ну и теперь вместо взятия слова из консоли мы передаём управление нашему новому объекту reader, вызывая у него метод read_from_file и передавая ему путь к словарику.

# ИСПОЛЬЗУЕМ КЛАСС WORDREADER В ПРОГРАММЕ

- Обратите внимание, что путь к файлику мы склеиваем из строк current_path и /data/words.txt:

```
slovo = reader.read_from_file(current_path + "/data/words.txt")
```

- Вот и всё, мы теперь читаем файл со списком слов и загадываем одно из них. Можно проверить это, запустив программу в консоли:

```
cd c:\%username\lesson14
ruby viselitsa.rb
```

- Обратите внимание, что теперь слово после названия программы писать не нужно.

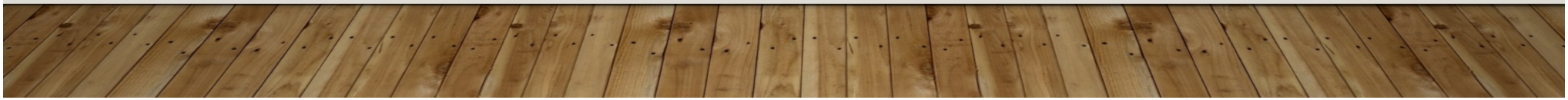
# ПОЧЕМУ ХРАНЕНИЕ СТРОКОВЫХ КОНСТАНТ В ПРОГРАММЕ – ЭТО ЗЛО!

- Как мы уже неоднократно говорили, данные следует хранить отдельно от кода программы. Давайте же посмотрим, что у нас творится в методе print_viselitsa класса ResultPrinter (файл result_printer.rb).
- А там — армагеддон, вот отрывок из файла:

```
when 6
  puts "
       -------
       |/
       |     ( )
       |      _|_
       |     / | \\
       |       |
       |      / \\
       |     /   \\
       |
     __|________
     |          |
     "
when 7
```

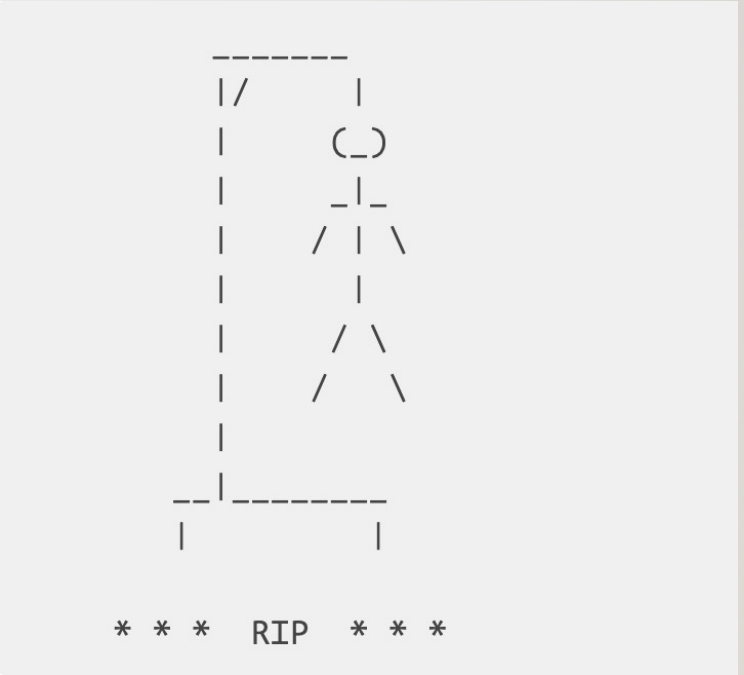
# ПОЧЕМУ ХРАНЕНИЕ СТРОКОВЫХ КОНСТАНТ В ПРОГРАММЕ – ЭТО ЗЛО!

