Lab session 4 – Ruby (part 2)

Unit	Programming languages: principles and design (6G6Z1110) Programming languages – SE frameworks (6G6Z1115)
Lecturer	Rob Frampton
Week	4
Portfolio element	This lab is not part of the portfolio.

Description

The aim of this lab exercise is to practise the several Ruby elements as studied in the lecture “Ruby – part 2”. By the end of this lab session, you should how to write classes in Ruby with methods, accessors and inheritance. Everything you need will be in the lecture slides.

Exercises

Exercise 1:

Implement a class called `Employee`. The class should have two instance variables (using Ruby’s `attr` shortcuts): `name`, which can be written and read publicly, and `id`, which is read-only. The class should also have an `initialize` method taking two arguments `id` and `name` which initialize the two instance variables. If `name` is not provided, it defaults to “John Smith”.

You can test your class by observing the output in `irb`:

```
irb(main):001:0> load 'lab5_ex1.rb'
=> true
irb(main):002:0> e = Employee.new('1234')
=> #<Employee:0x00000000034a0710 @id="1234", @name="John Smith">
irb(main):003:0> e = Employee.new('1234', 'Syd Barrett')
=> #<Employee:0x000000000323b790 @id="1234", @name="Syd Barrett">
irb(main):004:0> e.name = "David Gilmour"
=> "David Gilmour"
irb(main):005:0> e.id = '2345'
NoMethodError: undefined method `id=' for
#<Employee:0x000000000323b790 @id="1234", @name="David Gilmour">
Did you mean?  id
        from (irb):5
        from C:/Program Files/Ruby24-x64/bin/irb.cmd:19:in `<main>'
```

Exercise 2:

Add a `getSigniture` method to your class which returns a string with their name and id in the following format:

`Employee ID` *`<id>`*`:` *`<name>`*

For example:

```
irb(main):001:0> load 'lab5_ex2.rb'
=> true
irb(main):002:0> e = Employee.new('2345', 'Anna Calvi')
=> #<Employee:0x00000000385e4d8 @id="2345", @name="Anna Calvi">
irb(main):003:0> e.getSigniture
=> "Employee 2345: Anna Calvi"
```

Exercise 3:

Implement a class called `Developer` which extends `Employee`. Override the `getSigniture` method so that it calls the original implementation, but adds the string "`, Developer`" to the end and returns that.

For example:

```
irb(main):001:0> load 'lab5_ex3.rb'
=> true
irb(main):002:0> d = Developer.new('3456', 'Jon Boden')
=> #<Developer:0x0000000003244200 @id="3456", @name="Jon Boden">
irb(main):003:0> d.getSigniture
=> "Employee 3456: Jon Boden, Developer"
```

Exercise 4:

Make `getSigniture` in `Employee` and `Developer` private, and implement a new method `printSigniture` which calls `getSigniture` internally and prints it to the console.

For example:

```
irb(main):001:0> load 'lab5_ex4.rb'
=> true
irb(main):002:0> d = Developer.new('4567', 'Owen Brinley')
=> #<Developer:0x00000000036f17d0 @id="4567", @name="Owen Brinley">
irb(main):003:0> d.getSigniture
NoMethodError: private method `getSigniture' called for
#<Developer:0x00000000036f17d0 @id="4567", @name="Owen Brinley">
        from (irb):3
        from C:/Program Files/Ruby24-x64/bin/irb.cmd:19:in `<main>'
irb(main):004:0> d.printSigniture
Employee 4567: Owen Brinley, Developer
=> nil
```