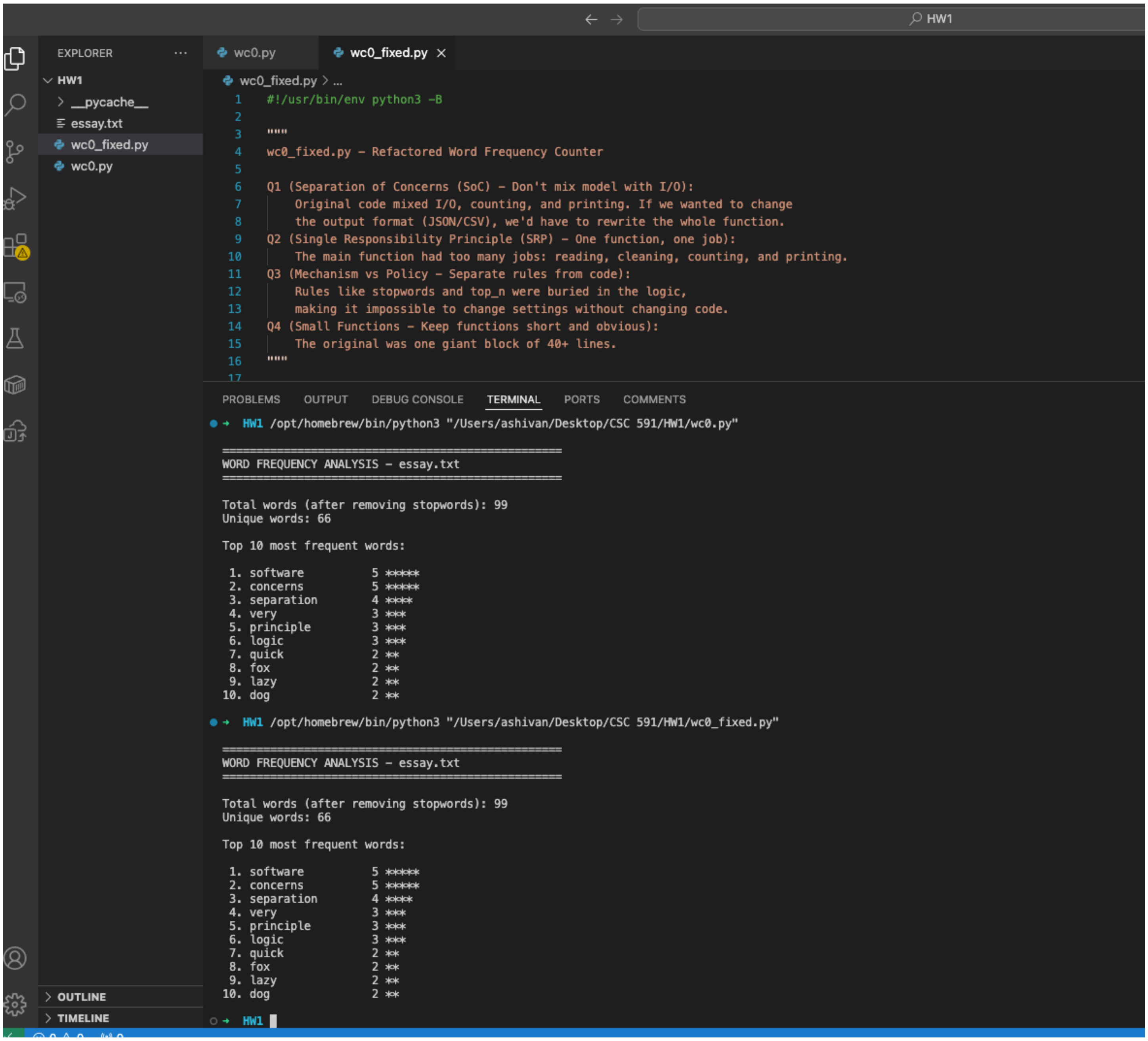
HW 1 - Group 10 Aniruddha Shivananda (ashivan@ncsu.edu) JonCarlo Migaly (jmigaly@ncsu.edu)

Screenshot of Execution-



```
#!/usr/bin/env python3 -B

"""
wc0_fixed.py - Refactored Word Frequency Counter

Q1 (Separation of Concerns (SoC) - Don't mix model with I/O):
    Original code mixed I/O, counting, and printing. If we wanted to change
    the output format (JSON/CSV), we'd have to rewrite the whole function.
