|  |
| --- |
| **Fundamentos de Ing. De Software** |
| Carrera: Inteligencia Artificial | Fecha Actualización: 20 de abril de 2024 |

**PUNTAJES Y NOTA**

|  |  |
| --- | --- |
| **Puntaje máximo: 100** | **Nota:** |
| **Puntaje mínimo de aprobación: 60** |

**Nombre Completo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**C.I. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Fecha: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**INSTRUCCIONES GENERALES:**

|  |
| --- |
| Objetivo:  Desarrollar una aplicación de consola en C# que funcione como calculadora para resolver diversos problemas de física. La aplicación deberá solicitar al usuario qué cálculos desea realizar y continuará ejecutándose hasta que el usuario decida terminar. Se debe utilizar una biblioteca de enlace dinámico (DLL) que contenga las funciones necesarias para los cálculos, y esta DLL debe ser llamada desde el proyecto de consola.  Requisitos del Proyecto:   1. Funcionalidad de la Calculadora:    * La calculadora debe ofrecer las siguientes funciones:      + Calcular la fuerza dados masa y aceleración (F = m \* a).      + Calcular el trabajo dado por fuerza y distancia (W = F \* d).      + Calcular la energía cinética dados masa y velocidad (E = 0.5 \* m \* v^2).    * Debe incluir una opción para terminar el programa. 2. Interfaz de Usuario:    * La aplicación debe preguntar al usuario qué fórmula desea calcular.    * Debe permitir al usuario ingresar los valores necesarios para cada fórmula.    * Debe mostrar el resultado y luego preguntar si desea realizar otro cálculo. 3. Estructura del Proyecto:    * El código debe estar organizado en clases y métodos adecuados para la separación de la lógica de la interfaz de usuario y la lógica de cálculo.    * Las funciones de cálculo deben estar contenidas en una DLL separada, que será llamada desde el proyecto de consola.    * Incluir comentarios adecuados en el código para explicar la funcionalidad. 4. Control de Versiones:    * Crear un repositorio en GitHub para el proyecto.    * Mantener un registro detallado de cambios con commits claros y descriptivos.    * Los commits deben reflejar una historia coherente de desarrollo del proyecto. 5. Documentación:    * Incluir un archivo README.md en el repositorio que explique el propósito de la aplicación, cómo ejecutarla y ejemplos de uso.    * Documentar cada función/método en el código, tanto en la DLL como en el proyecto de consola. 6. Pruebas Unitarias:    * Escribir pruebas unitarias para cada función de cálculo en la DLL.    * Las pruebas deben cubrir una variedad de casos, incluyendo entradas válidas e inválidas. |

**FECHA DE ENTREGA:**

- **1ra Instancia,** Martes 28 de Mayo de 2024

- **2da Instancia,** Miércoles 29 de mayo de 2024

El encargo debe ser entregado en la fecha y el horario de clases.

Entregas posteriores, incompletas o atrasadas no serán revisadas y se calificarán con nota 1.0.

**Rúbrica de Evaluación para el Examen de Calculadora de Física**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criterio | Excelente | Bueno | Suficiente | Insuficiente |
| Correctitud del código | Funciona correctamente, sin errores, realiza todos los cálculos especificados utilizando la DLL (40 puntos). | Funciona con pequeños errores que no afectan críticamente la funcionalidad, utiliza la DLL (30 puntos). | Funciona, pero con errores significativos que afectan algunos cálculos, utiliza parcialmente la DLL (20 puntos). | No funciona, los errores impiden realizar los cálculos o no utiliza la DLL (10 puntos). |
| Organización y estructura | Código bien organizado en clases y métodos, con una clara separación de responsabilidades entre la DLL y el proyecto de consola (20 puntos). | Estructura adecuada pero con algunos aspectos de la organización que podrían mejorar, separación parcial entre la DLL y el proyecto de consola (15 puntos). | Estructura básica cumplida, pero con mezcla de responsabilidades entre la DLL y el proyecto de consola (10 puntos). | Estructura pobre o inexistente, sin separación clara entre la DLL y el proyecto de consola (5 puntos). |
| Uso de control de versiones | Uso excelente de Git. Historial de commits detallado y descriptivo que refleja un desarrollo iterativo (20 puntos). | Buen uso de Git. Historial de commits bastante claro y mayormente descriptivo (15 puntos). | Uso básico de Git. Historial de commits no muy descriptivo o poco frecuente (10 puntos). | Uso inadecuado de Git. Historial de commits escaso o no descriptivo (5 puntos). |
| Pruebas Unitarias | Pruebas unitarias completas y detalladas para las funciones de la DLL que cubren todos los casos, incluyendo entradas válidas e inválidas (20 puntos). | Pruebas unitarias adecuadas para las funciones de la DLL pero podrían cubrir más casos o ser más detalladas (15 puntos). | Pruebas unitarias para las funciones de la DLL presentes pero falta cobertura de casos o detalles importantes (10 puntos). | Pruebas unitarias para las funciones de la DLL incompletas, mal realizadas o ausentes (5 puntos). |