- Это, фактически, графика нашей игры. И она у нас является частью текста нашей программы. Конечно, так лучше не делать. Умение отделять «зёрна от плевел» — важный навык программиста. Пока просто скажем, что графику вашей программы нужно выносить в отдельные файлы и папки.
- Фактически функционал метода print_viselitsa, как и всего класса не зависит о того, какую именно картинку он выводит. Что, если мы захотим поменять эту картинку, подрисовав её немного. К сожалению, для этого нам придётся править файл result_printer.rb, а это не совсем правильно, ведь логика работы класса останется неизменной.



# ПОЧЕМУ ХРАНЕНИЕ СТРОКОВЫХ КОНСТАНТ В ПРОГРАММЕ – ЭТО ЗЛО!

- Чаще всего, такие вещи выделяют в отдельные файлы. Давайте и мы поступим, как положено и вынесем картинки в отдельные текстовые файлы. Такие, как этот:

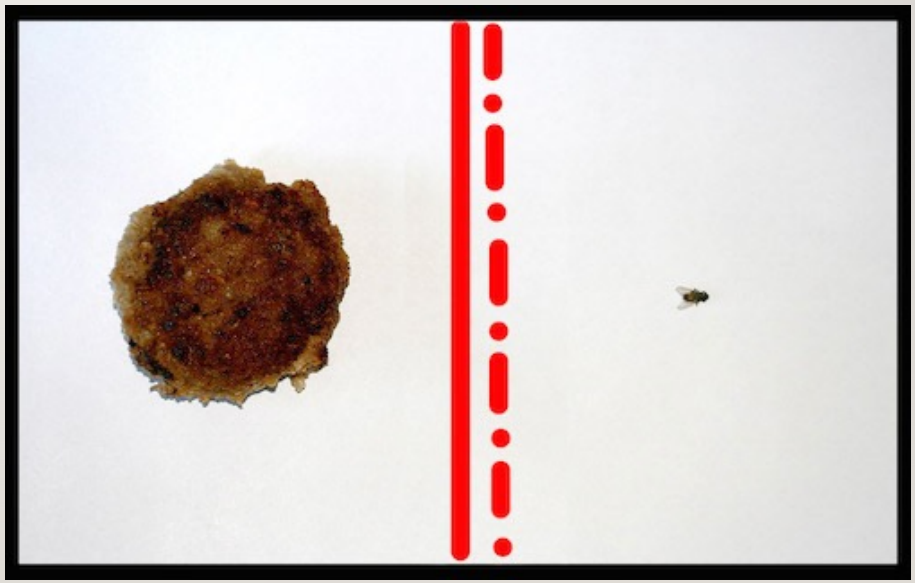


# ПОЧЕМУ ХРАНЕНИЕ СТРОКОВЫХ КОНСТАНТ В ПРОГРАММЕ – ЭТО ЗЛО!

- Самостоятельно перенесите псевдографику в отдельные файлы.
- Файлы с картинками (0.txt — 7.txt) мы положим в папку image, которую создадим в нашей рабочей папке c:\%username\lesson_14 (напомним, что данные отдельно, код программы — отдельно).

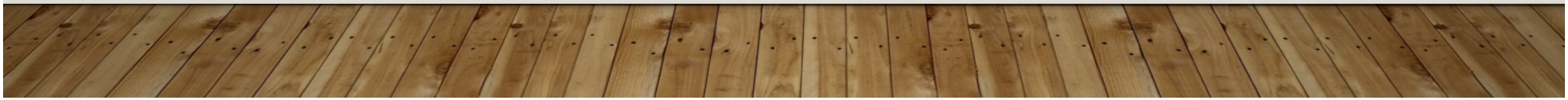
# ПОЧЕМУ ХРАНЕНИЕ СТРОКОВЫХ КОНСТАНТ В ПРОГРАММЕ – ЭТО ЗЛО!

