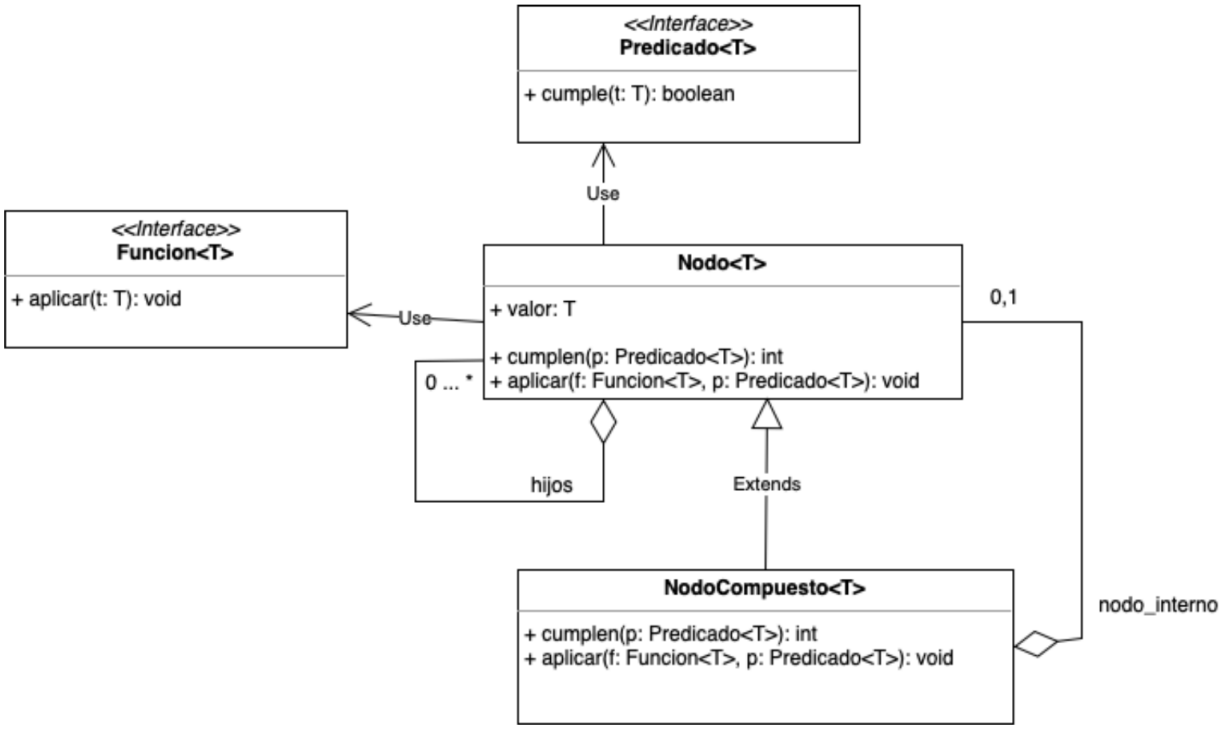
Desarrollo de Software
Prof. Carlos Alonzo
Primer Examen Parcial
18-10-2022 **Nombre:** _____

El examen es individual. Lea con detenimiento y tranquilidad cada uno de los enunciados. Justifique sus respuestas de manera clara y resumida.

1 Pregunta 1 (4 puntos)

En el Desarrollo de Software, una de las cualidades deseadas (y también un requerimiento no funcional) es la ***Reusabilidad***. Existen diversas técnias para alcanzar la reusabilidad de código o de un diseño, una de ellas es la ***Genericidad*** o ***Programación Genérica***.

Lea con detenimiento el siguiente fragmento de diseño que se le presenta en un **Diagrama de Clases UML**.



El dominio de la programación genérica es fundamental, ya que hoy en día está difundida tanto en el desarrollo de software de *backend* como de aplicaciones móviles o web.
Por tal motivo, usted debe demostrar que domina la genericidad. Para ello, usted debe implementar en **TypeScript** el método ***cumple*** de las clases genéricas Nodo y NodoCompuesto. Este método se encarga de calcular todos los objetos de tipo Nodo que cumplen con el predicado ***p***.

Desarrollo de Software
Primer Examen Parcial

2 Pregunta 2 (4 puntos)

En el Desarrollo de Software Orientado a Objetos, un consejo y una buena práctica ampliamente difundida es ***Evitar el uso de sentencias IF-ELSE o SWITCH-CASE***. Sabemos que el uso de estas sentencias, por lo general, no favorecen el cumplimiento del ***Principio Abierto-Cerrado (OCP)*** de los ***Principios SOLID***.

Usted debe seleccionar el **Patrón de Diseño** que favorece el cumplimiento de esta buena práctica. Debe explicar y justificar su respuesta sin ambiguedades, de manera concreta y resumida, indicando cómo se cumple el **Principio OCP** en dicho patrón.

3 Pregunta 3 (6 puntos)

A continuación se describe un problema de la vida real. Lea con detenimiento lo que se plantea. Usted debe modelar su diseño elaborando un **Diagrama de Clases UML** y debe justificar con claridad el **Patrón de Diseño** que empleará para dar solución al problema planteado:

1. Se desea implementar un sistema de archivos simple en forma de árbol (similar a Windows o LINUX) para un sistema operativo
2. Usted está tomando varias decisiones de diseño
3. Decisión 1: Todo el sistema de archivos está representado por un objeto de sistema de archivos que contiene una referencia a la carpeta padre
4. Decisión 2: Una carpeta contiene dos tipos de elementos: archivos y carpetas

4 Pregunta 4 (6 puntos)

El objetivo de este ejercicio es que usted construya un diseño más complejo aplicando varios **Patrones de Diseño**, lo que algunos autores llaman **Composición de Patrones**.

A continuación se detalla el sistema que usted debe diseñar:

1. Considere que se desea construir un sistema de creación de interfaces de usuario (UI) donde pueden existir distintos tipos de **Componentes**
2. Cada componente tiene un método **build** con el cual se construye

3. Un componente puede estar compuesto, a su vez, por otros componentes, formando así una estructura de componentes anidados

4. A los componentes se les puede aplicar **Decoradores**. De manera tal que un decorador, siempre ejecuta primero el método **build** del componente que decora y luego su propia lógica particular (la del decorador)

5. Los componentes pueden cambiar de estado, es decir, puede cambiar algún atributo que forma su estado.

6. Cuando el estado de un componente cambia, entonces el componente se encarga de notificar a los objetos **Observadores** que puedan estar interesados en el cambio de estado del componente

Usted debe construir el **Diagrama de Clases UML** de su diseño, indicando con claridad los **patrones de diseño** que ha empleado. Recuerde que debe argumentar su respuesta codificando (en TypeScript) los principales métodos de su diseño. Recuerde que la mejor manera de demostrar su dominio sobre la aplicación de los patrones, es a través del código de los métodos involucrados en dichos patrones.

5 BONUS (debe responder las 4 preguntas anteriores)

Explique cómo se relaciona el **Principio de Inversión de Dependencias** con los patrones de diseño **Abstract Factory** y **Factory Method**. Justifique por qué estos patrones favorecen que el diseño sea flexible y más desacoplado.