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

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

Laboratorio 7[1]
Clean Code

Contenidos

[1] Basado en material previamente desarrollado en cursos de Gustavo López y Rebeca Obando en la Universidad de Costa Rica

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. Aplicar los conceptos estudiados en clase sobre Código Limpio o *Clean Code*.
2. Identificar segmentos de código problemáticos en un programa que no cumplan con ciertos principios de código limpio.
3. Proponer soluciones a los problemas encontrados mediante la aplicación de principios de código limpio.

3. Parte I - material base del laboratorio

3.1. Programa base

Para este laboratorio se le brinda un proyecto en C# con diversas funcionalidades que servirá como base para que realice la revisión de código. Asegúrese de visualizar el programa y comprender las funcionalidades que este tiene.

3.2. Guía de identificación de problemas

Para identificar violaciones a los principios de código limpio puede utilizar el material que se le dió en clase. Además, puede utilizar la guía Clean Code .NET que tiene principios de código limpio aplicable al lenguaje C#. Este material fue escrito y publicado por Tang Chung de Vietnam y está inspirado en el libro *Clean Code: A Handbook of Agile Software Craftsmanship* de Robert C. Martin.

4. Parte II - Aplicación de código limpio

4.1. Artefacto a utilizar

Usted va a inspeccionar los archivos de código que contiene el proyecto y reportar los problemas que encuentre. Para ello, debe crear un documento tipo tabla para reportar los hallazgos de manera ordenada y clara.

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

Es importante que un reporte de problemas de código tenga todos los elementos para que una persona pueda fácilmente ubicar el problema. Algunos ejemplos sobre elementos para ubicar un problema en el código son:

- Nombre del paquete o módulo
- Nombre del archivo
- Nombre de la clase
- Nombre del método
- Número(s) de línea(s) de código afectados
- Alguna otra referencia de ser necesario

Para este laboratorio usted puede usar la siguiente tabla como ejemplo para cada archivo a revisar. Si lo prefiere puede crear una propia.

Archivo:	Hangman.cs	
Método	**# Línea(s)**	**Tipo de error y descripción**
PlayHangman	3	Mala indentación

4.2. Revisión de código

Utilizando las guías de código limpio, haga una revisión de código del proyecto que se le dio. Cree una tabla de reporte para cada uno de los 7 archivos de código fuente y reporte ahí al menos 7 problemas de distinto tipo para cada archivo. Si no encuentra 7 problemas para un archivo, puede compensar reportando en otro de los archivos la cantidad de problemas que le faltó.

5. Solución de problemas

De los errores que detectó, seleccione **5 de diferente tipo** y proponga la solución o corrección al problema. En cada caso explique con claridad a qué se debe el error y por qué es importante de corregir. Explique su solución y justifique por qué dicha solución es adecuada.

En las correcciones, agregue segmentos de código mostrando la versión anterior (con el problema) y la versión corregida con la solución propuesta. Estos segmentos de código pueden ser screenshots en su documento de reporte o, si lo prefiere, agregue archivos de código fuente como adjunto a su reporte (en este caso su entrega se llamará **ReporteLab7_CARNE.zip**.

6. Entregables del laboratorio

0.25%	Todos los entregables de este laboratorio deben estar en su repositorio de GitHub en un directorio con nombre **laboratorio7**.
0.25%	Cree un documento ordenado con su nombre y número de carné. Utilice el nombre **ReporteLab7_CARNE.(pdf|zip)** (su número de carné). No entregue archivos en Word o en ningún otro formato editable. Utilice PDF.
42%	Tablas con los reportes para los 7 archivos de código a revisar. En cada tabla se debe poder identificar para cada reporte o defecto: 1. (1pt) Archivo. 2. (1pt) Línea(s) de código afectada(s). 3. (1pt) Tipo o categoría del problema. 4. (2pt) Descripción del problema (cuando no sea evidente a partir de su tipo). (1pt) Para cada archivo deben reportar al menos 7 problemas (o bien compensar los faltantes con otros archivos)
50%	Soluciones a 5 problemas encontrados durante la revisión de código. En las soluciones se debe poder identificar: 1. (2pt) Problema que está corrigiendo (de los identificados en las tablas del punto anterior). 2. (2pt) Segmento de código o screenshot que muestre la versión con el problema. 3. (2pt) Breve explicación sobre a qué se debe el error y por qué es importante de corregir. 4. (2pt) Segmento de código o screenshot que muestre la versión corregida. 5. (2pt) Breve explicación de su solución y justificación de por qué dicha solución es adecuada.
0.5%	Agregue al reporte el enlace a su repositorio de GitHub.
7%	Agregue un pequeño resumen, no más de 200 palabras indicando claramente 1. Dos 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.