Q2 (Single Responsibility Principle (SRP) - One function, one job):
    The main function had too many jobs: reading, cleaning, counting, and printing.
Q3 (Mechanism vs Policy - Separate rules from code):
    Rules like stopwords and top_n were buried in the logic,
    making it impossible to change settings without changing code.
Q4 (Small Functions - Keep functions short and obvious):
    The original was one giant block of 40+ lines.
"""
```

```
● → HW1 /opt/homebrew/bin/python3 "/Users/ashivan/Desktop/CSC 591/HW1/wc0.py"

==================================================
WORD FREQUENCY ANALYSIS - essay.txt
==================================================

Total words (after removing stopwords): 99
Unique words: 66

Top 10 most frequent words:

 1. software          5 *****
 2. concerns          5 *****
 3. separation        4 ****
 4. very              3 ***
 5. principle         3 ***
 6. logic             3 ***
 7. quick             2 **
 8. fox               2 **
 9. lazy              2 **
10. dog               2 **
● → HW1 /opt/homebrew/bin/python3 "/Users/ashivan/Desktop/CSC 591/HW1/wc0_fixed.py"

==================================================
WORD FREQUENCY ANALYSIS - essay.txt
==================================================

Total words (after removing stopwords): 99
Unique words: 66

Top 10 most frequent words:

 1. software          5 *****
 2. concerns          5 *****
 3. separation        4 ****
 4. very              3 ***
 5. principle         3 ***
 6. logic             3 ***
 7. quick             2 **
 8. fox               2 **
 9. lazy              2 **
10. dog               2 **
○ → HW1 ▌
```

wc0_fixed.py -

```
#!/usr/bin/env python3 -B

"""
wc0_fixed.py - Refactored Word Frequency Counter

Q1 (Separation of Concerns (SoC) - Don't mix model with I/O):
    Original code mixed I/O, counting, and printing. If we wanted to change
    the output format (JSON/CSV), we'd have to rewrite the whole function.
Q2 (Single Responsibility Principle (SRP) - One function, one job):
    The main function had too many jobs: reading, cleaning, counting, and printing.
```

```python
Q3 (Mechanism vs Policy – Separate rules from code):
    Rules like stopwords and top_n were buried in the logic,
    making it impossible to change settings without changing code.
Q4 (Small Functions – Keep functions short and obvious):
    The original was one giant block of 40+ lines.
