UNIVERSIDAD NACIONAL DE ENTRE RÍOS

Facultad de Ingeniería

ALGORITMOS Y ESTRUCTURAS DE DATOS

Guía de Práctica Nº 1

Clases y Testing

Versión 1.1

Objetivos

- Aplicar manejo de excepciones para validación de datos.
- Implementar clases elementales.
- Procesar datos desde archivos.
- Realizar pruebas unitarias para testear los métodos de clases.

Ejercicios

1. Cuadrado

Implementar una clase que modele a un cuadrado. El único atributo que posee es su lado. Dotar de operaciones al Cuadrado para ingresar el lado, mostrarlo, calcular y mostrar el área, calcular y mostrar el perímetro.

Evaluar los métodos donde es necesario hacer una validación de datos e implementar.

2. Punto

Implementar una clase que modele un punto compuesto por dos coordenadas ‘x’ y ‘y’. El punto debe contar con operaciones para: proporcionar un valor al punto al momento de instanciarlo, proporcionar valores a sus atributos y para retornarlos.

Evaluar los métodos donde es necesario hacer una validación de datos e implementar.

3. Persona

Implementar la clase Persona. Una persona posee un nombre y un apellido. La clase debe asegurarse que sus campos estén capitalizados y los mismos (campos o atributos) deben estar “ocultos”.

Añadir métodos setters y getters. Opcional: Utilizar propiedades (*property*) para ocultar setters y getters.

Crear una unidad de prueba para validar tipo de datos.

4. Estudiante

Implementar una clase Estudiante, para almacenar sus datos personales (legajo, apellido y nombre, documento y promedio).

En un archivo de texto están guardados los datos de los estudiantes de una escuela. Cada línea posee los datos de un estudiante separados por comas. Leer el archivo, agregar cada registro en un objeto de tipo Estudiante, almacenarlos en una lista y mostrar los datos ordenados por legajo. (**Utilizar listas de python**)

Evaluar los métodos donde es necesario hacer una validación de datos e implementar.

5. Persona con datos aleatorios

Crear la clase PersonaAleatoria realizando una herencia desde la clase Persona de modo tal que sus datos se generen aleatoriamente. Para esto, los datos pueden ser tomados desde diferentes fuentes como:

- Archivo de texto con lista de nombres y apellidos.
- Una API web.
- La consola de comandos (no recomendado).

6. Producto

Implementar una clase que modele un producto. El producto debe tener como atributos: nombre, precio, y número de unidades (cantidad). Debe contar con operaciones para establecer los valores de sus atributos al momento de instanciarlo, y para modificar y retornar el valor de dichos atributos. La clase debe contar con un método para aplicar un descuento al precio del producto a partir de un valor porcentual, implementar también una función para mostrar la información del producto por consola.

Implementar una unidad de prueba para verificar el funcionamiento de cada operación en la clase. Evaluar las operaciones donde es necesario hacer una validación de datos.

7. Calculadora de IMC

Implementar una clase 'CalculadoraIMC' que permita calcular el índice de masa corporal a partir del peso (en Kg) y la altura (en m). El IMC calculado debe ser un valor flotante con dos decimales de precisión. La clase debe tener un método que le devuelva información al usuario a partir del IMC calculado de la siguiente manera:

- si IMC < 18.5 --> Tu IMC {} está 'Debajo de lo normal'
- si 18.5 < IMC < 25 --> Tu IMC {} es 'Normal'
- si 25 < IMC < 30 --> Tu IMC {} indica 'Sobrepeso'
- si MC > 30 --> Tu IMC {} indica 'Obesidad'

Agregar control en los datos de entrada para evitar alturas y pesos fuera de rangos aceptables.

Implementar una unidad de prueba para verificar el funcionamiento de cada operación en la clase.

8. AnalizadorTexto

Se desea modelar una clase que analice un texto que se recibe como parámetro en su inicialización. El analizador debe tener funcionalidades para depurar el texto eliminando los siguientes caracteres: ',', '.', ':', ';', '-', '_' también debe devolver el número de palabras presentes en el texto depurado y contar la ocurrencia de una palabra recibida por parámetro en el texto depurado.

Implementar una unidad de prueba para verificar el funcionamiento de cada operación en la clase.

9. Conversor de unidades de temperatura

Queremos modelar un conversor de unidades de temperatura para convertir valores de temperatura entre las unidades Fahrenheit, Celsius y Kelvin (identificadas con los caracteres 'F', 'C' y 'K'), el conversor debe manejar los casos en que la temperatura no sea físicamente realizable (< 0 K) y si se ingresan otras opciones diferentes a 'F', 'C' y 'K' como parámetros. Para convertir una temperatura de una unidad a otra, primero se convierte la temperatura a Kelvin y después a las unidades requeridas. Si la unidad de temperatura original es la misma a la que se quiere convertir, se debe retornar el mismo valor sin modificar.

FÓRMULAS:

$T_K = T_C + 273.15$

$T_K = (T_F + 459.67)*5/9$

Implementar una unidad de prueba para verificar el funcionamiento de cada operación en la clase.