2-2-2025

Gustavo Andrés García Anguiano

Pruebas de software y aseguramiento de la calidad

Actividad 4.2

INDICE

Clase General usada por los 3 programas con los métodos de las operaciones requeridas.

```
"""
This module contains a class that reads numbers from a file and saves results to a file.
"""
import re
from collections import defaultdict
class GeneralClass:
    """
    A class to read numbers from a file and save results to a file.
    """
    @staticmethod
    def get_numbers(filename):
        """Reads numbers from a file and returns them as a list."""
        numbers = []
        with open(filename, 'r',  encoding='utf-8') as file:
            for line in file:
                try:
                    numbers.append(int(line.strip()))
                except ValueError:
                    print(f"Invalid data found and skipped: {line.strip()}")
        return numbers
    @staticmethod
    def get_words(filename):
        """Reads words from a file and returns them as a list."""
        words = []
        with open(filename, 'r', encoding='utf-8') as file:
            for line in file:
                words.extend(re.findall(r'\b\w+\b', line.lower()))
        return words
    @staticmethod
    def save_results(filename, results, execution_time, program_name):
        """Function to save the results to a file."""
        with open(filename, 'w', encoding='utf-8') as file:
            if program_name == 'compute_statics':
                file.write("Descriptive Statistics:\n")
                file.write(f"Mean: {results[0]}\n")
                file.write(f"Median: {results[1]}\n")
                file.write(f"Mode: {results[2]}\n")
                file.write(f"Standard Deviation: {results[3]}\n")
                file.write(f"Variance: {results[4]}\n")
                file.write(f"Execution Time: {execution_time:.4f} seconds\n")
            elif program_name == "convert_numbers":
                file.write("Number | Binary | Hexadecimal\n")
                for num, binary, hexa in results:
                    file.write(f"{num} | {binary} | {hexa}\n")
                file.write(f"Execution Time: {execution_time:.4f} seconds\n")
            elif program_name == "word_count":
                file.write("Word | Frequency\n")
                for word, count in sorted(results.items(), key=lambda x: x[1], reverse=True):
                    file.write(f"{word} | {count}\n")
                file.write(f"Execution Time: {execution_time:.4f} seconds\n")
    @staticmethod
    def get_compute_statistics(numbers):
        """Generate descriptive statistics from a list of numbers without using python libraries."""
        if not numbers:
            return None
        mean = sum(numbers) / len(numbers)
        sorted_numbers = sorted(numbers)
        mid = len(sorted_numbers) // 2
        median = (sorted_numbers[mid] if len(numbers) % 2 != 0
                else (sorted_numbers[mid - 1] + sorted_numbers[mid]) / 2)
        frequency = {}
        for num in numbers:
            frequency[num] = frequency.get(num, 0) + 1
        mode = max(frequency, key=frequency.get)
        variance = sum((x - mean) ** 2 for x in numbers) / len(numbers)
        std_dev = variance ** 0.5
```

```
        return mean, median, mode, std_dev, variance
    @staticmethod
    def get_convertion(numbers):
        """Converts numbers to binary and hexadecimal representations."""
        conversions = [(num, bin(num)[2:], hex(num)[2:]) for num in numbers]
        return conversions
    @staticmethod
    def get_count_word_freq(words):
        """Counts the frequency of each distinct word."""
        frequency = defaultdict(int)
        for word in words:
            frequency[word] += 1
        return frequency
```

computeStatics.Py

```
"""
Code to compute all descriptive statistics from a file containing numbers.
The results shall be print on a screen and on a file named StatisticsResults.txt.
"""

import argparse
import time
import pyclasses.read_and_save as rs

def main():
    """Main function to parse arguments, compute statistics, and save results."""
    parser = argparse.ArgumentParser(description="Creata compute descriptive statistics from file.")
    parser.add_argument("filename", help="Path to the file containing numerical data.")
    args = parser.parse_args()
    start_time = time.time()
    numbers = rs.GeneralClass.get_numbers(args.filename)
    results = rs.GeneralClass.get_compute_statistics(numbers)
    execution_time = time.time() - start_time
    if results:
        print("Descriptive Statistics:")
        print(f"Mean: {results[0]}")
        print(f"Median: {results[1]}")
        print(f"Mode: {results[2]}")
        print(f"Standard Deviation: {results[3]}")
        print(f"Variance: {results[4]}")
        print(f"Execution Time: {execution_time:.4f} seconds")
        rs.GeneralClass.save_results(
            "StatisticsResults.txt", results, execution_time, "compute_statics"
            )
    else:
        print("No valid numbers found in the file.")

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

convertNumbers.py

```
"""
Code to convert the numbers to binary and hexadecimal base.
The results shall be print on a screen and on a file named ConvertionResults.txt.
"""

import argparse
import time
import pyclasses.read_and_save as rs

def main():
    """Main function to read numbers from a file and convert them to binary and hexadecimal."""
```

```
    parser = argparse.ArgumentParser(description="Convert numbers to binary and hexadecimal.")
    parser.add_argument("filename", help="Path to the file containing numerical data.")
    args = parser.parse_args()
    start_time = time.time()
    numbers = rs.GeneralClass.get_numbers(args.filename)
    results = rs.GeneralClass.get_convertion(numbers)
    execution_time = time.time() - start_time
    if results:
        print("Conversions:")
        for num, binary, hexa in results:
            print(f"{num} -> Binary: {binary}, Hexadecimal: {hexa}")
        print(f"Execution Time: {execution_time:.4f} seconds")
        rs.GeneralClass.save_results(
            "ConvertionResults.txt", results, execution_time, "convert_numbers"
            )
    else:
        print("No valid numbers found in the file.")

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

wordCount.py

```
"""
Code to identify all distinct words and the frequency of them
(how many times the word “X” appears in the file).
The results shall be print on a screen and on a file named WordCountResults.txt.
"""

import argparse
import time
import pyclasses.read_and_save as rs

def main():
    """Main function to read words from a file and count their frequency."""
    parser = argparse.ArgumentParser(description="Count word frequency in a file.")
    parser.add_argument("filename", help="Path to the file containing text data.")
    args = parser.parse_args()
    start_time = time.time()
    words = rs.GeneralClass.get_words(args.filename)
    word_counts = rs.GeneralClass.get_count_word_freq(words)
    execution_time = time.time() - start_time
    if word_counts:
        print("Word Frequencies:")
        for word, count in sorted(word_counts.items(), key=lambda x: x[1], reverse=True):
            print(f"{word}: {count}")
        print(f"Execution Time: {execution_time:.4f} seconds")
        rs.GeneralClass.save_results(
            "WordCountResults.txt", word_counts, execution_time, "word_count")
    else:
        print("No words found in the file.")

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