Ruby 101

Ruby Basics:Creating Classes Continued

Ruby Basics:

Initializing an instance:

Tell Ruby what to do when you instanstiate a class.

```
class Table
    def initialize ilength = 3, iwidth = 3, iheight = 3
        @length = ilength
        @width = iwidth
        @height = iheight
    end
end
```

Ruby Basics:

```
my_table = Table.new
```
=> Length, Width, and Height all equal 3.

```
my_table = Table.new 5, 10, 3
```
=> Length = 5, Width = 10, and Height = 3.

Ruby Basics:

Private methods:

Can only be accessed within the class code, not outside of it.

```
class Table

    private

    def table_top_area
        @length * @width
    end

end
```

Ruby 101

Ruby Basics:

```
my_table = Table.new
```

```
my_table.table_top_area
```

Ruby Basics:

Inheritance:

Default inheritance from "Object" class.

Give access to all of a parent's properties and methods.

```
class LibraryTable < Table

    def cost
        table_top_area * 10 + 4 * @height * 3
    end

end
```

Ruby Basics:

Protected Methods:

Make sure a method is only accessed in the code of sibling objects or child objects of the defining class.

Useful for comparisons.

```
protected
```

Ruby Basics:

Try it yourself:

Write a class that models a simple account book. There should be methods for deducting and adding to the account, with a memo for each transaction. There should also be a method which prints out all the transactions.

Ruby 101

Ruby Basics:

```
class AccountBook
    def initialize
        @ledger = []
        @current_total = 0
    end
    def add_money amount, memo = ""
        @ledger[@ledger.count] = [amount, memo]
        @current_total += amount
    end
    def subtract_money amount, memo = ""
        @ledger[@ledger.count] = [amount * -1, memo]
        @current_total -= amount
    end
```

Ruby 101

Ruby Basics:

```
    def printout
       tab = 0
       puts "Amount:\tMemo:\tTotal:"
       @ledger.each do |line|
          tab += line[0]
          puts "#{line[0]}\t#{line[1]}\t#{tab}"
       end
    end
end
```