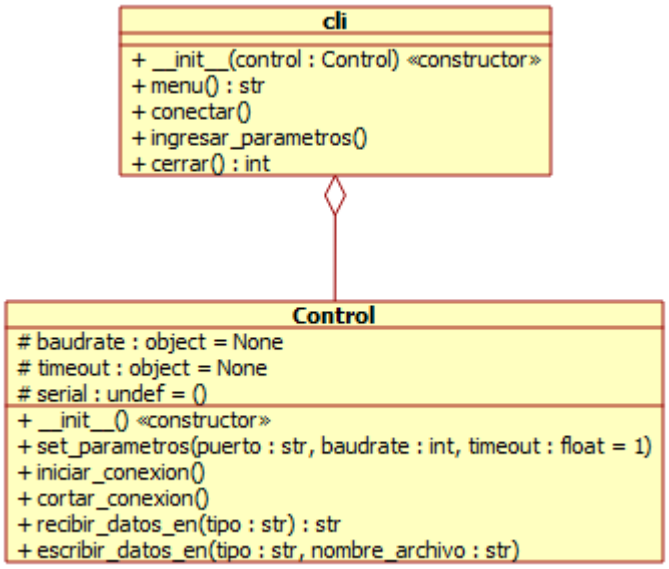
Trabajo práctico N° 2

Ejercicio N° 1




```
1. Ingresar Parametros
2. Iniciar Conexión
3. Recibir Datos y Ver
4. Recibir Datos y Escribir
5. Salir
Ingrese una opcion: 1
Ingrese el puerto: COM5
Ingrese el baudrate: 19200
```

```
Ingrese una opcion: 2
Conectando...
```

```
1. Ingresar Parametros
2. Iniciar Conexión
3. Recibir Datos y Ver
4. Recibir Datos y Escribir
5. Salir
Ingrese una opcion: 3
Ingrese el formato a recibir: j
Recibiendo datos...
{
  "dispositivo_id": 2,
  "porcentaje_valvula": 2,
  "estado_nivel": "medio",
  "caudal": 64.55,
}
```

```
1. Ingresar Parametros
2. Iniciar Conexión
3. Recibir Datos y Ver
4. Recibir Datos y Escribir
5. Salir
Ingrese una opcion: 4
Ingrese el formato a recibir: c
Recibiendo datos...
Ingrese el nombre del archivo: dat
```

Clase cli

Atributos:

- _control: instancia de la clase Control, que maneja la conexión serial y el intercambio de datos.

Métodos:

- __init__(control: Control): Inicializa la clase, recibe una instancia de Control y lanza el menú principal en un bucle.
- menu(): Muestra el menú principal y gestiona las acciones según la opción seleccionada por el usuario.
- conectar(): Inicia la conexión serial a través de la clase Control.
- ingresar_parametros(): Permite ingresar los parámetros de puerto y baudrate para la conexión.
- recibir_datos(escritura: bool = False): Recibe datos desde el puerto serial; si escritura es True, guarda los datos en un archivo.
- cerrar(): Cierra la conexión serial llamando al método correspondiente en Control.

Clase Control

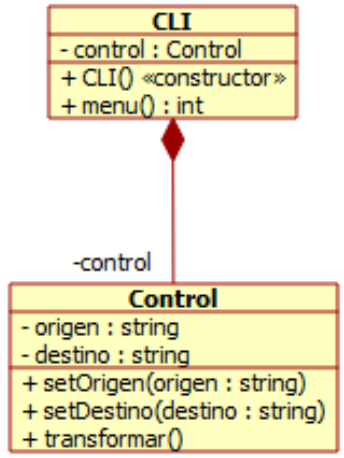
Atributos:

- _puerto: puerto serial a conectar.
- _baudrate: velocidad de transmisión de datos.
- _timeout: tiempo de espera para la conexión.
- _serial: instancia de la clase serial.Serial() para manejar la conexión serial.

Métodos:

- __init__(): Inicializa los atributos del puerto, baudrate y timeout, además de la instancia de serial.Serial.
- set_parametros(puerto: str, baudrate: int, timeout: float = 1): Configura los parámetros del puerto, baudrate y timeout.
- iniciar_conexion(): Establece la conexión serial con los parámetros configurados.
- cortar_conexion(): Cierra la conexión serial.
- recibir_datos_en(tipo: str): Recibe datos desde el puerto serial en un formato específico ("x", "j", "c") y los devuelve como texto.
- escribir_datos_en(tipo: str, nombre_archivo: str): Recibe datos y los escribe en un archivo CSV en la ruta especificada.

