sandeepsuryaprasad / python_tutorials Private

Code Issues Pull requests Actions Projects Wiki Securit

master python_tutorials / 10_oops / 6_inheritance_logging.py / Jump to Go to file ...

Sandeep Suryaprasad clean up Latest commit 7965b72 on 24 Jun History

0 contributors

113 lines (96 sloc) | 4.19 KB Raw Blame

```python
class ConsoleLogger:
    def log(self, message):
        print(message)  # prints something in the console

class TextFileLogger:
    def __init__(self, file_object):
        self.file_object = file_object

    def log(self, message):
        self.file_object.write(message.strip())
        self.file_object.write("\n")
        self.file_object.flush()

class CSVLogger:
    def __init__(self, file_object):
        self.file_object = file_object

    def log(self, message):
        words = message.split()
        from csv import writer
        csv_writer = writer(self.file_object)
        csv_writer.writerow(words)
        self.file_object.flush()
# ------------------------------------------------------------------------
# BAD DESIGN
# ------------------------------------------------------------------------
# Inheritance (IS-A relationship)
```

```python
class FilteredConsoleLogger(ConsoleLogger):
    def __init__(self, pattern):
        self.pattern = pattern

    # overriding the log method present in Parent class
    def log(self, message):
        if self.pattern in message:     # adding extra functionality (filteri
            super().log(message)    # reusing the existing functionality of p

class FilteredTextFileLogger(TextFileLogger):
    def __init__(self, pattern, file_object):
        self.pattern = pattern
        super().__init__(file_object)

    def log(self, message):
        if self.pattern in message:
            super().log(message)

class FilteredCSVLogger(CSVLogger):
    def __init__(self, pattern, file_object):
        self.pattern = pattern
        super().__init__(file_object)

    def log(self, message):
        if self.pattern in message:
            super().log(message)

# ------------------------------------------------------------------------
# USING MULTIPLE INHERITANCE
# ------------------------------------------------------------------------
class FilteredLogger:
    def __init__(self, pattern):
        self.pattern = pattern

    def log(self, message):
        if self.pattern in message:
            super().log(message)
# ------------------------------------------------------------------------
class FilteredConsoleLogger(FilteredLogger, ConsoleLogger):
    def __init__(self, pattern):
        FilteredLogger.__init__(self, pattern)

class FilteredTextFileLogger(FilteredLogger, TextFileLogger):
    def __init__(self, pattern, file_object):
        FilteredLogger.__init__(self, pattern)
```

```
73          TextFileLogger.__init__(self, file_object)
74
75  class FilteredCSVLogger(FilteredLogger, CSVLogger):
76      def __init__(self, pattern, file_object):
77          FilteredLogger.__init__(self, pattern)
78          CSVLogger.__init__(self, file_object)
79  # ------------------------------------------------------------------------
80  # Composition (HAS-A relationship)
81  # ------------------------------------------------------------------------
82  class FilteredLogger:
83      def __init__(self, pattern, logger_type):
84          self.pattern = pattern
85          self.logger_type = logger_type
86
87      # Polymorphic behavior or it is also called as Duck Typing
88      def log(self, message):
89          if self.pattern in message:
90              self.logger_type.log(message)
91
92  # ------------------------------------------------------------------------
93  with open('./data_files/sample.log') as f:
94      with open("events.csv", "w") as fw:
95          text_logger = TextFileLogger(fw)
96          logger = FilteredLogger("INFO", text_logger)
97          for line in f:
98              logger.log(line)
99  # ------------------------------------------------------------------------
100 # ------------------------------------------------------------------------
101 # "pattern" as class variable
102 class FilteredLogger:
103     pattern = None  # class variable
104     def log(self, message):
105         if self.pattern in message:
106             super().log(message)
107
108 class FilteredConsoleLogger(FilteredLogger, ConsoleLogger):
109     pattern = "ERROR"
110
111 class FilteredTextLogger(FilteredLogger, TextFileLogger):
112     pattern = "ERROR"
113 # ------------------------------------------------------------------------
```