Lecture 12
CSE/ISE 337 Scripting Languages
Ruby 4 : Lecture 12

Instr: Prof Abid M. Malik
3/4/2023

Last Time

- Exceptions
- Classes
  - Inheritance
  - Polymorphism
  - Singleton
  - Modules
  - Mixin
- Maths Library
- Symbol

Today's Agenda

- Object Oriented Programming
  - Module
  - Concept of Mixin
  - Math modules
  - private, public, or protected
  - Aliasing
- Symbol in Ruby
- Some examples from OOP
- Regular Expressions (if time permits)
- More examples (if time permits)

Practice problems in Ruby Programming

Problem 1

```
=begin

Ignoring case when sorting strings

=end

list = ["Albania", "antreater", "zorilla", "Zaire"]

list1 = list.sort

print list1
```

["Albania", "Zaire", "antreater", "zorilla"]

["Albania", "antreater", "Zaire", "zorilla"] (required)

Problem 2

```
=begin

Group by Length: Write a method that takes an array of strings and groups them
into a hash

where the keys are the lengths of the strings, and the values are arrays of
strings with that length.

=end

# Example usage:

puts group_by_length(["cat", "dog", "elephant", "bird", "lion"])

# Output: {3=>["cat", "dog"], 8=>["elephant"], 4=>["bird", "lion"]}
```

Problem 3

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

```
people = [
  Person.new("A", 25),
  Person.new("P", 30),
  Person.new("R", 25)
]
```


Problem 4

Given an array of numbers, write a Ruby method called filter_even_numbers that takes an array of integers as input and returns a new array containing only the even numbers from the original array. Use a block to implement the filtering logic.

```
# Test
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
puts filter_even_numbers(numbers)
# Output: [2, 4, 6, 8, 10]
```

Problem 5

Given a hash containing students' names as keys and their corresponding scores as values, write a Ruby method called top_students that takes a hash as input and returns a new hash containing only the top scoring students. Use a block to implement the filtering logic.

```
# Input
scores = {
  "Alice" => 85,
  "Bob" => 72,
  "Charlie" => 90,
  "David" => 95,
  "Eve" => 88
}
puts top_students(scores)
# Output: {"Charlie"=>90, "David"=>95, "Eve"=>88}
```


Problem 6

You are given an array of strings representing words. Write a Ruby method called group_anagrams that groups the words into anagrams. An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

```
# Test
words = ["eat", "tea", "tan", "ate", "nat", "bat"]
puts group_anagrams(words)
{"aet"=>["eat", "tea", "ate"], "ant"=>["tan", "nat"], "abt"=>["bat"]}
```

Quick overview of a class in Ruby

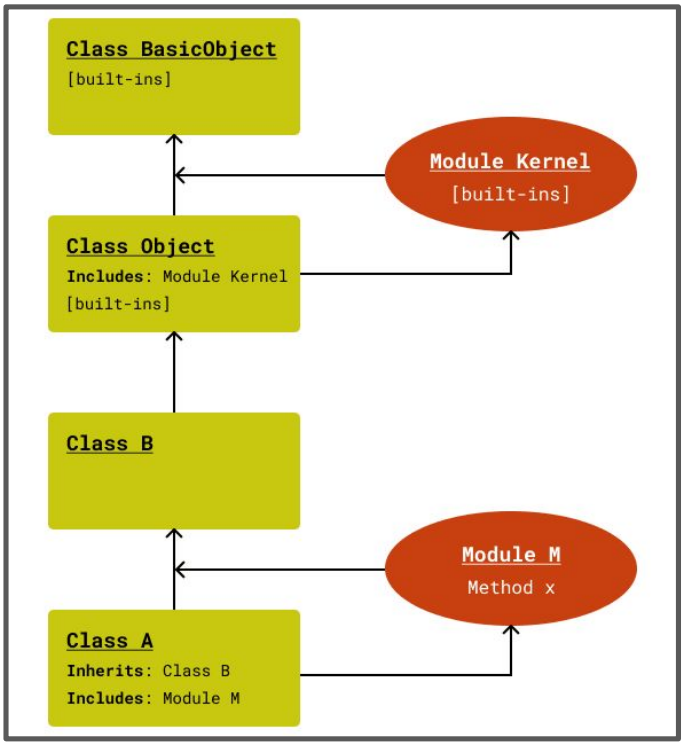
- class keyword
- instance variables
- class variables
- getter and setter
- accessors
- class methods
- class inheritance
- super keyword
- self keyword
- singleton
- constructor

Modules

- Modules defines characteristics
- Class defines an entity
- Examples (jupyter notebook file)
  - Animal/Person class
  - Dice class
  - Stack class

Modules in Ruby

- Methods lookup path
- Code on jupyter notebook



Concept of prepend in Module

- It works almost the same as include, the difference is that when you prepend a module to the class, the object will look for the method in the module first, before looking for it in the class.
  - Check code on Jupyter notbook

Extending modules using extend method

- Similar to Singleton method

Use of "super" in Modules

- Ordering of Modules
- Namespace
- Aliasing

Examples on OOP

- Jupyter notebook

Concept of Namespace using Modules in Ruby

In Ruby, the `::` operator is used for namespace resolution. It allows you to access constants, classes, and modules defined within modules or classes.

Maths Module

Math module consists of the module methods for basic trigonometric and transcendental functions.

Concept of Aliasing in Ruby

- **Check the last presentation on this**

11 private, public, or protected

`private`, `public`, and `protected` are access control keywords used to control the visibility of methods within a class. They determine which methods are accessible from outside the class and which are not.

13 Symbols

Symbols were made to be used as *identifiers* (i.e., variables, method names, constant names).

In contrast, strings were made to be used as *data*.

Symbols vs strings

- **A string, in Ruby, is a *mutable* series of characters or bytes.**
- **Symbols, on the other hand, are *immutable* values. Just like the integer 2 is a value.**

Symbols are faster

Since, symbols are immutable values, working with symbols requires less memory, and it's just easier for computers to work with literal values than it is to work with complex objects. So, lookups for symbols are faster as well.

Do another detailed example using OOP

- **This will touch all the concepts of Ruby as well**

Questions!

Next Time (3/6/2024)

- **Regular Expressions**
- **More examples**