|  |
| --- |
| **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: ERWIN ALEJANDRO OJEDA JUSTINIANO**

**C.I.: 13976731 Fecha: 25/04/2024**

<https://github.com/OjedaErwin123/Examen2.git>

**INSTRUCCIONES GENERALES:**

|  |
| --- |
| **Examen de gestión de Código fuente: Calculadora de Física para MRU y MRUA**  **Objetivo:**  Desarrollar una aplicación de consola en C# que funcione como calculadora para resolver problemas de Movimiento Rectilíneo Uniforme (MRU) y Movimiento Rectilíneo Uniformemente Acelerado (MRUA). La aplicación deberá solicitar al usuario qué cálculos desea realizar y continuará ejecutándose hasta que el usuario decida terminar.  **Requisitos del Proyecto:**   1. **Funcionalidad de la Calculadora**:    * La calculadora debe ofrecer al menos las siguientes funciones para MRU:      + Calcular la distancia (**d = v \* t**).      + Calcular la velocidad (**v = d / t**).      + Calcular el tiempo (**t = d / v**).    * Para MRUA, debe ofrecer las siguientes funciones:      + Calcular la posición final (**x = x0 + v0 \* t + 0.5 \* a \* t^2**).      + Calcular la velocidad final (**v = v0 + a \* t**).      + Calcular el tiempo (**t = (v - v0) / a**).    * 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.    * 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.    * Incluir la(s) historia(s) de usuario generadas en markdown en el repositorio de git y github en una carpeta independiente del código fuente.    * Documentar cada función/método en el código. 6. **Historias de Usuario**:    * Redactar al menos dos historias de usuario que describan los requerimientos y expectativas del usuario final respecto a la funcionalidad de la calculadora.   **DETALLES DE ENTREGA:**   * Formato de entrega: Enlace al repositorio de GitHub. |

A partir de la entrega del planteamiento de este Proyecto final de la asignatura, **usted dispone de 2 días** para desarrollar el encargo.

**FECHA DE ENTREGA:**

- **1ra Instancia,** Jueves 25 de abril de 2024

- **2da Instancia,** Martes 30 de abril 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 (40 puntos). | Funciona con pequeños errores que no afectan críticamente la funcionalidad (30 puntos). | Funciona, pero con errores significativos que afectan algunos cálculos (20 puntos). | No funciona o los errores impiden realizar los cálculos (10 puntos). |
| Organización y estructura | Código bien organizado en clases y métodos, con una clara separación de responsabilidades (20 puntos). | Estructura adecuada pero con algunos aspectos de la organización que podrían mejorar (15 puntos). | Estructura básica cumplida, pero con mezcla de responsabilidades en clases y métodos (10 puntos). | Estructura pobre o inexistente de clases y métodos (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). |
| Calidad de la documentación | Documentación completa, incluye README.md con descripciones claras, ejemplos de uso, y todos los métodos están documentados (10 puntos). | Documentación buena pero falta detalle en algunos métodos o en el uso de la aplicación (7.5 puntos). | Documentación presente pero solo cumple con los requisitos mínimos. Carece de detalles importantes (5 puntos). | Documentación escasa o ausente (2.5 puntos). |
| Historias de usuario en Markdown | Historias de usuario completas y detalladas en Markdown, ubicadas en una carpeta dedicada, reflejan claramente las necesidades y expectativas del usuario (10 puntos). | Historias de usuario en Markdown bien realizadas pero podrían ser más detalladas o claras (7.5 puntos). | Historias de usuario cumplen con formato pero faltan detalles importantes o claridad (5 puntos). | Historias de usuario incompletas, mal realizadas o en formato incorrecto (2.5 puntos). |