‘lil Library app

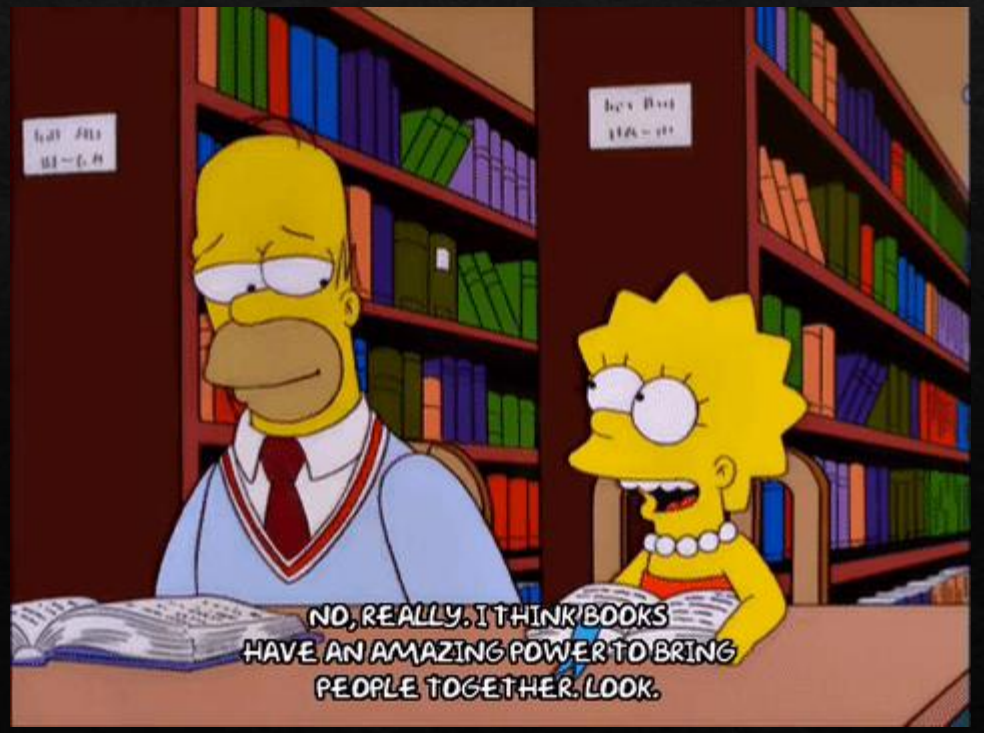
By Alison Killen

Other app ideas

- Text based game
- Star/constellation index
- Chose the library because I could visualise how it worked and had lots of scope for improvement

Why this app? + Purpose

- I love libraries! And use their services a lot
- Wanted to make a simulation of a library app where users can browse+borrow books
- For anyone who enjoys libraries



Main functions/user stories

- Users can browse books by genre
- Users can borrow a book
- Users can view account status
- Options menu

Logic

- Book class – properties + methods
- User class – what users can do
- Library class – what the library can do
- Main app – calling everything
- Options module - determines most of control flow

What can the library class do

- A few things! Various methods to return for determining list of genres to display, listing those books, enable users to choose genre by number

```ruby
# puts the book from library if the title matches title in library
def getBook(book_title)
  @library.each do |book|
    if book.title.downcase == book_title.downcase
      return book
    end
  end
  # maybe could puts "here is your book: #{book_title}"
  return nil
end

# each book has its own id which is its index position in array
def getBookById(index)
  return @library[index]
end

def listBooksByGenre(genre_choice) #this method needs to take user input of genre
  indexes = []
  @library.each_with_index do |book, index| #only want to push books that match genre_choice
    if book.genre.downcase == genre_choice.downcase
      indexes.push(index)
      #BUG HERE - need to print books from 1 not from 0, and error if you type in gibberish
          #when choosing a book - because thats index0, it will think you want to borrow title with id of 0
      puts "Id: #{index}"
      puts book
      puts
    end
  end
  return indexes
```

```ruby
# Library class defines all methods the library can perform
require_relative ('./module.rb')

class Library
  include Options

  def initialize
    @library = [] # array of Book
    @genre_choices = []
  end

  def showGenreChoices
    puts @genre_choices
  end

  def addBook(book)
    @library.push(book)
  end

  #if a genre is in the library of books, it will be pushed into the genre choices array
  def getGenres()
    @library.each do |book|
      if !@genre_choices.include?(book.genre)
        @genre_choices.push(book.genre)
      end
    end
    return @genre_choices
  end
```

