



**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 > ...
1  import json
2  import sys
3  import os
4
5  class JsonReader:
6      def __init__(self, ruta_archivo):
7          self.ruta_archivo = ruta_archivo
8          self.datos = {}
9
10     def cargar(self):
11         if not os.path.exists(self.ruta_archivo):
12             raise FileNotFoundError(f"No se encuentra el archivo: {self.ruta_archivo}")
13         with open(self.ruta_archivo, 'r') as archivo:
14             contenido = archivo.read()
15             self.datos = json.loads(contenido)
16
17     def obtener_valor(self, clave):
18         return self.datos.get(clave, f"Clave '{clave}' no encontrada en el archivo.")
19
20 def main():
21     if len(sys.argv) < 2:
22         print("Uso: python getJason.py <archivo_json> [clave]")
23         sys.exit(1)
24
25     archivo = sys.argv[1]
26     clave = sys.argv[2] if len(sys.argv) >= 3 else 'token1'
27
28     lector = JsonReader(archivo)
29     try:
30         lector.cargar()
31         resultado = lector.obtener_valor(clave)
32         print(resultado)
33     except Exception as e:
34         print("Error:", e)
35
36 if __name__ == "__main__":
37     main()
38
```

```
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
```

2.

```

TP7 > getJason_v1_1.py > ...
1  import os
2  import sys
3  import json
4  class JsonReaderSingleton:
5      _instancia = None
6
7      def __new__(cls, *args, **kwargs):
8          if cls._instancia is None:
9              cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
10             return cls._instancia
11
12         def __init__(self, ruta_archivo=None):
13             if not hasattr(self, 'inicializado'):
14                 self.ruta_archivo = ruta_archivo
15                 self.datos = {}
16                 self.inicializado = True
17
18         def cargar(self):
19             try:
20                 with open(self.ruta_archivo, 'r', encoding='utf-8') as archivo:
21                     self.datos = json.load(archivo)
22             except FileNotFoundError:
23                 print(f"Error controlado: El archivo '{self.ruta_archivo}' no fue encontrado.")
24                 sys.exit(4)
25             except json.JSONDecodeError:
26                 print(f"Error controlado: El archivo '{self.ruta_archivo}' no tiene formato JSON válido.")
27                 sys.exit(6)
28             except Exception as e:
29                 print(f"Error controlado inesperado al leer el archivo: {e}")
30                 sys.exit(7)
31
32         def obtener_valor(self, clave):
33             return self.datos.get(clave, f"La clave '{clave}' no se encontró en el archivo.")
34
35     def main():
36         if '-v' in sys.argv or '--version' in sys.argv:
37             print("Versión 1.1")
38             sys.exit(0)
39
40         if len(sys.argv) < 2:
41             print("Uso: python getJason_v1_1.py <archivo_json> [clave]")
42             sys.exit(2)
43
44         archivo = sys.argv[1]
45

```

3.

```

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 > ...

```
1  import os
2  import sys
3  import json
4  from abc import ABC, abstractmethod
5
6
7  class JsonReaderBase(ABC):
8      """Interfaz abstracta para lectores de JSON."""
9
10     @abstractmethod
11     def cargar(self):
12         pass
13
14     @abstractmethod
15     def obtener_valor(self, clave):
16         pass
17
18
19  class JsonReaderSingleton(JsonReaderBase):
20      """Implementación Singleton de lector de archivos JSON."""
21
22      _instancia = None
23
24      def __new__(cls, *args, **kwargs):
25          if cls._instancia is None:
26              cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
27          return cls._instancia
28
29      def __init__(self, ruta_archivo=None):
30          if not hasattr(self, 'inicializado'):
31              self.ruta_archivo = ruta_archivo
32              self.datos = {}
33              self.inicializado = True
34
35      def cargar(self):
36          try:
37              with open(self.ruta_archivo, 'r', encoding='utf-8') as archivo:
38                  self.datos = json.load(archivo)
39          except FileNotFoundError:
40              print(f"Error controlado: El archivo '{self.ruta_archivo}' no fue encontrado.")
41              sys.exit(4)
42          except json.JSONDecodeError:
43              print(f"Error controlado: El archivo '{self.ruta_archivo}' no tiene formato JSON válido.")
44              sys.exit(6)
45          except Exception as e:
46              print(f"Error inesperado al leer el archivo: {e}")
47              sys.exit(7)
48
49      def obtener_valor(self, clave):
50          return self.datos.get(clave, f"La clave '{clave}' no se encontró en el archivo.")
51
```

```

52
53 def main():
54     if '-v' in sys.argv or '--version' in sys.argv:
55         print("Versión 1.1")
56         sys.exit(0)
57
58     if len(sys.argv) < 2:
59         print("Uso: python getJason_v1_1.py <archivo_json> [clave]")
60         sys.exit(2)
61
62     archivo = sys.argv[1]
63
64     if not archivo.endswith(".json"):
65         print("Error: El archivo debe tener extensión .json")
66         sys.exit(3)
67
68     if not os.path.exists(archivo):
69         print(f"Error: El archivo '{archivo}' no existe")
70         sys.exit(4)
71
72     clave = sys.argv[2] if len(sys.argv) > 2 else "token1"
73
74     try:
75         lector: JsonReaderBase = JsonReaderSingleton(archivo)
76         lector.cargar()
77         resultado = lector.obtener_valor(clave)
78         print(resultado)
79     except Exception as e:
80         print("Error controlado en ejecución:", e)
81         sys.exit(5)
82
83
84 if __name__ == "__main__":
85     main()
86

