Lic. en Sistemas de Información

Asignatura: Ingeniería en Software 2

Trabajo práctico N° 7

Alumna: Blanchet Ibarra Sofia

1.

C: > Users > Invitado1 > OneDrive > Documentos > Facultad > tercero > IS2 > TP7 > getJason.py.py > ...

```python
import json
import sys
import os

class JsonReader:
    def __init__(self, ruta_archivo):
        self.ruta_archivo = ruta_archivo
        self.datos = {}

    def cargar(self):
        if not os.path.exists(self.ruta_archivo):
            raise FileNotFoundError(f"No se encuentra el archivo: {self.ruta_archivo}")
        with open(self.ruta_archivo, 'r') as archivo:
            contenido = archivo.read()
            self.datos = json.loads(contenido)

    def obtener_valor(self, clave):
        return self.datos.get(clave, f"Clave '{clave}' no encontrada en el archivo.")

def main():
    if len(sys.argv) < 2:
        print("Uso: python getJason.py <archivo_json> [clave]")
        sys.exit(1)

    archivo = sys.argv[1]
    clave = sys.argv[2] if len(sys.argv) >= 3 else 'token1'

    lector = JsonReader(archivo)
    try:
        lector.cargar()
        resultado = lector.obtener_valor(clave)
        print(resultado)
    except Exception as e:
        print("Error:", e)

if __name__ == "__main__":
    main()
```

2.

```python
class JsonReaderSingleton:
    _instancia = None

    def __new__(cls, *args, **kwargs):
        if cls._instancia is None:
            cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
        return cls._instancia

    def __init__(self, ruta_archivo=None):
        if not hasattr(self, "inicializado"):
            self.ruta_archivo = ruta_archivo
            self.datos = {}
            self.inicializado = True
```

3.

TP7 > getJason_v1_1.py > ...

```python
import os
import sys
import json
class JsonReaderSingleton:
    _instancia = None

    def __new__(cls, *args, **kwargs):
        if cls._instancia is None:
            cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
        return cls._instancia

    def __init__(self, ruta_archivo=None):
        if not hasattr(self, 'inicializado'):
            self.ruta_archivo = ruta_archivo
            self.datos = {}
            self.inicializado = True

    def cargar(self):
        try:
            with open(self.ruta_archivo, 'r', encoding='utf-8') as archivo:
                self.datos = json.load(archivo)
        except FileNotFoundError:
            print(f"Error controlado: El archivo '{self.ruta_archivo}' no fue encontrado.")
            sys.exit(4)
        except json.JSONDecodeError:
            print(f"Error controlado: El archivo '{self.ruta_archivo}' no tiene formato JSON válido.")
            sys.exit(6)
        except Exception as e:
            print(f"Error controlado inesperado al leer el archivo: {e}")
            sys.exit(7)

    def obtener_valor(self, clave):
        return self.datos.get(clave, f"La clave '{clave}' no se encontró en el archivo.")

def main():
    if '-v' in sys.argv or '--version' in sys.argv:
        print("Versión 1.1")
        sys.exit(0)

    if len(sys.argv) < 2:
        print("Uso: python getJason_v1_1.py <archivo_json> [clave]")
        sys.exit(2)

    archivo = sys.argv[1]
```

```
45
46      if not archivo.endswith(".json"):
47          print("Error: El archivo debe tener extensión .json")
48          sys.exit(3)
49
50      if not os.path.exists(archivo):
51          print(f"Error: El archivo '{archivo}' no existe")
52          sys.exit(4)
53
54      clave = sys.argv[2] if len(sys.argv) > 2 else "token1"
55
56      try:
57          lector = JsonReaderSingleton(archivo)
58          lector.cargar()
59          resultado = lector.obtener_valor(clave)
60          print(resultado)
61      except Exception as e:
62          print("Error controlado en ejecución:", e)
63          sys.exit(5)
64
65  if __name__ == "__main__":
66      main()
67
```

4.

```
from abc import ABC, abstractmethod

class JsonReaderBase(ABC):

    @abstractmethod
    def cargar(self):
        pass
    @abstractmethod
    def obtener_valor(self, clave):
        pass


from abc import ABC, abstractmethod

class JsonReaderBase(ABC):
    @abstractmethod
    def cargar(self):
        pass

    @abstractmethod
    def obtener_valor(self, clave):
        pass
```

5.

TP7 > getJason_v1_1.py > ...

```python
import os
import sys
import json
from abc import ABC, abstractmethod


class JsonReaderBase(ABC):
    """Interfaz abstracta para lectores de JSON."""

    @abstractmethod
    def cargar(self):
        pass

    @abstractmethod
    def obtener_valor(self, clave):
        pass


class JsonReaderSingleton(JsonReaderBase):
    """Implementación Singleton de lector de archivos JSON."""

    _instancia = None

    def __new__(cls, *args, **kwargs):
        if cls._instancia is None:
            cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
        return cls._instancia

    def __init__(self, ruta_archivo=None):
        if not hasattr(self, 'inicializado'):
            self.ruta_archivo = ruta_archivo
            self.datos = {}
            self.inicializado = True

    def cargar(self):
        try:
            with open(self.ruta_archivo, 'r', encoding='utf-8') as archivo:
                self.datos = json.load(archivo)
        except FileNotFoundError:
            print(f"Error controlado: El archivo '{self.ruta_archivo}' no fue encontrado.")
            sys.exit(4)
        except json.JSONDecodeError:
            print(f"Error controlado: El archivo '{self.ruta_archivo}' no tiene formato JSON válido.")
            sys.exit(6)
        except Exception as e:
            print(f"Error inesperado al leer el archivo: {e}")
            sys.exit(7)

    def obtener_valor(self, clave):
        return self.datos.get(clave, f"La clave '{clave}' no se encontró en el archivo.")