What can the book class do

- Initialises a book with title, author, year, genre, format, borrowed attributes
- Can set a book as available or not available

src > bookclass.rb

```ruby
class Book
  attr_reader :title, :author, :year, :genre, :format
  def initialize (title, author, year, genre, format)
    @title = title
    @author = author
    @year = year
    @genre = genre
    @format = format
    @borrowed = "Available"
  end

  # borrow book
  def borrow
    @borrowed = "Currently borrowed - not available in library"
  end

  # return book - not using this method yet but would like to in future
  def returnBook
    @borrowed = "Available"
  end

  # will show book Availability status as "Available" to borrow
    def isAvailable
    return @borrowed == "Available"
  end


  def to_s #overridding the to_s function in the class because it will default to putting
      #object ID and we want it to put the instance of the object
    return  "  Title: #{@title}\n"+
```

What can the user class do

- Initialises user class with username and user_id, borrowed books array
- Borrowbook method pushes chosen book object into borrowed books array
- Display user details.....displays the user details. Displays users borrowed books by iterating over borrowed books array and putting each title/author

src > userclass.rb

```ruby
# User class defines all methods the user can perform
class User
    attr_reader :username
    def initialize (username)
        @username = username
        @user_id = rand(1000)
        #[array of "book" objects which the User has under his / her name]
        @borrowed_books = []
    end

    def borrowBook(book)
        @borrowed_books.push(book)
    end

    def display_user_details
        puts "Here is your account status:"
        puts
        puts "User Name: #{@username}"
        puts "User Library Card Number: #{@user_id}"
        # counter to count books borrowed needed here maybe, or in future have a method
        puts "Book Allowance Remaining = currently unlimited"
        puts
        puts "Books Borrowed:"
        puts
        @borrowed_books.each do |book|
            puts "  Title: #{book.title}\n" +
                 "  Author: #{book.author}"
            puts
        end
    end
```

The module – aka the options menu

- Most of the code is here
- Enables users to interact with and manipulate library data
- Displays options and gets number to choose
- Method for displaying user details
- Method for getting genre user wants to browse
- Execute options choices, validates, and then loop back to options menu
- Defines handle_borrowed_book which checks if book is borrowed or not and if not sets it as borrowed if user wants to borrow it
- Defines handlequit method that allows users to confirm and quit app

src > module.rb

```ruby
module Options

  require "tty-progressbar"
  require "colorize"

 def displayOptions(user, library)

  puts "What would you like to do? Type a number to choose."
  puts

  options = [ "View my account status",
              "Browse books by genre",
              "Quit".colorize(:red)]

  counter = 1

  options.each do |option|
    puts "[#{counter}] #{option}"
    counter += 1
  end

  #had to change all the gets to STDIN.gets because of CLI arg being parsed
  answerchoice = STDIN.gets.chomp.to_i

  # validate answer choice and prompt for valid answer
  # user input validation has been implemented at every stage of the app to handle errors /control flow
  until answerchoice >= 1 && answerchoice <= 3
    puts "That is not a valid option I'm afraid! Try entering a number from 1-3 to select an option."
    answerchoice = STDIN.gets.chomp.to_i
  end
```

