IIC3745 - Testing (2020 / II)

Actividad 2 - Cobertura en base a lógica

Fecha de entrega: Martes 6 de octubre, 23:59

Introducción

Luego de ayudar al equipo de desarrollo de *League of Legend* a mejorar la ~~frágil~~ seguridad de su aplicación, te has vuelto un/una astro del *testing*, y otras empresas quieren contar con tus servicios.

Una famosa empresa de imprenta llamada *ImprenTest* capta tu atención con una innovadora propuesta de *software*, la cual automáticamente asigna precios de encuadernación a *PDFs* que envían sus clientes en cantidades industriales a través de su plataforma online.

El problema es que con la creciente demanda de clientes están tan enfocados en mantener su plataforma con *zero-downtime* que no han tenido tiempo de hacer pruebas a su código. Es por esto que han acudido a tí y con gusto has aceptado ayudarlos.

Aplicación

El programa consiste en una única clase que representa el PDF a avaluar. Esta recibe los siguientes parámetros para crear una instancia:

- `pages_of_text`: número de hojas con una cara.
- `double_pages_images`: número de imágenes a color en doble cara.
- `images`: número de imágenes en blanco y negro.
- `is_hardcover`: si se desea impresión en tapa dura.
- `is_premium_book`: si el libro es de categoría premium[1].

Además, tiene implementados los siguientes métodos:

- `total_pages_images`: suma total de las páginas que incluyen imágenes.
- `total_pages`: suma total de las páginas del libro.
- `bind_book_price`: método central que evalúa al libro. Recibe un parámetro *booleano* que indica si el cliente es frecuente para hacerle un descuento.

[1] Un libro de categoría premium es aquel con hojas de mayor calidad, impresión láser, empastado con hilo, con cubreportada, etc.

Testing

Cobertura

Tu labor consiste en diseñar y programar los casos de prueba **mínimamente necesarios**[2] para asegurar el correcto funcionamiento de la aplicación bajo los siguientes criterios de cobertura:

- *Clause coverage (CC)*
- *Correlated active clause coverage (CACC)*

Análisis detallado de pruebas

Para cada uno de los dos criterios de cobertura anteriores, debes explicitar los *predicados y cláusulas*, listar los *test requirements* (*TR*) y especificar los *test cases* (*TC*) junto a los parámetros de *input* utilizados y el *output* esperado.

A modo de ejemplo, si el código entregado fuera:

```ruby
def sum_price(attribute_1, attribute_2)
  actual_price = 2000

  if attribute_1 == 2 or attribute_2 > 2
    actual_price += 1000
  end

  if attribute_1 == 1 and attribute_2 < 2
    actual_price += 3000
  end

  return actual_price
end
```

[2] Un test deja de ser mínimo si todos los *test requirements* (*TR*) cubiertos por dicho test son cubiertos en otros *test cases* (*TC*) propuestos. En otras palabras, si se elimina un *test case* (*TC*) y el coverage se mantiene, entonces no es un test mínimamente necesario para dicha cobertura.

El formato esperado (o similar) para el diseño de pruebas sería:

Los predicados son:
1. p1: `(attribute_1 == 2 or attribute_2 > 2)`
2. p2: `(attribute_1 == 1 and attribute_2 < 2)`

Las cláusulas son:
1. c1: `(attribute_1 == 2)`
2. c2: `(attribute_1 == 1)`
3. c3: `(attribute_2 > 2)`
4. c4: `(attribute_2 < 2)`

Los *test requirements* (*TR*) estarían condicionados al *coverage* que se está analizando, pero en caso del *predicate coverage* los *test requirements* (*TR*) serían:
1. TR1: p1 debe ser `True`
2. TR2: p1 debe ser `False`
3. TR3: p2 debe ser `True`
4. TR4: p2 debe ser `False`

Dado los *test requirements* (*TR*), los *test cases* (*TC*) serían:
1. Para cumplir con TR1 y TR4:
    a. Input: `attribute_1 = 2, attribute_2 = 3`
    b. Output: `3000`
2. para cumplir con TR2 y TR3
    a. Input: `attribute_1 = 1, attribute_2 = 1`
    b. Output: `5000`

SimpleCov

Finalmente, dado que en paralelo estás desarrollando el proyecto semestral (y quieres lucirte con *ImprenTest*), decides utilizar *SimpleCov* para *soltar la mano*. Utilizando dicha herramienta debes cumplir con 100% de cobertura en:

- *Line coverage*[3] *(LC).*
- *Branch coverage*[4] *(BC).*

NOTA: Para estos criterios no es necesario hacer el análisis detallado de la sección anterior, basta solamente con programar los casos de prueba mínimos para cada criterio.

[3] Equivalente a la cobertura de nodos basada en grafos (NC)
[4] Equivalente a la cobertura de aristas basada en grafos (EC) y a la cobertura de predicados basada en lógica (PC).

Requisitos técnicos

Debes utilizar el lenguaje de programación *ruby 2.7.1* en conjunto con las herramientas *RSpec* 3.9.0 y *SimpleCov* 0.19.0 para la implementación de tus casos de prueba. Para ello, debes especificar las dependencias de tu aplicación en un archivo *Gemfile* tal como se muestra en este repositorio.

Formato de entrega

Esta actividad puede ser desarrollada en parejas, pero está diseñada para ser lograda individualmente. Se abrirá un cuestionario en Canvas con fecha límite martes 6 de octubre a las 23:59 hrs para entregar tus respuestas, y otro con fecha límite para el día siguiente con un descuento de 2 puntos. Tú (o tu equipo) deberás subir un archivo comprimido *.zip* que incluya su archivo *Gemfile*, el código base `pdf_pricer.rb` y los siguientes archivos *.rb* adicionales:

- `pdf_pricer_cc_spec.rb` - archivo que contenga los casos de prueba que cumplen con el criterio *clause coverage.*
- `pdf_pricer_cacc_spec.rb` - archivo que contenga los casos de prueba que cumplen con el criterio *correlated active clause coverage.*
- `pdf_pricer_lc_spec.rb` - archivo que contenga los casos de prueba que cumplen con los criterios *code coverage* según *SimpleCov.*
- `pdf_pricer_bc_spec.rb` - archivo que contenga los casos de prueba que cumplen con los criterios *branch coverage* según *SimpleCov.*

Dentro del comprimido también debes incluir uno (o más) archivos *.md* con el detalle de los *predicados, cláusulas*, *TR*, *TC*, *inputs* y *outputs* para los primeros dos criterios de cobertura con el formato indicado anteriormente (o similar).

Finalmente, si trabajan en parejas, asegúrense de incluir los nombres, correos y números de alumno/a de ambos/as integrantes del grupo en al menos uno de los archivos *.md*. **Solamente un integrante debe subir la entrega por pareja**. En caso de que ambos/as lo suban, se tomará en cuenta aquella con la que el ayudante se haya encontrado primero y **no habrá opción de solicitar una corrección de otra versión**. Se aplicará un descuento de **5 décimas** a aquellos que no cumplan con las indicaciones para el formato de entrega.