Ejercicio N° 2




```
1. Transferir Datos CSV a XML
2. Salir
Ingrese una opcion: 1
Ingrese nombre de origen:
dat.csv
Ingrese nombre de destino:
dat.xml
Transformando C://Users//pato//Documents//UNIVERSIDAD//8S//POO//TPSPOO//13612-TP2//13612
-TP2-2//anexo//dat.csv a C://Users//pato//Documents//UNIVERSIDAD//8S//POO//TPSPOO//13612
-TP2//13612-TP2-2//anexo//dat.xml
1. Transferir Datos CSV a XML
2. Salir
Ingrese una opcion: 2
```

Clase CLI

Atributos:

- control: instancia de la clase Control, que gestiona la transformación de archivos CSV a XML.

Métodos:

- CLI(): Constructor que inicializa la clase y ejecuta un bucle que muestra el menú y permite realizar la transformación de archivos o salir del programa.
- menu(): Muestra el menú de opciones al usuario y devuelve la opción seleccionada.

Clase Control

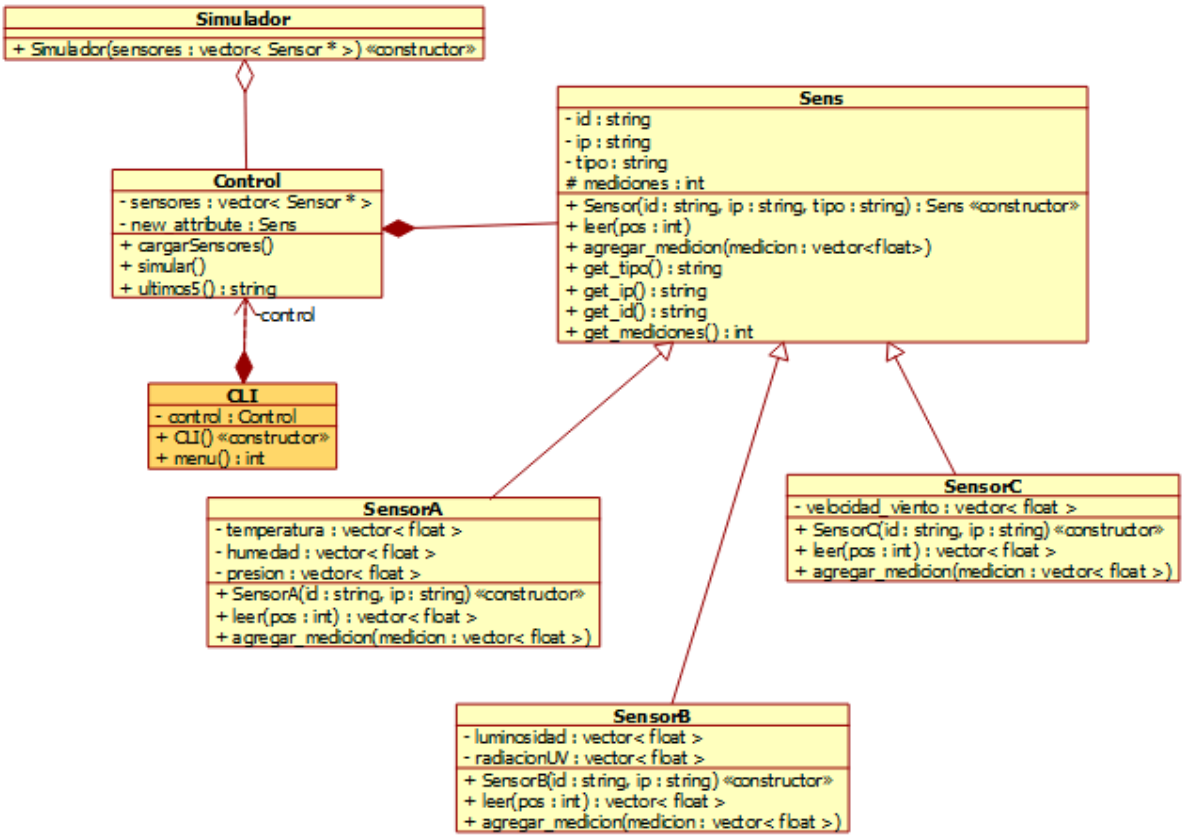
Atributos:

- origen: ruta completa del archivo CSV de origen.
- destino: ruta completa del archivo XML de destino.

Métodos:

- setOrigen(string origen): Establece la ruta del archivo CSV de origen concatenando la ruta base con el nombre del archivo ingresado.
- setDestino(string destino): Establece la ruta del archivo XML de destino concatenando la ruta base con el nombre del archivo ingresado.
- transformar(): Convierte el archivo CSV de origen en un archivo XML, línea por línea, escribiendo el contenido en el archivo de destino.

