REPORT

LAB 번호 : HW1

과목 및 분반 : 자바프로그래밍2 2분반

제출일 : 2025.09.16

학번 : 32203919 (컴퓨터공학과)

이름 : 장천명

1.기존 코드의 문제점

기존의 App.java의 경우, SOLID 원칙을 위배한 Monolithic 구조

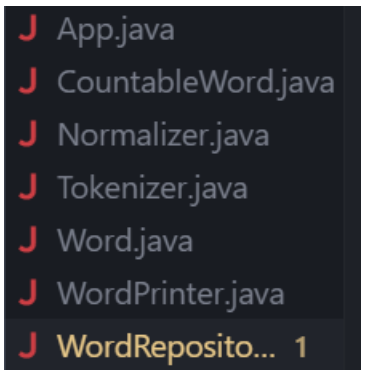
- **단일 책임 위반(SRP)**
    - 하나의 클래스가 파일 입출력. 파싱, 정규화, 저장, 출력을 모두 담당
- **개방 폐쇄 위반(OCP)**
    - 새로운 구분자나 정규화가 필요할 때 기존 코드를 수정해야 함
- **리스코프 치환 위반(LSP)**
    - Equals와 Hashcode가 일치 하지 않아 HashMap에서 예상과 다른 동작 발생 (Java에서는 equal()와 hashcode()가 함께 구현되어야함
    - CountableWord가 Word를 상속받았는데, 동일한 동작을 보장하지 못함. (CountalbeWord에는 equal만 구현되어 있음, Word는 equal, hashcode 둘 다 구현 X)
- **인터페이스 분리 위반(ISP)**
    - 넓은 인터페이스 문제로 일부 구현체에게 불필요한 메서드 강제
- **의존성 역전 위반(DIP)**
    - 고수준 모듈이 저수준 모듈에 직접 의존하여 확장성 부족

2.해결책

단일 책임 원칙(SRP) 준수

개선 방법 - 각 클래스가 하나의 책임만 담당하도록 분리

Word.java, Tokenizer.java, Normalizer.java, WordRepository.java, WordPrinter.java로 나눔



개선 후 – 각 클래스가 명확한 하나의 책임의 가져 유지보수성 향상

개방 폐쇄 원칙(OCP) 준수

개선 방법 – 인터페이스와 의존성 주입을 통한 확장성 확보

```
// 의존성 조립
Tokenizer tokenizer = new Tokenizer();
Normalizer normalizer = new Normalizer();
WordRepository repository = new WordRepository();
WordPrinter printer = new WordPrinter();
```

개선 후 – 새로운 구현체로 교체 시 코드 수정 없이 생성자에게만 변경 가능

리스코프 치환 원칙(LSP) 준수

개선 방법 – equal()와 HashCode() 일치 구현

```
// CountableWord.java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    CountableWord that = (CountableWord) o;
    return word.equals(that.word);
}

@Override
public int hashCode() {
    return word.hashCode();  // equals와 일치!
```

개선 후 - HashMap에서 예상대로 동작하여 부모 클래스 치환 가능

인터페이스 분리 원칙(ISP) 준수

개선 방법 – 인터페이스 제거 및 구체적 클래스 사용

```
// 개선: 구체적 클래스로 직접 구현
public class CountableWord {
    public void increaseCount() { count++; }
    public void decreaseCount() { if (count > 0) count--; }
    public int getCount() { return count; }
```

개선 후 – 불필요한 인터페이스 제거로 코드 단순화(구체적 클래스로 직접 구현)

의존성 역전 원칙(DIP) 준수

개선 방법 – 의존성 주입을 통한 추상화 의존

```
// 의존성 조립
Tokenizer tokenizer = new Tokenizer();
Normalizer normalizer = new Normalizer();
WordRepository repository = new WordRepository();
WordPrinter printer = new WordPrinter();
```

개선 후 – 구체적 구현에 의존하지 않고 추상화에 의존하여 확장성 확보(의존성을 외부에서 주입)

---

3. YourCode

```
// yourcode: 단어의 길이를 반환합니다.
public int getLength() {
    return value.length();
```

단어의 길이를 반환하는 코드 추가(Word.java)

```
// yourcode: 단어 길이 기능 테스트
System.out.println(x:"\n=== yourcode: 단어 길이 기능 테스트 ===");
testWordLengthFeature(repository);
```

```
// yourcode: 단어 길이 기능을 테스트하는 메서드
private static void testWordLengthFeature(WordRepository repository) {
    System.out.println(x:"단어별 길이 정보:");
    for (Map.Entry<Character, List<CountableWord>> entry : repository.getAll().entrySet()) {
        for (CountableWord cw : entry.getValue()) {
            Word word = cw.getWord();
            System.out.println("- " + word + " (길이: " + word.getLength() + "글자, 개수: " + cw.getCount() + "번)");
```

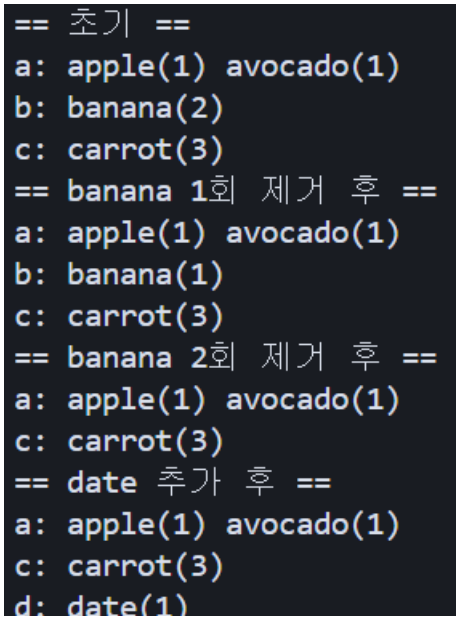
App.java에 단어 길이 테스트 코드 추가

```
=== yourcode: 단어 길이 기능 테스트 ===
단어별 길이 정보:
- apple (길이: 5글자, 개수: 1번)
- avocado (길이: 7글자, 개수: 1번)
- carrot (길이: 6글자, 개수: 3번)
- date (길이: 4글자, 개수: 1번)
```

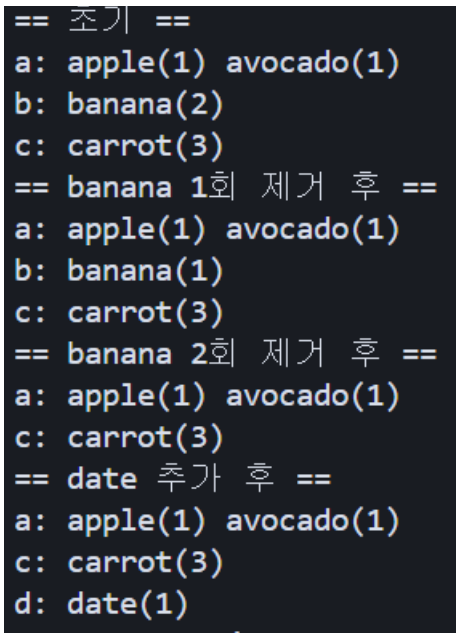
---

4. 실행 결과창

개선 전 실행 결과



개선 후 실행 결과



기능적으로 동일하나, 구조는 개선되었음을 알 수 있음

---

* Gemini를 통해 도움 받았음

* SOLID 원칙 참고 블로그