CI-0126 Ingeniería de Software
Grupo 01 - I-2023

Prof. Dr. Allan Berrocal Rojas
allan.berrocal@ucr.ac.cr

Laboratorio 8
Pruebas Unitarias Simples

Contenidos

Prof. Dr. Allan Berrocal Rojas
allan.berrocal@ucr.ac.cr

1. Requisitos

1. Microsoft Visual studio 2019 o mayor instalado (cualquier versión eg. *Community*, *Enterprise)*
2. Cuenta en GitHub

2. Objetivos

1. Complementar los conceptos estudiados en clase sobre pruebas de software.
2. Aplicar los conceptos vistos en clase sobre la metodología *Test Driven Development* (TDD) a partir de un ejercicio práctico.
3. Aplicar los conceptos vistos en clase sobre la aplicación de pruebas unitarias a partir de un ejercicio práctico.

3. Parte 1 - TDD

Esta parte consiste en realizar un ejercicio donde aplica la metodología TDD para desarrollar un programa pequeño y agregar pruebas unitarias.

3.1. Tutorial

Abra el siguiente tutorial en un navegador web, *Walkthrough: Test-driven development using Test Explorer*. Siga las instrucciones de todos los pasos para desarrollar un pequeño programa que calcula la raíz cuadrada de un número.

3.2. Reporte

Cree un documento de texto para documentar el progreso y luego guárdelo en formato PDF con este nombre **`reporte_tdd.pdf`**. En el reporte agregue al menos dos screenshots que evidencien la realización de los principales pasos del tutorial, por ejemplo, mostrando los resultados de la ejecución de pruebas unitarias.

3.3. Preguntas

En su documento de reporte, conteste brevemente las siguientes preguntas.

A) De la sección, *Create a test and generate code*, explique muy brevemente ¿Por qué la prueba que ejecutó en el paso #6 falló?
B) De la sección, *Extend the range of inputs*, explique muy brevemente ¿Por qué la prueba que ejecutó en el paso #2 falló?

Prof. Dr. Allan Berrocal Rojas
allan.berrocal@ucr.ac.cr

4. Parte 2 - Pruebas Unitarias

Esta parte consiste en realizar un ejercicio donde aprenderá a crear pruebas unitarias en el ambiente de *Visual Studio Test Explorer*. Para ello va a crear una pequeña aplicación que simula procesos simples en un banco y creará pruebas unitarias para verificar el correcto funcionamiento del programa a nivel de clases y métodos.

4.1. Conceptos

Abra la siguiente página en un navegador web, *Unit tests basics* y lea con detalle hasta antes de la sección ***Q&A*** (esta sección es de lectura opcional pero recomendada).

4.2. Tutorial

Abra el siguiente tutorial en un navegador web, *Walkthrough: Create and run unit tests for managed code*. Siga las instrucciones de todos los pasos para desarrollar un pequeño programa que simula operaciones en un banco.

4.3. Reporte

Cree un documento de texto para documentar el progreso y luego guárdelo en formato PDF con este nombre **`reporte_bank.pdf`**. En el reporte agregue al menos dos screenshots que evidencien la realización de los principales pasos del tutorial, por ejemplo, mostrando los resultados de la ejecución de pruebas unitarias.

4.4. Preguntas

En su documento de reporte, conteste brevemente las siguientes preguntas.

A) Basado en el tutorial del ejercicio de pruebas unitarias, en la última sección, *Retest, rewrite, and reanalyze*, explique muy brevemente ¿Por qué se necesita agregar la instrucción **`Assert.Fail`** a este caso de prueba?
B) Basado en la lectura de conceptos sobre pruebas unitarias, responda muy brevemente ¿cuál es el valor de las pruebas unitarias en el flujo de desarrollo de software?
C) Basado en la lectura de conceptos sobre pruebas unitarias, responda muy brevemente ¿cuáles son las principales partes que componen una prueba unitaria?
D) Basado en la lectura de conceptos sobre pruebas unitarias, responda muy brevemente ¿Para qué sirve establecer un ***timeout*** a un caso de prueba?

Prof. Dr. Allan Berrocal Rojas
allan.berrocal@ucr.ac.cr

5. Entregables del laboratorio

1% Todos los entregables de este laboratorio deben estar en su repositorio de GitHub en un directorio con nombre **laboratorio8**.

28% En su repositorio de Git, en el directorio llamado **`laboratorio8`** deben aparecer dos subdirectorios internos:

1. (14%) En el primer subdirectorio (**`reporte_tdd`**) debe tener el proyecto creado en el tutorial que realizó en la parte 1 de este laboratorio.
2. (14%) Y el segundo subdirectorio (**`reporte_bank`**) debe tener el proyecto que realizó en la parte 2.

2% Cada subdirectorio mencionado anteriormente debe tener un documento de reporte en formato PDF con el nombre que se le solicitó.

26% Contenido del **`reporte_tdd.pdf`**

1. (10%) Al menos 2 screenshots solicitados
2. (8%) Respuesta a la pregunta A) ¿Por qué la prueba del paso #6 falló?
3. (8%) Respuesta a la pregunta B) ¿Por qué la prueba del paso #2 falló?

34% Contenido del **`reporte_bank.pdf`**

1. (10%) Al menos 2 screenshots solicitados
2. (6%) Respuesta a la pregunta A) ¿Por qué se necesita agregar la instrucción **`Assert.Fail`** a este caso de prueba?
3. (6%) Respuesta a la pregunta B) ¿Cuál es el valor de las pruebas unitarias en el flujo de desarrollo de software?
4. (6%) Respuesta a la pregunta C) ¿cuáles son las principales partes que componen una prueba unitaria?.
5. (6%) Respuesta a la pregunta D) ¿Para qué sirve establecer un ***timeout*** a un caso de prueba?

1% Agregue al reporte el enlace a su repositorio de GitHub.

8% Agregue un pequeño resumen, no más de 200 palabras indicando claramente

1. Tres cosas que no sabía y aprendió en el laboratorio
2. Una cosa que se le hizo difícil de realizar y explique por qué fue difícil.
3. Una cosa que se le hizo fácil de realizar y explique por qué fue fácil.
4. Indique cuánto tiempo tardó en realizar el laboratorio.