So, the first thing that I had to do while creating the Spotify simulator was to define a user class, as, well, you need to set up some kind of personalized environment. Before defining the class, I forced the user to create a username, email, and password, which would be fed into the argument once a new user is created. The initialize method saves the variables previously defined into local variables. Then, I defined the standard writer methods, such as getname (which returns a new user’s name), getpassword (which does the same for password), and getemail, (which does the same for email). Finally, I created a reader method called createplaylist, which would create a new playlist, and save it in the playlist class, which is defined above the user class. In the initialize method inside the playlist class, it sets the name to whatever the user inputs. Also, @playlist is defined as a local variable. The playlistname method simply returns the name of the playlist. In addition, the method “addsong” adds a song to the playlist, and tells the user what songs are inside the playlist. On a broader scale, the User class adds a more personalized user experience, and the playlist class adds more functionality to the simulator, such as adding songs to playlists. The writer methods in the user class allow the user to look at their user credentials, similarly with the playlist class. If there was any way to improve the code, it would be to put the first few lines of code into the initialize method in the use class. This would make it so that you wouldn’t force the user to type in new credentials. Also, creating a hash for the playlist would be more effective, so as to take advantage of the key value pair. The code is as follows:

#This is a Spotify Simulator, pls enjoy!

puts "Hello, and welcome to spotify. Please unput your name"

name = gets.chomp

puts "Great. Now type in your email."

email = gets.chomp

puts "Excellent. Now type in your password."

password = gets.chomp

puts "Please confirm your password."

until gets.chomp == password

answer = gets.chomp

password = answer

puts "Confirmation failed, please retype your password."

end

class Playlist

def initialize

puts "What would you like to name your playlist?"

answer = gets.chomp

@name = answer

@playlist = []

end

def playlistname

return @name

def addsong(song)

@song = song

@playlist.push(@song)

puts "So far, your playlist consists of #{@song}."

end

end

class User

def initialize(name, email, password)

@name = name

@email = email

@password = password

end

def getname

return @name

end

def getpassword

return @password

end

def getemail

return @email

end

def createplaylist

@playlist = Playlist.new

end

end

Sources:

<http://mlomnicki.com/programming/ruby/2011/07/20/class-vs-instance-methods.html>

<http://programmers.stackexchange.com/questions/162349/should-i-prefer-instance-methods-over-class-methods-in-ruby>

https://blog.engineyard.com/2015/instances-classes-and-modules