Inheritance, Modules and Mixins

León Jaramillo

softserve

Requiring Files

- Often, we need to reuse code developed with that purpose.

- It is about using classes and methods grouped in frameworks and libraries which are coded in different files.

- To use code written in another file, we should require it. There are two ways:
  - Requiring the file providing its absolute path:

  ```
  require [Absolute path]
  ```

  - Requiring the file providing its relative path:

  ```
  require_relative [Relative path]
  ```

- When you write code intended to be reused from other files (requiring it), it is useful to implement it within a module.

Modules

- Modules is **a way to group** classes, methods and constants.
- We can use them to build **toolboxes**, **libraries** and similar stuff.
- A module defines, automatically, a **namespace**, which prevents name clashes.
- A module has **no** instances nor subclasses.

- We can define a module as follows:

```
module ModuleName
   [Statements]
end
```

- Then, we can use module's **members** as follows. That member could be a class, a method or a constant:

```
ModuleName.member
```

- Also, we can **include** a module within a class:

```
class OneClass
   include ModuleName
   ...
end
```

Inheritance

- Ruby supports **(simple) inheritance**.
- Inheritance allows a class (subclass) to **inherit behavior and characteristics** from another class (superclass).
- Hence, the subclass inherits the **attributes** and the **methods** of the superclass.
- We can **override** a method from the superclass in the subclass, similarly as in other languages.
- To use inheritance, we use < as follows:

```
class SubClass < ExistingSuperClass
...
end
```

Mixins

- Mixins allow us to **simulate multiple inheritance**.
- It consists not in inheriting from another class, but in **including one or more modules**.
- **Including multiple modules**, is like inheriting the behavior of multiple classes.
- **Mixins** allow all of this, with the `include` statement.

softserve

Useful Resources

- Learning Ruby: From Zero to Hero: https://www.freecodecamp.org/news/learning-ruby-from-zero-to-hero-90ad4eecc82d/

- Ruby - Object Oriented: https://www.tutorialspoint.com/ruby/ruby_object_oriented.htm

- Implementing OOP concepts with Ruby: https://www.geeksforgeeks.org/object-oriented-programming-in-ruby-set-1/

softserve

Homework

Based on the `Student` class written in the previous homework, do the following:

1. Write a subclass that inherits from that class.
2. In this class add the attributes and methods that you desire.
3. Write a `Course` class, including its name, semester and any data that you'd like to include.
4. Make the new class able to have many courses as an attribute.
5. Test the classes creating different objects in different scenarios.

COLOMBIA DC

Thanks! Any question?

softserve