SCRIPTING
INGENIERÍA EN DISEÑO DE ENTRETENIMIENTO DIGITAL
UNIVERSIDAD PONTIFICIA BOLIVARIANA

PARCIAL PRÁCTICO 2

FECHA: Septiembre 30 de 2023 **PROFESOR:** Andrés Pérez Campillo

NOTAS PREVIAS:

- Por favor **LEA TODO EL ENUNCIADO** antes de responder cualquier pregunta. Si existe alguna imprecisión con la redacción que pueda sugerir ambigüedad en la pregunta o si el enunciado contiene preguntas sobre temas no tratados en el curso o sus prerrequistos, hacer la observación pública y el profesor la atenderá.
- La fecha límite para hacer subidas al repositorio es el **30 de septiembre de 2023 a las 11:59 hrs.**
- **Usted debe crear el código completamente. Debe crear un repositorio en GitHub donde aloje un proyecto de pruebas de NUnit en Visual Studio con el código que implemente este parcial.**
- **La entrega se realizará vía Aula Digital, y bajo ningún concepto se aceptará entrega por otro medio. Si presenta la entrega por fuera del Aula Digital, se asumirá como NO ENTREGADA.**
- Incluya en su solución un archivo *readme*. Detalle los miembros que participaron de la entrega. Si usted trabaja en pareja y no hace referencia a su compañero, solo se le calificará a quien figure en la entrega.
- Puede consultar las notas de clase, material del curso y el manual de referencia de *C#*, o sitios como *dotnetpealrs*, *codeproject*, foros técnicos, etc.
- **ESTÁ PROHIBIDO** el uso de cualquier soporte no autorizado por el profesor.
- **La actividad se entregará de manera INDIVIDUAL o GRUPAL POR PAREJAS**. **Colaborar con algún compañero de clase fuera de su grupo de trabajo, comunicarse por cualquier medio con algún tercero ajeno al curso, solicitar ayuda en foros técnicos durante el tiempo de evaluación está incluido en los soportes NO AUTORIZADOS por el profesor y se constituye en caso de fraude.**
- En caso de fraude, se anulará el examen asignando una nota de **CERO PUNTO CERO (0.0)** y se procederá de acuerdo al Reglamento Estudiantil de Pregrado de la Universidad.

COMPONENTE PRÁCTICO (5.0)

El Jugón SAS está en etapa de diseño de su próximo juego, y se le ha encargado a usted implementar una propuesta para prueba de concepto. En esta ocasión, la prueba de concepto consiste en un juego de carreras de motos que se centrará en la personalización de los componentes para mejorar los parámetros del vehículo y como estrategia de monetización.

En este momento, se le ha pedido implementar el entorno de pruebas unitarias para diversas funcionalidades del juego. De momento, nos centraremos en las características de las motos.

Los siguientes son los requisitos funcionales que debe probar:

- Cada moto *Bike* tiene una serie de componentes que se pueden personalizar:
    - *FrontWheel* - Llanta delantera.
    - *BackWheel* - Llanta trasera.
    - *Chassis* - Chasis
    - *Engine* - Motor
    - *Muffler* - Mofle
- Las motos *deben* tener todas esas partes para poder ser usadas en una carrera.
- Las motos tienen los siguientes parámetros:
    - *Speed:* Velocidad máxima de la moto
    - *Acceleration:* Factor de aceleración de la moto
    - *Handling:* Facilidad para maniobrar la moto.
    - *Grip*: Agarre del vehículo en superficies.
- En ningún caso, una moto puede no tener chasis. Al tiempo, no puede existir ningún chasis fuera de una moto.
- Los valores de cada parámetro estarán dados en una escala análoga de 0.0 a 10.0, donde 0.0 representa el menor valor y 10.0 el máximo.
- En ningún caso, una moto completa puede tener ninguno de sus parámetros en 0.0.
- Cuando una moto está completa, el valor inicial de todos sus parámetros es 1.0
- En caso de faltar algún componente, se deben satisfacer las siguientes reglas:
    - Falta motor: *Speed* será 0.0
    - Falta cualquiera de las llantas: *Acceleration* será 0.0
    - Faltan *ambas* llantas: *Handling* será 0.0, *Grip* será 0.0
    - Falta mofle: *Acceleration* se divide a la mitad.
- Las partes pueden modificar uno o más parámetros de la moto, pero se excluyen algunos según la parte - esto es, las partes *no* modifican parámetros según sigue:
    - El motor *no* modificará nunca *Grip*.
    - El mofle *no* modificará nunca *Speed*, *Handling*, *Grip*
    - El chasis *no* modifica ningún parámetro.
    - Las llantas pueden modificar todos los parámetros.
- Los valores que agrega cada parte a cada parámetro no pueden ser superiores a 5.0, y, en total no puede agregar más de 10.0 puntos sumando todos los parámetros que modifique.
    - Durante la construcción de cada parte se debe garantizar que la regla anterior se satisfaga.