- Другими словами – мухи отдельно, котлеты отдельно! Данные, нужные для работы программы, лучше отделять от логики программы (то есть от всех инструкций, методов и классов) и хранить в разных файлах.



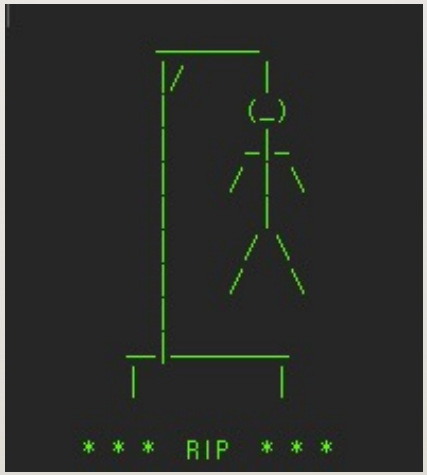
# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- Теперь давайте поправим наш файл result_printer.rb так, чтобы он выводил содержимое одного из этих файлов в зависимости от того, сколько ошибок ему передали в качестве аргумента.
- Обратите внимание на название файлов, они названы цифрами не просто так — цифра в названии файла соответствует количеству ошибок, которые допустил пользователь.



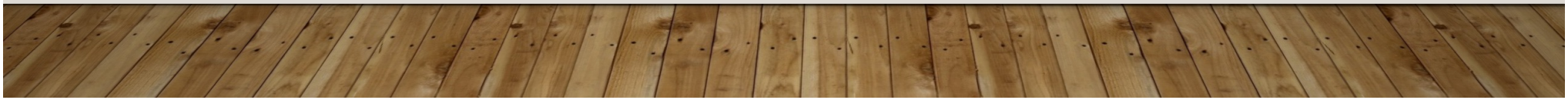
# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- То есть, в 0.txt у нас пустая виселица, а в 7.txt у нас повешенный человечек.



# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- Давайте загрузим все эти файлы в поле класса ResultPrinter при его создании. Каждый раз, когда что-то происходит при создании класса у прилежного ученика в голове должно возникать слово «конструктор», который описывается методом initialize.



# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- Добавим в конструктор класса ResultPrinter:

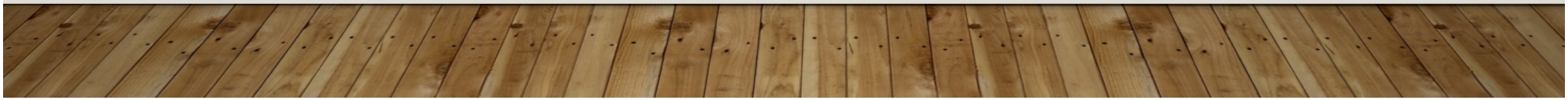
```
class ResultPrinter
  def initialize
    @status_image = []

    current_path = File.dirname(__FILE__)
    counter = 0

    while counter <= 7 do
      file_name = current_path + "/image/#{counter}.txt"

      if File.exist?(file_name)
        f = File.new(file_name, "r:UTF-8")
        @status_image << f.read
        f.close
      else
        @status_image << "\n [ изображение не найдено ] \n"
      end

      counter += 1
    end
  end
end
```

# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- Обратите внимание, что имена файлов мы собираем из двух частей: уже полюбившейся нам конструкции current_path (чтобы программу можно было запускать из любого места) и строки "/images/#{counter}.txt", в которую мы вставляем номер картинки из переменной counter, которая проходит путь от 0 до 7.
- А если вдруг такой файл не нашёлся, мы вставляем в массив вместо картинки строку «изображение не найдено», нет смысла заканчивать игру, если просто нет файла с виселицей, которая нужна для только красоты.

# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- Таким образом в поле класса @status_image у нас теперь храниться массив из 7-ми элементов, каждый из которых является строковой константой, которую можно вывести на экран, когда нам это потребуется: напомню, что мы можем использовать этот массив в любом методе нашего класса.
- Ну и конечно же, не забываем закрыть файл сразу после того, как получили от него всё, что нужно.



