# REPORT


LAB 번호 : LAB5

과목 및 분반 : 자바프로그래밍 2분반

제출일 : 2025.06.04

학번 : 32203919 (컴퓨터공학과)

이름 : 장천명

DKU 단국대학교 DANKOOK UNIVERSITY

## 1. 구현 방식의 설명

model, parser, finder, sorter, util로 패키지를 구성

- model에는 재해별 클래스랑 enum들로 구성
- parser에는 각 재해 데이터를 읽어서 객체로 바꿔주는 파서들로 구성
- finder에는 다양한 검색 전략을 쉽게 바꿔쓸 수 있게 전략 패턴을 적용
- comparator랑 sort에는 정렬 기준과 정렬기준들로 구성


---

## 2. 주요 코드 설명

**NaturalHazard.java**

```java
public abstract class NaturalHazard { // 자연재해의 기본 추상 클래스
    private int year;
    private int month;
    private int day;
    private String location;
    private double latitude;
    private double longitude;

    public NaturalHazard(int year, int month, int day, String location, double latitude, double longitude) {
        this.year = year;
        this.month = month;
        this.day = day;
        this.location = location;
        this.latitude = latitude;
        this.longitude = longitude;      You, 지난주 • Lab4
    }
```

public abstract class NaturalHazard로 추상 클래스를 사용하였다.

---

**Tsunami.java**

```
public class Tsunami extends NaturalHazard {
    private TsunamiEventValidity tsunamiEventValidity;
    private TsunamiCauseCode tsunamiCauseCode;
    private String country;
    private double maximumWaterHeight;
    private int numberOfRunup;
       You, 그저께 • Lab5

    public Tsunami(int year, int month, int day,
                   TsunamiEventValidity tsunamiEventValidity,
                   TsunamiCauseCode tsunamiCauseCode, String country,
                   String location, double latitude, double longitude,
                   double maximumWaterHeight, int numberOfRunup) {
        super(year, month, day, location, latitude, longitude);
        this.tsunamiEventValidity = tsunamiEventValidity;
        this.tsunamiCauseCode = tsunamiCauseCode;
        this.country = country;
        this.maximumWaterHeight = maximumWaterHeight;
        this.numberOfRunup = numberOfRunup;
    }
```

Tsunami.java는 Natural Hazard를 상속받았다. NaturalHazard에서 year, month, day, location, latituude, longitude를 상속받고 추가로 tsunami에만 필요한 tsunamiEventvalidity, tsuunamiCauseCode, country, maximumWaterHeigh, numberOfRunup을 받는다.

---

**TsunamiEventValidity.java**

```java
package model;
      You, 그저께 • Lab5 …
public enum TsunamiEventValidity {
    ERRORNEOUS_ENTRY(-1),
    EVENT_THAT_ONLY_CAUSED_A_SEICHE(code:0),
    VERY_DOUBTFUL_TSUNAMI(code:1),
    QUESTIONABLE_TSUNAMI(code:2),
    PROBABLE_TSUNAMI(code:3),
    DEFINITE_TSUNAMI(code:4);

    private int code;

    TsunamiEventValidity(int code) {
        this.code = code;
    }

    public int getCode() {
        return code;
    }

    // 코드에 해당하는 validity 반환
    public static TsunamiEventValidity fromCode(int code) {
        for (TsunamiEventValidity validity : values()) {
            if (validity.getCode() == code) {
                return validity;
            }
        }
        throw new IllegalArgumentException("Invalid tsunami event validity code: " + code);
    }
}
```

TsunamiEventValidity를 열거형을 구현한 enum으로, 각 enum은 code 값을 가진다. 그래서 int code을 받아서, 해당하는 enum으로 반환한다.

