Ruby – Homework 1

Short Answer and Thinking Assignment

1. Hello World

```
# 1) Hello World:
puts "Hello World"
```

Data Structures

- A list is a data structure that can hold objects such as integers, strings, doubles, and Boolean values.
- A hash (dictionary) is a data structure that can hold data in key, value pairs, such as (“name” => “Jacob”).
- A set is like a list but a set cannot hold any duplicates.

List (Array) Example

```
# 3) Array 1-10 and print in console:
list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
puts list
```

Hash Example

```
# 4) Dictionary of information:
dict = {"first_name" => "Jacob", "last_name" => "Rozell",
"tiger_email_id" => "jcr0058", "banner_id" => "903831908",
 "fav_movies" => ["Star Wars", "Boondocks Saints"] }

puts dict
```

```ruby
class Person
  def initialize(age, height)
    @age = age
    @height = height
  end

  def shout()
    puts "AHHH!"
  end
```

Class and object in OOP with Ruby

- In Object-Oriented programming, classes are the backbone of objects through data and methods. Objects are an instance of the class that have access to those methods.

OOP in Ruby cont.

- Now that the Person class is created, we can make a person object, an instance of the class, to utilize the methods found in the class backbone. The object clay has access to the methods found in the class Person. clay.shout will return "AHHH!" from the method shout() found in the Person class.

```
clay = Person.new(20, 5.7)
clay.shout
```

```
AHHH!
```

Encapsulation

- Encapsulation is how objects cannot be accessed without getters/setters. Objects cannot access variables directly.
- From the example code, toy_story.getName will return the "Toy Story" because it has access to a getter method.

```
class Movie
  def initialize(name, year, genre)
    @name = name
    @year = year
    @genre = genre
  end

  def getName()
    puts @name
  end
end

toy_story = Movie.new("Toy Story", "1995", "children")
toy_story.getName
```

Encapsulation

- However, toy_story.year will throw an error because the toy_story object doesn't have access to the variable year.

```
class Movie
  def initialize(name, year, genre)
    @name = name
    @year = year
    @genre = genre
  end

  def getName()
    puts @name
  end
end

toy_story = Movie.new("Toy Story", "1995", "children")
toy_story.getName
```

```ruby
class Movie
  def initialize(name, year, ge
    @name = name
    @year = year
    @genre = genre
  end

  def getName()
    puts @name
  end
end
```

Inheritance

Inheritance allows a child class to inherit all methods and variables from a parent class.

```ruby
class Mystery < Movie
end

scooby_doo = Mystery.new("Scooby Doo", "2002", "mystery")
scooby_doo.getName
```

Inheritance

The Mystery class is the child of the Movie class, so it has inherited the methods found in Movie.

Scooby_doo.getName will return “Scooby Doo”

Abstraction

Although base Ruby doesn't support abstract classes, they can be mimicked.

Anytime the Instrument class is initialized as a object it will raise an error.

However, when a Guitar is initialized it will work normally.

```ruby
class Instrument
  def initialize()
    raise "Error: Subclass must overwrite initialize"
  end
end

class Guitar < Instrument
  def initialize(family, model)
    @family = family
    @model = model
  end
end
```

Animal Class Example

- Here an Animal class is made with getter/ setter methods for the 2 variables found in the initialize method.
- A method called bark() is also created.
- A dog object is made with 4 legs and fur. The dog then can be called to bark.

```ruby
class Animal

  def initialize(num_of_legs, fur)
    @num_of_legs = num_of_legs
    @fur = fur
  end

  def getLegs()
    return @num_of_legs
  end

  def getFur()
      return @fur
  end

  def setLegs(legsIn)
    @num_of_legs = legsIn
  end

  def setFur(furIn)
    @fur = furIn
  end

  def bark()
    "Woof!"
  end

end

bear = Animal.new(4, true)
dog = Animal.new(4, true)
dog.bark
```

attr_reader, attr_writer, and attr_accessor

- attr_reader will return the value of an instance variable without the need of a getter method.
- attr_writer will allow one to set the value of an instance variable without the use of a setter method.
- attr_accessor lets one change the value of a method and return the value at the same time.

Thinking Problem

- Assignment: Write an optimized solution:
- You have been given an array with millions of entries in it. The list is not sorted, but this is a special list, which contains all entries in incremental order and at some point, it starts decreasing. How do you find that fluctuation point index? (NOTE: array can contain duplicate numbers.)