"""

from types import SimpleNamespace as obj
import json   # Bonus 1: for JSON output

# Policies (AQ3: Policies seperated from mechanism)
CONFIG = {
    "language": "english",

    "stopwords_english": [
        "the", "a", "an", "and", "or", "but", "in", "on", "at", "to", "for",
        "of", "is", "was", "are", "were", "be", "been", "with"
    ],

    # Bonus 4: support for other languages
    "stopwords_spanish": ["el", "la", "los", "con", "y", "o", "en"],

    # Bonus 2: external stopword file
    "stopwords_file": None,

    "punctuation": '.,!?;:"()[]',
    "top_n": 10
}

# Data Structure
def WordData():
    return obj(words={}, total_words=0, unique_words=0)

# Model (AQ1: SoC)
def read_file(file):
    """AQ2: Single Responsibility – Only handles file I/O streaming"""
    with open(file) as f:
        for line in f:
            yield line

def clean_text(line: str):
    """AQ2/AQ4: Focused logic for initial string splitting"""
    return line.lower().split()

def clean_punc(word: str):
    """AQ2/AQ4: Specifically handles punctuation stripping policy"""
    return word.strip(CONFIG["punctuation"])

def get_stopwords():
    """AQ3: Select stopwords policy"""
    if CONFIG["stopwords_file"]:
        return load_stopwords(CONFIG["stopwords_file"])
    return CONFIG[f"stopwords_{CONFIG['language']}"]
```

```python
def adjust_count(line: str, word_data):
    """AQ2/AQ4: Only handles dictionary updates and counter increments"""
    stopwords = get_stopwords()
    for word in line:
        word = clean_punc(word)
        if word and word not in stopwords:
            if word not in word_data.words:
                word_data.unique_words += 1
            word_data.total_words += 1
            word_data.words[word] = word_data.words.get(word, 0) + 1

def sort_word_data(word_data):
    """AQ2: Dedicated function for sorting results for the view"""
    sorted_words = sorted(
        word_data.words.items(),
        key=lambda x: x[1],
        reverse=True
    )
    word_data.words = sorted_words

def processText(file_name: str):
    """AQ2: Orchestrator that coordinates the model logic flow"""
    word_data = WordData()
    for line in read_file(file_name):
        line = clean_text(line)
        adjust_count(line, word_data)
    sort_word_data(word_data)
    return word_data

# --- Presentation (AQ1: Separation of Concerns) ---
def showHeader(file_name: str):
    """AQ4: Small function for header formatting"""
    print(f"\n{'='*50}")
    print(f"WORD FREQUENCY ANALYSIS - {file_name}")
    print(f"{'='*50}\n")

def showStats(words):
    """AQ1: Pure view logic"""
    print(f"Total words (after removing stopwords): {words.total_words}")
    print(f"Unique words: {words.unique_words}\n")
    print(f"Top {CONFIG['top_n']} most frequent words:\n")

    # AQ3: Uses the separated policy
    top_words = list(words.words)[:CONFIG["top_n"]]
    for i, (word, count) in enumerate(top_words, 1):
        bar = "*" * count
        print(f"{i:2}. {word:15} {count:3} {bar}")

def report(words: dict, file_name: str):
    """AQ2: Master presentation function"""
    showHeader(file_name)
    showStats(words)
    print()

# Bonus 1: Multiple output formats
def toJSON(words):
    """Return results as JSON (no printing)"""
    return json.dumps(
        {
            "total_words": words.total_words,
            "unique_words": words.unique_words,
            "frequencies": dict(words.words)
        },
        indent=2
    )

def toCSV(words):
    """Return results as CSV (no printing)"""
    rows = ["word,count"]
    for word, count in words.words:
        rows.append(f"{word},{count}")
    return "\n".join(rows)

# Bonus 2: Load stopwords from file
def load_stopwords(path):
    """Load stopwords from external file"""
    with open(path) as f:
        return [w.strip() for w in f if w.strip()]

# Bonus 3: Simple unit tests
# NOTE:
# These tests are provided externally by the instructor.
# They intentionally test the counting mechanism in isolation,
# even though the default policy removes some words (e.g., "and").
# I have kept tests unchanged to match the provided specification.
def test_clean_punc():
    assert clean_punc("hello,") == "hello"

def test_adjust_count():
    # AQ3 NOTE:
    # This test assumes "and" is counted.
    # In the full program, this word is removed by policy (stopwords),
    # but the test is preserved as provided to validate core behavior.
    data = WordData()
    adjust_count(["the", "cat", "and", "dog"], data)
    assert data.words == {"cat": 1, "and": 1, "dog": 1}

# --- Main ---
if __name__ == "__main__":
    # 1. Process data (Model logic)
    data = processText("essay.txt")

    # 2. Show data (Presentation logic)
    report(data, "essay.txt")
```