Ejercicio N° 3




```
1. Cargar sensores
2. Simular
3. Salir
Ingrese una opcion: 1
1. Cargar sensores
2. Simular
3. Salir
Ingrese una opcion: 2
Simulando...
Presione cualquier tecla para terminar...
```

```
1. Cargar sensores
2. Simular
3. Salir
Ingrese una opcion:
3
Sensor S1 192.168.0.2 A
Medicion 3: Temperatura: 49.000000 Humedad: 12.000000 Presion: 179.000000
Medicion 2: Temperatura: 33.000000 Humedad: 13.000000 Presion: 684.000000
Medicion 1: Temperatura: 21.000000 Humedad: 50.000000 Presion: 675.000000
Medicion 0: Temperatura: 0.000000 Humedad: 0.000000 Presion: 0.000000

Sensor S2 192.168.0.3 B
Medicion 3: Luminosidad: 718.000000 Radiacion UV: 7.000000
Medicion 2: Luminosidad: 874.000000 Radiacion UV: 8.000000
Medicion 1: Luminosidad: 706.000000 Radiacion UV: 5.000000
Medicion 0: Luminosidad: 0.000000 Radiacion UV: 0.000000

Sensor S3 192.168.0.4 C
Medicion 3: Velocidad del viento: 12.000000
Medicion 2: Velocidad del viento: 52.000000
Medicion 1: Velocidad del viento: 82.000000
Medicion 0: Velocidad del viento: 0.000000

Sensor S4 192.168.0.5 A
Medicion 3: Temperatura: 39.000000 Humedad: 19.000000 Presion: 163.000000
Medicion 2: Temperatura: 49.000000 Humedad: 0.000000 Presion: 641.000000
Medicion 1: Temperatura: 6.000000 Humedad: 14.000000 Presion: 45.000000
Medicion 0: Temperatura: 0.000000 Humedad: 0.000000 Presion: 0.000000

Sensor S5 192.168.0.6 B
Medicion 3: Luminosidad: 700.000000 Radiacion UV: 2.000000
Medicion 2: Luminosidad: 152.000000 Radiacion UV: 9.000000
Medicion 1: Luminosidad: 547.000000 Radiacion UV: 5.000000
Medicion 0: Luminosidad: 0.000000 Radiacion UV: 0.000000
```

Clase CLI

Atributos:

- control: instancia de la clase Control, que gestiona la carga de sensores, simulación y visualización de datos.

Métodos:

- CLI(): Constructor que ejecuta un bucle que permite cargar sensores, simular y visualizar las últimas 5 mediciones.
- menu(): Muestra el menú de opciones al usuario y devuelve la opción seleccionada.

Clase Control

Atributos:

- sensores: vector que contiene instancias de sensores de diferentes tipos (SensorA, SensorB, SensorC).

Métodos:

- cargarSensores(): Carga la configuración de los sensores desde un archivo de configuración (config.cfg).
- simular(): Inicia la simulación de los sensores a través de la clase Simulador.
- ultimos5(): Devuelve las últimas 5 mediciones de cada sensor en formato de texto.

Clase Sensor

Atributos:

- id: identificador del sensor.
- ip: dirección IP del sensor.
- tipo: tipo de sensor (A, B, C).
- mediciones: contador de mediciones registradas.

Métodos:

- Sensor(id, ip, tipo): Constructor que inicializa los atributos del sensor.
- get_tipo(): Devuelve el tipo de sensor.
- get_id(): Devuelve el identificador del sensor.
- get_ip(): Devuelve la dirección IP del sensor.
- get_mediciones(): Devuelve el número de mediciones registradas.

Clase SensorA

Atributos:

- temperatura: vector que almacena las mediciones de temperatura.
- humedad: vector que almacena las mediciones de humedad.
- presion: vector que almacena las mediciones de presión.

Métodos:

- SensorA(id, ip): Constructor que inicializa un sensor de tipo A con valores por defecto.
- leer(pos): Devuelve las mediciones de temperatura, humedad y presión en la posición especificada.
- agregar_medicion(medicion): Agrega una nueva medición de temperatura, humedad y presión.

Clase SensorB

Atributos:

- luminosidad: vector que almacena las mediciones de luminosidad.
- radiacionUV: vector que almacena las mediciones de radiación UV.

