Cassandra

A robot fortune-teller

Process

- Had 2 versions: basic with minimum functionality and extended with many features.
- Writing the main logic
- Refactoring with classes and methods (TDD)

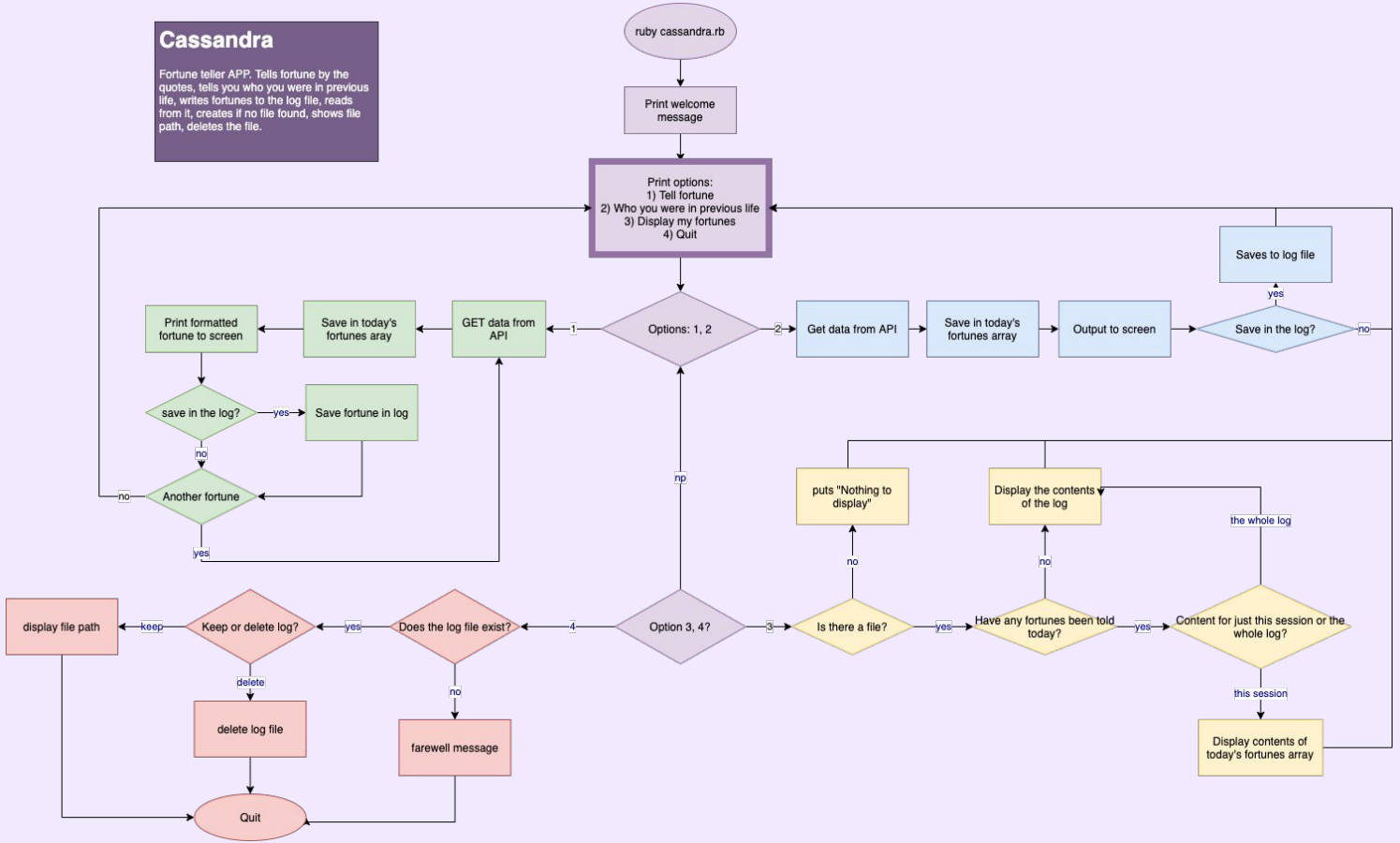
Favourite parts

- Using gems
- TTY prompt: gets rid of the validation issue
- TDD: love/hate relationship, helped a lot with breaking down problems
- Work with API-s: new, exciting and easy

Challenges / what I wish I did differently

- **Choosing** what to do: focusing on what I can not do instead of what I can
- Didn’t use **TDD** approach right away
- Figuring out what I wanted tests to do and writing **GOOD functional tests** (still not there)
- Neat and readable code
- Making the presentation under 10 min 😩

Flowchart


Cassandra
Fortune teller APP. Tells fortune by the quotes, tells you who you were in previous life, writes fortunes to the log file, reads from it, creates if no file found, shows file path, deletes the file.
ruby cassandra.rb
Print welcome message
Print options:
1) Tell fortune
2) Who you were in previous life
3) Display my fortunes
4) Quit
Options: 1, 2
1
2
GET data from API
Save in today's fortunes aray
Print formatted fortune to screen
save in the log?
yes
Save fortune in log
no
Another fortune
no
yes
Get data from API
Save in today's fortunes array
Output to screen
Save in the log?
yes
no
Saves to log file
Option 3, 4?
3
4
Is there a file?
yes
no
puts "Nothing to display"
Have any fortunes been told today?
yes
no
Display the contents of the log
Content for just this session or the whole log?
the whole log
this session
Display contents of today's fortunes array
Does the log file exist?
yes
no
Keep or delete log?
keep
delete
display file path
delete log file
farewell message
Quit