# РИСУЕМ КАРТИНКИ ВИСЕЛИЦЫ ИЗ ФАЙЛОВ

- Теперь метод print_viselitsa станет проще и понятнее, смотрите:

```
def print_viselitsa(errors)
  puts @status_image[errors]
end
```

- Вот и всё! Посмотрите, нам не пришлось менять саму программу, а также файл game.rb или другие методы класса ResultPrinter.
- В этом уроке мы улучшили нашу игру Виселица так, чтобы в коде программы не было строковых констант, научились загружать графику из отдельных текстовых файлов и загадывать слово, выбирая его произвольно из отдельного файла words.txt.

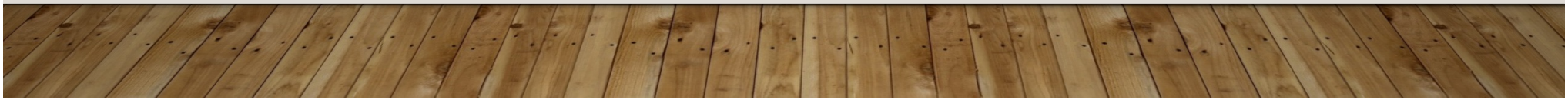
# ВЫБРАТЬ ФИЛЬМ НА ВЕЧЕР

- Напишите программу, которая рекомендует фильм к просмотру, доставая его произвольно из файла со списком фильмов.
- Создайте файл, со списком фильмов (одна строка - одно название). И напишите программу, которая берёт произвольную строку и выводит название фильма на экран:

```
Сегодня Вам предлагается к просмотру фильм:
Бриллиантовая рука
```

# ВЫБРАТЬ ФИЛЬМ НА ВЕЧЕР. ПОДСКАЗКА

- Создайте файл movies.txt со списком фильмов, прочитайте все его строчки в массив с помощью метода readlines и выведите произвольную строку с помощью метода sample.
- Если не получается — посмотрите прошлую лекцию, в которой мы точно также выбираем произвольную цитату из файла.



# ФИЛЬМ НА ВЕЧЕР С ОПИСАНИЕМ

- Дополните предыдущую программу описанием к фильму. В файле со списком фильмов следующей строкой после название идет краткое описание. То есть каждая вторая строка файла содержит описание к фильму на предыдущей строке.
- А при вызове программы, выводите произвольный фильм с описанием:

```
Кавказская пленница, или Новые приключения Шурика

Отправившись в одну из горных республик собирать фольклор,
герой фильма Шурик влюбляется в симпатичную девушку,
«спортсменку, отличницу, и просто красавицу».
Но ее неожиданно похищают, чтобы насильно выдать замуж.
Наивный Шурик не сразу смог сообразить, что творится
у него под носом, однако затем отважно ринулся освобождать
«кавказскую пленницу»...
```

# ФИЛЬМ НА ВЕЧЕР С ОПИСАНИЕМ. ПОДСКАЗКА

- Создайте файл movies.txt со списком фильмов с описаниями (обязательно проверьте, чтобы описание занимало ровно одну строчку без переносов, лучше сделайте описания краткими, чтобы не запутаться).
- Прочитайте все строки файла movies.txt в массив с помощью метода readlines, и выберите с помощью метода rand произвольное число от 0 до N, где N — количество строк в файле (элементов в массиве, метод length) + 1.

```
number = rand(movies.legnth + 1)
```

- Если произвольное число оказалось нечётным, уменьшите его на 1, после этого выведите строчку, соответствующую произвольному числу и строчку следующую за ней.

# СПРАВОЧНАЯ ИНФОРМАЦИЯ

- Виселица ASCII art
- Игра Виселица
- Документация класса File

# СПАСИБО ЗА ВНИМАНИЕ!

**Третья версия «Виселицы»**