```

```python

def main():
    if '-v' in sys.argv or '--version' in sys.argv:
        print("Versión 1.1")
        sys.exit(0)

    if len(sys.argv) < 2:
        print("Uso: python getJason_v1_1.py <archivo_json> [clave]")
        sys.exit(2)

    archivo = sys.argv[1]

    if not archivo.endswith(".json"):
        print("Error: El archivo debe tener extensión .json")
        sys.exit(3)

    if not os.path.exists(archivo):
        print(f"Error: El archivo '{archivo}' no existe")
        sys.exit(4)

    clave = sys.argv[2] if len(sys.argv) > 2 else "token1"

    try:
        lector: JsonReaderBase = JsonReaderSingleton(archivo)
        lector.cargar()
        resultado = lector.obtener_valor(clave)
        print(resultado)
    except Exception as e:
        print("Error controlado en ejecución:", e)
        sys.exit(5)


if __name__ == "__main__":
    main()

```

6.

```python
"""
getJason_v1_1.py - Versión refactorizada 1.1

Copyright © 2024 UADER - FCyT - Ingeniería en Sistemas de Información
Todos los derechos reservados.

Descripción:
Este programa permite cargar un archivo JSON desde línea de comandos
y obtener el valor asociado a una clave. El diseño implementa el
patrón Singleton, se ejecuta desde la línea de comandos con
validación robusta de argumentos y sigue una estrategia de
Branching by Abstraction con una clase base abstracta.

Ejemplo de uso:
    python getJason_v1_1.py sitedata.json token1
"""


import os ···


class JsonReaderBase(ABC):
    """Interfaz abstracta para lectores de archivos JSON."""

    @abstractmethod
    def cargar(self):
        """Carga el contenido del archivo JSON."""
        pass

    @abstractmethod
    def obtener_valor(self, clave):
        """Devuelve el valor asociado a una clave específica."""
        pass

class JsonReaderSingleton(JsonReaderBase):
    """Implementación Singleton de lector de archivos JSON."""

    _instancia = None

    def __new__(cls, *args, **kwargs):
        """Crea una única instancia de la clase."""
        if cls._instancia is None:
            cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
        return cls._instancia

    def __init__(self, ruta_archivo=None):
        """Inicializa la instancia con la ruta del archivo JSON."""
        if not hasattr(self, 'inicializado'):
            self.ruta_archivo = ruta_archivo
            self.datos = {}
            self.inicializado = True

    def cargar(self):
        """Carga el contenido del archivo JSON, con manejo de errores controlado."""
        try:
```

```
        try:
            with open(self.ruta_archivo, 'r', encoding='utf-8') as archivo:
                self.datos = json.load(archivo)
        except FileNotFoundError:
            print(f"Error controlado: El archivo '{self.ruta_archivo}' no fue encontrado.")
            sys.exit(4)
        except json.JSONDecodeError:
            print(f"Error controlado: El archivo '{self.ruta_archivo}' no tiene formato JSON válido.")
            sys.exit(6)
        except Exception as e:
            print(f"Error inesperado al leer el archivo: {e}")
            sys.exit(7)

    def obtener_valor(self, clave):
        """Obtiene el valor de una clave en el JSON, o un mensaje si no existe."""
        return self.datos.get(clave, f"La clave '{clave}' no se encontró en el archivo.")


def main():
    """Función principal que procesa los argumentos de línea de comandos."""

    # Opción de mostrar versión
    if '-v' in sys.argv or '--version' in sys.argv:
        print("Versión 1.1")
        sys.exit(0)

    # Validar número de argumentos
    if len(sys.argv) < 2:
        print("Uso: python getJason_v1_1.py <archivo_json> [clave]")
        sys.exit(2)

    archivo = sys.argv[1]

    # Validar extensión del archivo
    if not archivo.endswith(".json"):
        print("Error: El archivo debe tener extensión .json")
        sys.exit(3)

    # Validar existencia del archivo
    if not os.path.exists(archivo):
        print(f"Error: El archivo '{archivo}' no existe")
        sys.exit(4)

```

```

    # Establecer clave por defecto si no se proporciona
    clave = sys.argv[2] if len(sys.argv) > 2 else "token1"

    try:
        lector: JsonReaderBase = JsonReaderSingleton(archivo)
        lector.cargar()
        resultado = lector.obtener_valor(clave)
        print(resultado)
    except Exception as e:
        print("Error controlado en ejecución:", e)
        sys.exit(5)


if __name__ == "__main__":
    main()

```

7.

```
PS C:\Users\Invitado1\OneDrive\Documentos\Facultad\tercero\IS2\TP7> pylint getJason_v1_1.py
************* Module getJason_v1_1
getJason_v1_1.py:65:0: C0301: Line too long (102/100) (line-too-long)
getJason_v1_1.py:1:0: C0103: Module name "getJason_v1_1" doesn't conform to snake_case naming style (invalid-name)
getJason_v1_1.py:31:8: W0107: Unnecessary pass statement (unnecessary-pass)
getJason_v1_1.py:36:8: W0107: Unnecessary pass statement (unnecessary-pass)
getJason_v1_1.py:67:15: W0718: Catching too general exception Exception (broad-exception-caught)
getJason_v1_1.py:109:11: W0718: Catching too general exception Exception (broad-exception-caught)

-----------------------------------
Your code has been rated at 9.00/10

PS C:\Users\Invitado1\OneDrive\Documentos\Facultad\tercero\IS2\TP7>
```