Data structures: Classes

- **FortuneTeller**
  - initializes with a name
  - user interaction, including pauses, progress bar
  - gets fortune from the API/other method and returns it
- **Log**
  - initializes with a file path
  - all file handling: read, write, delete, create if none found, show file path.
  - has an array of current session’s fortunes.
- **PreviousLife**
  - added later on
  - should probably be a module
  - has 1 method that fetches data from API, parses it into a table and returns it

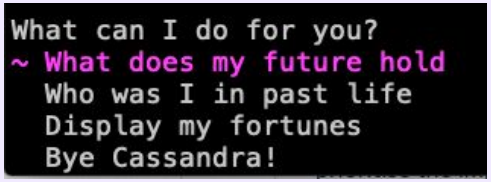
Testing

- Rspec
- Suite of 27 automated tests for classes
    - After removing .defined? tests for the methods that have other tests
- Extensive manual testing

Some features

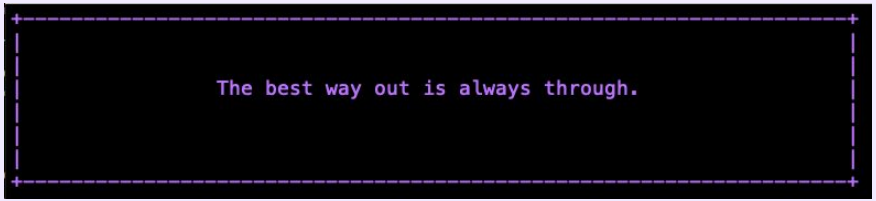
“What does my future hold”

- selected from the main menu
- a GET method inside FortuneTeller class getting and parsing a random quote from Quotable API
- Outputs a formatted string, styled with tty-box and rainbow
- Inside loops: save it in the log, have another go




```ruby
class FortuneTeller
  include HTTParty
  base_uri "api.quotable.io/"

def tell_fortune()
    fortune = self.class.get('/random').parsed_response["content"]
    return "\n#{fortune}"
  end
```



Past lives

- Handled with PreviousLife class with method for getting and parsing API data
- FortuneTeller uses it in .tell_previous_life method
- Outputs a table, styled with terminal-table and rainbow

```ruby
include HTTParty
  base_uri "pipl.ir/v1/"

  def display_previous_life
    person_hash = self.class.get('/getPerson').parsed_response["person"]
    rows = []
    rows << ['First name', person_hash["personal"]["name"]]
    rows << ['Last name', person_hash["personal"]["last_name"]]
    rows << ['Age', person_hash["personal"]["age"]]
    rows << ['Birth place', person_hash["personal"]["country"]]
    rows << ['Partner\'s name', person_hash["personal"]["father_name"]]
    rows << ['Marriage', person_hash["marriage"]["married"]]
    rows << ['Eye color', person_hash["personal"]["eye_color"]]
    rows << ['Height', "#{person_hash["personal"]["height"]} m"]
    rows << ['Weight', "#{person_hash["personal"]["weight"]} kg"]
    rows << ['Profession', person_hash["work"]["position"]]
    rows << ['Salary', "#{person_hash["work"]["salary"]} per month"]
    table = Terminal::Table.new :rows => rows
  end
```

```ruby
def tell_previous_life()
    fortune = PreviousLife.new.display_previous_life()
    return fortune
  end
```

```
+---------------+-------------------+
| First name    | Nady              |
| Last name     | Foster            |
| Age           | 43                |
| Birth place   | Germany           |
| Partner's name| Krysta            |
| Marriage      | true              |
| Eye color     | Blue              |
| Height        | 1.48 m            |
| Weight        | 32 kg             |
| Profession    | Doctor            |
| Salary        | $5.000 per month  |
+---------------+-------------------+
```

File handling with Log class

- Read and create file if none found
- Write to file (with a formatted date and time)
- Delete file
- Show file path

```
def read_from_file
  begin
    fortunes = File.readlines(@file_path).map{|fortune| fortune.strip}
  rescue
    puts "Looks like you don't have any saved fortunes yet. Creating your personal
    Fortunes Book.\n"
    File.open("./logs/fortunes-from-cassandra.txt", "w") {|file| file.write("")}
    fortunes = @todays_fortunes
  end
end
```

```
def write_to_file(fortune)
    File.open(@file_path, "a") {|file| file.write("\n\n#{Time.now.strftime(
"%d %B, %Y %H:%M")}\n#{fortune}") }
end

def delete_file
  File.delete(@file_path)
end

def show_file_path
  File.expand_path(File.dirname(@file_path))
end
```

Command line arguments

- 4 options: help, file path, name and version
- -h parses the readme file with tty-markdown and displays it in customized format
- -p and -n pass the second argument as a variable to the programme using flag, *rest = ARGV

```
when '-h'
    parsed = TTY::Markdown.parse_file("../README.md", theme: {
      em: :bright_cyan,
      header: [:bright_magenta, :bold],
      hr: :bright_cyan,
      link: [:bright_cyan, :underline],
      list: :bright_cyan,
      strong: [:bright_cyan, :bold],
      table: :bright_blue,
      quote: :bright_blue,
      image: :bright_black,
      note: :bright_cyan,
      comment: :bright_black
      })
    puts parsed
    exit
```

```
when '-p'
  file_path = Log.new(rest[0])
when '-n'
  username = rest[0]
```

Exception handling

- Separate class of a Validation error with custom message. Raised in the main loop “else” statement, although TTY-prompt eliminated the need for it

```
class ValidationError < StandardError
  def message
    return "\nYou should be very clear while talking to
    spirits! Please input the valid option"
  end
end
```

- Rescue StandardError in the main loop

```
rescue
  puts "\nThe future is unclear. Try again later."
  exit
end
```

Live demo 🔮