```

6.

```
1  """
2  getJson_v1_1.py - Versión refactorizada 1.1
3
4  Copyright © 2024 UADER - FCyT - Ingeniería en Sistemas de Información
5  Todos los derechos reservados.
6
7  Descripción:
8  Este programa permite cargar un archivo JSON desde línea de comandos
9  y obtener el valor asociado a una clave. El diseño implementa el
10 patrón Singleton, se ejecuta desde la línea de comandos con
11 validación robusta de argumentos y sigue una estrategia de
12 Branching by Abstraction con una clase base abstracta.
13
14 Ejemplo de uso:
15 |   python getJson_v1_1.py sitedata.json token1
16 """
17
18
19 > import os ...
20
21
22
23
24
25 class JsonReaderBase(ABC):
26     """Interfaz abstracta para lectores de archivos JSON."""
27
28     @abstractmethod
29     def cargar(self):
30         """Carga el contenido del archivo JSON."""
31         pass
32
33     @abstractmethod
34     def obtener_valor(self, clave):
35         """Devuelve el valor asociado a una clave específica."""
36         pass
37
38 class JsonReaderSingleton(JsonReaderBase):
39     """Implementación Singleton de lector de archivos JSON."""
40
41     _instancia = None
42
43     def __new__(cls, *args, **kwargs):
44         """Crea una única instancia de la clase."""
45         if cls._instancia is None:
46             cls._instancia = super(JsonReaderSingleton, cls).__new__(cls)
47         return cls._instancia
48
49     def __init__(self, ruta_archivo=None):
50         """Inicializa la instancia con la ruta del archivo JSON."""
51         if not hasattr(self, 'inicializado'):
52             self.ruta_archivo = ruta_archivo
53             self.datos = {}
54             self.inicializado = True
55
56     def cargar(self):
57         """Carga el contenido del archivo JSON, con manejo de errores controlado."""
58         try:
```

```

58         try:
59             with open(self.ruta_archivo, 'r', encoding='utf-8') as archivo:
60                 self.datos = json.load(archivo)
61         except FileNotFoundError:
62             print(f"Error controlado: El archivo '{self.ruta_archivo}' no fue encontrado.")
63             sys.exit(4)
64         except json.JSONDecodeError:
65             print(f"Error controlado: El archivo '{self.ruta_archivo}' no tiene formato JSON válido.")
66             sys.exit(6)
67         except Exception as e:
68             print(f"Error inesperado al leer el archivo: {e}")
69             sys.exit(7)
70
71     def obtener_valor(self, clave):
72         """Obtiene el valor de una clave en el JSON, o un mensaje si no existe."""
73         return self.datos.get(clave, f"La clave '{clave}' no se encontró en el archivo.")
74
75
76 def main():
77     """Función principal que procesa los argumentos de línea de comandos."""
78
79     # Opción de mostrar versión
80     if '-v' in sys.argv or '--version' in sys.argv:
81         print("Versión 1.1")
82         sys.exit(0)
83
84     # Validar número de argumentos
85     if len(sys.argv) < 2:
86         print("Uso: python getJson_v1_1.py <archivo_json> [clave]")
87         sys.exit(2)
88
89     archivo = sys.argv[1]
90
91     # Validar extensión del archivo
92     if not archivo.endswith(".json"):
93         print("Error: El archivo debe tener extensión .json")
94         sys.exit(3)
95
96     # Validar existencia del archivo
97     if not os.path.exists(archivo):
98         print(f"Error: El archivo '{archivo}' no existe")
99         sys.exit(4)
100

```

```

100
101     # Establecer clave por defecto si no se proporciona
102     clave = sys.argv[2] if len(sys.argv) > 2 else "token1"
103
104     try:
105         lector: JsonReaderBase = JsonReaderSingleton(archivo)
106         lector.cargar()
107         resultado = lector.obtener_valor(clave)
108         print(resultado)
109     except Exception as e:
110         print("Error controlado en ejecución:", e)
111         sys.exit(5)
112
113
114
115 if __name__ == "__main__":
116     main()
117

```



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>
```