- Una vez que la moto esté completa, al equipar una parte nueva, la parte nueva debe reemplazar a la anterior.
- El total de cada parámetro será la suma del valor que cada parte agregue a dicho parámetro más el valor inicial por estar completa (ej. *Speed* = 1.0 + *frontWheelSpeed* + *backWheelSpeed* + *engineSpeed*).

Su implementación debe ser entregada en un proyecto de pruebas y cumplir con las siguientes características:

1. Código escrito en inglés **(0.25)**
2. *Readme* **(0.25)**
    a. Incluye los miembros del equipo, si es más de uno.
    b. Incluye consideraciones de su diseño orientado a objetos.
    c. Incluye consideraciones o particularidades de su proyecto.
3. **Diseño Orientado a Objetos (1.0):**
    a. Debe identificar clases candidatas a superclases, subclases, potenciales interfaces o clases abstractas. En caso de no usarlas, justificar el por qué.
    b. Reflejar relaciones de composición o agregación en la instanciación de clases.
    Pista: Para este ejercicio se le pide pensar una arquitectura extensible a futuro, por lo que debe aplicar todos los principios de la OOP.
4. **Diseño Dirigido por Pruebas (TDD) (3.5):**
    a. Las pruebas *deben* correr y pasar completamente **(2.0 / 3.5)**
    b. Debe configurar el entorno de pruebas para que cada prueba unitaria pueda ser ejecutada individual e independientemente.
    c. Usted debe escribir los objetos y datos de prueba, inicializarlos y limpiarlos al ejecutar *cada prueba*.
    d. Probar los siguientes escenarios:
        i. **BikeCanBeCreated:** Verifica que una moto pueda ser creada correctamente. Debe intentar crear una moto con y una moto sin chasis. Ambas deben ser válidas.
        ii. **BikesCanBeUsed**: Verifica que una selección de motos puede ser usada o no en una carrera.
            a. Una moto con todas sus partes puede ser usada
            b. Una moto sin motor no puede ser usada
            c. Una moto sin una llanta no puede ser usada
            d. Una moto sin mofle no puede ser usada
        iii. **PartCanBeAdded:** Verifica que una selección de partes pueden ser agregadas a una selección de motos según las siguientes condiciones:
            a. Una moto sin llanta delantera puede equiparse una llanta delantera sin problemas.
            b. Una moto con llanta trasera puede equiparse la llanta trasera nueva reemplazando la anterior.
            c. Una moto sin mofle puede equiparse el mofle.
            d. Una moto no puede reemplazar el chasis.
        iv. **PartsModifyParameters**: Verifica que la adición de partes de una moto modifique los parámetros según se especifica:

a. Crear una moto sin partes, probar el valor inicial de los parámetros según la descripción entregada
b. Crear partes con modificación de parámetros 0.
c. Agregar parte a parte, probar los valores de cada parámetro según la descripción entregada arriba y cada parte agregada (ej. Agregar el motor hace que el valor de Speed deje de ser 0.0 si este motor modifica el Speed).
d. Al completar todas las partes, verificar que todos los parámetros satisfagan los valores según la descripción entregada.
e. Repetir esta prueba con partes que modifiquen parámetros con valores a su discreción, y probar que los valores de los parámetros con partes mejoradas agreguen al total según la descripción de arriba).

v. **MaxParameterValueOnConstructor:** Verifica que el valor que agrega cada parte a la moto se inicialice correctamente al crear una instancia de cada parte y se satisfaga la descripción entregada.