Universidad de La Laguna
E.S. Ingeniería y Tecnología
Sección: Ingeniería Informática
Área: Lenguajes y Sistemas Informáticos

Lenguajes y Paradigmas de Programación
Práctica de laboratorio **#12**

Diseñar e implementar en Ruby un programa que utilice el patrón concurrente *productor-consumidor*.

1. Implementar una clase `Naranjero`.

   - Esta clase debe contar con métodos que indiquen la `altura` del árbol, su `edad` y el número de naranjas que produce `contador`.
   - También debe tener un método que incremente la edad de árbol `uno_mas`. Además de incrementar la edad, cada año el árbol ha de crecer y hacerse más alto, en la cantidad que decida el programador, y después de un cierto número de años el árbol debe morir. Durante los primero años el árbol no debería producir fruta, pero sí debería hacerlo una vez trascurridos un cierto número de ellos, también fijados por el programdor. Se espera que un árbol más viejo produzca más que un árbol joven. Por supuesto, se debería ser capaz de contar el número de naranjas que produce el árbol cada año.
   - Finalmente, se ha implementar un método que permita recoger una naranja `recolectar_una`. Su invocación reduce el contador en uno y devuelve al recolector una cadena que dice lo *deliciosa* que estaba, o quizás sólo le dice que *no hay* más naranjas o que el árbol está *muerto*.

2. Implementar un *thread* para hacer crecer un narajero.

3. Implementar un *thread* para recolectar naranjas.

4. Utilizar la metodología de desarrollo dirigido por pruebas (*Test Driven Development - TDD*) y la herramienta ***RSpec***.

5. Repartir las tareas entre los miembros del **Equipo de Trabajo**.

   Utilizar la estructura del 'directorio de trabajo del equipo' generada con *Bundler* en prácticas anteriores.

   Todos los miembros del equipo, han de realizar al menos una confirmación e incorporarla al repositorio compartido.

6. La salida pude ser algo similar a lo siguiente:

```
$ ruby oranges.rb
Age increaser going to sleep for 2
Orange picker going to sleep for 0
Orange picker woke up after sleeping for 0
Sorry, no Oranges to pick
Age increaser woke up after sleeping for 2
Age increaser increased the age
Orange picker waiting patiently...
Age increaser going to sleep for 4
Orange picker going to sleep for 4
Orange picker woke up after sleeping for 4
Age increaser woke up after sleeping for 4
```

```
Sorry, no Oranges to pick
Age increaser increased the age
Orange picker waiting patiently...
Age increaser going to sleep for 2
Orange picker going to sleep for 0
Orange picker woke up after sleeping for 0
Sorry, no Oranges to pick
Age increaser woke up after sleeping for 2
Age increaser increased the age
Orange picker waiting patiently...
Orange picker going to sleep for 0
Age increaser going to sleep for 0
Orange picker woke up after sleeping for 0
Age increaser woke up after sleeping for 0
The Orange is delicious
Age increaser increased the age
Orange picker waiting patiently...
Age increaser going to sleep for 2
Orange picker going to sleep for 3
Age increaser woke up after sleeping for 2
Age increaser increased the age
Age increaser going to sleep for 1
Orange picker woke up after sleeping for 3
The Orange is delicious
Age increaser woke up after sleeping for 1
Age increaser increased the age
Orange picker waiting patiently...
Age increaser going to sleep for 2
Orange picker going to sleep for 3
Age increaser woke up after sleeping for 2
Age increaser increased the age
Age increaser going to sleep for 4
Orange picker woke up after sleeping for 3
Sorry, the Orange tree is dead
Age increaser woke up after sleeping for 4
```

Referencia

Learn to Program. Chris Pine. The Pragmatic Programmers. ISBN 978-1-93435-636-4. 2009