The main app

src > libraryapp.rb

```ruby
require_relative('./userclass')
require_relative ('./module.rb')
# require_relative ('./ascii.txt')
include Options

require "colorize"


# create the library
library_array =
    [    {:title=>"Breathing", :genre=>"Lifestyle", :author=>"Mr. Emmett Windy", :year=>"1999", :format=> "Hardcopy"}
         {:title=>"Cooking with Veggies", :genre=>"Lifestyle", :author=>"Gordon Ramsey", :year=>"2006", :format=> "Ha
         {:title=>"Rock Climbing", :genre=>"Lifestyle", :author=>"The Grinch", :year=>"2016", :format=> "Hardcopy"},
         {:title=>"Stuck In The Well", :genre=>"Mystery", :author=>"Goosebumps McFadden", :year=>"2010", :format=> "H
         {:title=>"Train Murder", :genre=>"Mystery", :author=>"French Dude", :year=>"1906", :format=> "Hardcopy"},
         {:title=>"How to Code", :genre=>"Mystery", :author=>"Devops Engineer", :year=>"2020", :format=> "Hardcopy"},
         {:title=>"We Love the Greeks!", :genre=>"History", :author=>"The Romans", :year=>"500", :format=> "Hardcopy"
         {:title=>"We Went to the Moon", :genre=>"History", :author=>"NASA", :year=>"1969", :format=> "Hardcopy"},
         {:title=>"Fashion", :genre=>"History", :author=>"Rupaul", :year=>"2020", :format=> "Hardcopy"}]

#generate new instance of library class
library = Library.new
# populate library with books
library_array.each do |book_info|
  book = Book.new(book_info[:title], book_info[:author], book_info[:year], book_info[:genre], book_info[:format])
  # add book to library
  library.addBook(book)
end
# populate genres in the library
library.getGenres
```

libraryapp.rb

```ruby
  # File.readlines("ascii.txt") do |line|
  #   puts line
  # end
  puts "     __...--~~~~~-._   _.-~~~~~--...__ ".colorize:green
  puts "  //               `V'               |".colorize:green
  puts " //      Welcome to the Library        | ".colorize:green
  puts "//__...--~~~~~-._  |  _.-~~~~~--..... | ".colorize:green
puts "//__.....----~~~~._/ | /_.~~~~----.......|".colorize:green
puts "====================||/=================".colorize:green
puts                     "                   `---`".colorize:green
end

# Welcome - get name

def welcome
  system 'clear'
  render_ascii_art
  puts
  puts "Welcome to the Library! What's your name?".colorize:blue
  name = STDIN.gets.chomp.to_s.capitalize
  return name
end

#def main

  name_from_arg_arr = (ARGV.length > 0) && ARGV
  # puts name_from_arg_arr
  # return
  # person_name = name_from_arg_arr[0]
```

- Requires other files
- Contains library hash
- Creates new library and populates it with books
- Method to render ascii art
- Defines welcome method to get user name
- If user passes name in from CLI arg in shell script, it will use that name
- If user doesn't pass name in, it will get name using welcome method
- Runs options module on loop – all methods etc called from here

Struggles

- So many to choose from!
- Understanding classes and defining methods within them
- Control flow, making sure nothing breaks the app – testing, testing, testing
- Understanding how to use diff methods within a class method (modules)
- Felt like one step forward, five steps back - Making some new feature and then it breaks everything
- Options menu – refactoring all control flow and restructuring around it

Learnings

- The 3 C's - Coding, coffee, crying
- Development journal helped keep me on track and keep me sane
- Pattern of work – plan sections, breaks, flow state, explain it to somebody

- Classes
- Importance of control flow – redid diagram lots – think about it earlier
- More commits = more happiness (for when you break everything)
- Love your error messages
- Comment well
- INDENT as you go. Otherwise end errors everywhere
- Keep refining control flow - make everything modular asap

Don't worry about making it pretty at first

Scope for Improvement/Additional Features

- Had to scrap a lot of features I thought of for the sake of keeping it simple
- Right now is single user/single session
- Add more gems/make it pretty
- Limit on amount of books they can borrow
- Put not available books on “hold”
- Late fees function
- Users can add books to the library
- Return a book feature – (scrapped)
- More genres + books – generated by faker
- More book formats – ebooks, audiobooks (scrapped)
- Accessibility – speech to text
- Library of Alexandria ‘skin’ – with cool art and book titles

Thank you!

- Thank you so much to all the educators! Shoutout to Janel for spending lots of time with me in the mornings and to Simon for helping me with gems and scripting
- Thanks to all classmates on discord ☺