Métodos:

- SensorB(id, ip): Constructor que inicializa un sensor de tipo B con valores por defecto.
- leer(pos): Devuelve las mediciones de luminosidad y radiación UV en la posición especificada.
- agregar_medicion(medicion): Agrega una nueva medición de luminosidad y radiación UV.

Clase SensorC

Atributos:

- velocidad_viento: vector que almacena las mediciones de velocidad del viento.

Métodos:

- SensorC(id, ip): Constructor que inicializa un sensor de tipo C con valores por defecto.
- leer(pos): Devuelve la medición de velocidad del viento en la posición especificada.
- agregar_medicion(medicion): Agrega una nueva medición de velocidad del viento.

Clase Simulador

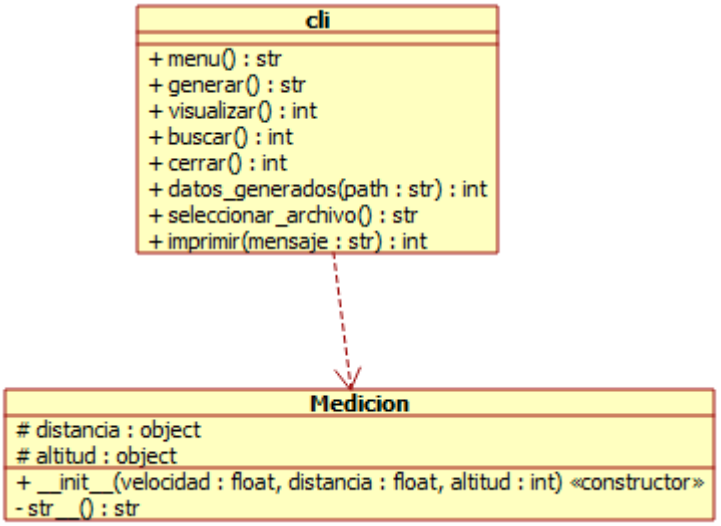
Atributos:

- sensores: vector que contiene instancias de diferentes sensores para simular.

Métodos:

- Simulador(sensores): Constructor que inicia una simulación aleatoria de mediciones para los sensores.

Ejercicio N° 4




```
1. Generar Datos y Guardar
2. Visualizar Datos
3. Buscar Medición Mayor Altitud
4. Salir
Ingrese una opción: 1
Ingrese la cantidad de mediciones a generar: 5
Ingrese el nombre del archivo: datos
Generando...
Datos generados y guardados en c:\Users\pato\Documents\UNIVERSIDAD\8S\POO\TPSPOO\13612-TP2\13612-TP2-4\anexo\datos

1. Generar Datos y Guardar
2. Visualizar Datos
3. Buscar Medición Mayor Altitud
4. Salir
Ingrese una opción: 2
Ingrese el nombre del archivo: datos
¿Desea mostrar el promedio de velocidad? (s/n): s
Visualizando...
Velocidad: 34.47 m/s, Distancia: 825.46 m, Altitud: 25 m
Velocidad: 22.91 m/s, Distancia: 941.11 m, Altitud: 94 m
Velocidad: 5.67 m/s, Distancia: 525.8 m, Altitud: 31 m
Velocidad: 49.28 m/s, Distancia: 279.58 m, Altitud: 21 m
Velocidad: 87.72 m/s, Distancia: 327.74 m, Altitud: 57 m
Promedio de velocidad: 40.010000000000005
```

Clase cli

Métodos:

- menu(): Muestra el menú de opciones y devuelve la opción seleccionada si es válida.
- generar(): Solicita la cantidad de mediciones a generar y el nombre del archivo para guardarlas. Devuelve estos valores si son válidos.
- visualizar(): Muestra un mensaje indicando que se están visualizando los datos.
- buscar(): Muestra un mensaje indicando que se está buscando la medición de mayor altitud.
- cerrar(): Muestra un mensaje indicando que se está cerrando la aplicación.
- datos_generados(path: str): Imprime un mensaje confirmando la generación y guardado de datos en una ruta especificada.
- seleccionar_archivo(): Solicita el nombre de un archivo y si se desea mostrar el promedio de velocidad. Valida los inputs y devuelve el nombre del archivo y la opción seleccionada.
- imprimir(mensaje: str): Imprime un mensaje proporcionado.

Clase Medicion

Atributos:

- _velocidad: velocidad de la medición (en m/s).
- _distancia: distancia de la medición (en metros).
- _altitud: altitud de la medición (en metros).

Métodos:

- __init__(velocidad: float, distancia: float, altitud: int): Inicializa una instancia de Medicion con los valores de velocidad, distancia y altitud.
- __str__(): Devuelve una representación en texto de la medición, mostrando velocidad, distancia y altitud.