Ruby Programming

EX.No : 3

Aim

To implement a ruby Programming to solve a below mentioned problems.

1. Hobby Matcher

Program a class “Person” that implements matching Persons for their hobbies, implementing those two user stories:Vision/Goal: As a User of the Service, I want to be able to find other People with the same hobbies as I do.

Story 1: As a User of the Service I want to be able to enter my Hobbies as a comma-separated list.

Story 2: As a User of the Service I want to see a list with People with whom I share hobbies including the hobbies we share, ordered by the number of shared hobbies (descending).

Implement this in a Ruby-Class “Person” (use Person as a starting point), holding the name and a list of hobbies in an Array of a single person.

It should be possible to pass the list of hobbies as a single comma-seperated String (“Go, Geocaching, Stunt Kites, Bicycles”) to a setter method and be stored internally as an Array. (String#split might be useful for this.).

Write one method returning the common hobbies of two persons. Write another Method finding all other people with similar hobbies.

2. Iteration over an Array

You have a new client who is interested in creating an invoicing program for an online store. This is very early in development, and all you need to do is get a start at creating a way to save prices on an order and do some simple calculations using two different methods. The prices on an order will vary in number. An order can have anywhere from 1 to n prices, but I don’t expect you to write the code to ask for and accept input … just create an array in your code for testing for now.

Method 1: add all of the prices together to get a total

Method 2: take 1/3 off each price on an order and calculate (and display) the discount

For now, let’s say we are in testing mode. Rather than worry about accepting user input, create in your code an arbitrary collection of prices that will be used for testing the methods.

Have your program:

Print to the screen the list of prices you are using to test.

Print the total of the prices

Print the discount

3. Guess Random Number

Write a program to play ‘guess the number’. Your program should:

1. Generate a random number between 1 and 100 using ‘rand(100)’
2. Ask the user for a guess using ‘gets’
3. Use a conditional to check if the user guessed correctly or needs to try again
4. Limit the user to 10 guesses
5. print out a “congrats” or “sorry you lose message” depending on the outcome

4.Classes and Inheritance

You are very busy and have another client who wants to develop an app for sharing videos, music, photos, etc. After you and your team stop rejoicing, you get down to work in developing the classes and class methods you will need for this project. For now, just focus on defining the classes for videos and music and start with very limited functionality, but keep in mind that you will almost certainly be adding features in subsequent sprints.

You notice that music and videos have some of the same functionality: the client thinks that users will 1) want to play songs and videos and 2) leave comments on them.

However, music and videos also have significant differences. We often classify songs by beats per minute whereas we care about frames per second, display size, and resolution on videos. So there are similarities and differences. So, you want to create a superclass that encompasses the similarities and have movie and video subclasses inherit those but extend the functionality for attributes that are different.

Again, you don’t have to create the user interface. Use embedded, “dummy” data when creating objects for testing and output.

Have your program:

1. Define a superclass with attribute reader for comments, an initialize method, an add_comment method that uses ‘push’, and a play method (which just prints “Playing” or something)
2. Define two subclasses that inherit from the superclass and extend functionality by adding a resolution attribute for videos and a beats per minute attribute for music using attr_accessor
3. Create a video
4. Create a song
5. Add comments for both the song and the video (these can be hard coded testing method calls rather than reading input)
6. Print out the comments for the song and the video

Note: both of these are from the book Head First Ruby by Jay McGavren. The book is highly readable, but it does walk you through concepts slowly and in detail. I personally find it very useful, but its pace may be frustrating for others. You might want to use these exercises from the Head First Ruby site to practice more Ruby.