---

**TsunamiCauseCode.java**

```java
package model;

public enum TsunamiCauseCode {
    UNKNOWN(code:0),
    EARTHQUAKE(code:1),
    QUESTIONABLE_EARTHQUAKE(code:2),
    EARTHQUAKE_AND_LANDSLIDE(code:3),
    VOLCANO_AND_EARTHQUAKE(code:4),
    VOLCANO_EARTHQUAKE_AND_LANDSLIDE(code:5),
    VOLCANO(code:6),
    VOLCANO_AND_LANDSLIDE(code:7),
    LANDSLIDE(code:8),
    METEOROLOGICAL(code:9),
    EXPLOSION(code:10),
    ASTRONOMICAL_TIDE(code:11);

    private int code;       You, 그저께 • Lab5

    TsunamiCauseCode(int code) {
        this.code = code;
    }

    public int getCode() {
        return code;
    }

    public static TsunamiCauseCode fromCode(int code) {
        for (TsunamiCauseCode cause : values()) {
            if (cause.getCode() == code) {
                return cause;       You, 그저께 • Lab5
            }
        }
        throw new IllegalArgumentException("Invalid tsunami cause code: " + code);
    }
}
```

TsunamiEventValidity랑 비슷하게 열거형을 구현한 enum으로, 각 enum 마다 int code를 가진다. 마찬가지로 int code를 받아서, 해당하는 enum으로 반환한다.

---

**IParser.java**

```java
package parser;

import java.util.List;

...
public interface IParser<T> {
    List<T> parse(String[][] data); // 데이터를 파싱하여 리스트로 반환
}
```

List<T>를 사용함으로써, parsing 결과로 반활한 객체의 타입을 제네릭 하였다. 다양한 데이터 타입에 대해 재사용할 수 있다. 그리고  인터페이스여서 선언만 하고, 자식이 정의를 오버라이딩해서 사용하도록 한다.

---

**TsunamiParser.java**

```java
package parser;

import java.util.ArrayList;
import java.util.List;
import model.Tsunami;
import model.TsunamiEventValidity;
import model.TsunamiCauseCode;

You, 그저께 | 1 author (You)
public class TsunamiParser implements IParser<Tsunami> {
    @Override
    public List<Tsunami> parse(String[][] tsunamiData) {
        List<Tsunami> tsunamis = new ArrayList<>();
        for (String[] data : tsunamiData) {      You, 그저께 • Lab5
            int year = Integer.parseInt(data[0]);
            int month = Integer.parseInt(data[1]);
            int day = data[2].isEmpty() ? 1 : Integer.parseInt(data[2]);
            TsunamiEventValidity validity = TsunamiEventValidity.fromCode(Integer.parseInt(data[3]));
            TsunamiCauseCode causeCode = TsunamiCauseCode.fromCode(Integer.parseInt(data[4]));
            String country = data[5];
            String location = data[6];
            double latitude = Double.parseDouble(data[7]);
            double longitude = Double.parseDouble(data[8]);
            double maxWaterHeight = Double.parseDouble(data[9]);
            int numberOfRunup = Integer.parseInt(data[10]);

            tsunamis.add(new Tsunami(year, month, day, validity, causeCode, country, location, latitude, longitude,
                maxWaterHeight, numberOfRunup));
        }

        return tsunamis;
    }
```

VolcanoParser, EarthquakeParser도 비슷하게 구현되었으며, IParser의 인터페이스를 구현한다.  2차원 배열 데이터를 받아서, Tsunami 객체 리스트로 변환하는 역할을 한다. List<Tsunami>로 tsunamis에 파싱된 객체들을 담았다. 8장에서 배운 내용처럼 List를 tsunamis.add 를 사용해서 추가한다.

**IfinderStrategy.java**

```java
package finder;

You, 9시간 전 | 1 author (You)
public interface IFinderStrategy<T> {
    boolean match(T item);
}
```

IfinderStrategy 인터페이스이며, 타입 파라미터 T를 사용하여 다양한 타입의 객체에서 사용할 수 있게 하였다. 이 인터페이스는 디자인 패턴 중 하나인 Strategy 패턴을 구현하기 위한 인터페이스다.

---

**Finder.java**

```java
package finder;

import java.util.ArrayList;
import java.util.List;
      You, 그저께 • Lab5 ...
You, 그저께 | 1 author (You)
public class Finder<T> {
    private IFinderStrategy<T> strategy; // 전략 객체

    public Finder(IFinderStrategy<T> strategy) {
        this.strategy = strategy; // 전략 객체 초기화
    }

    public List<T> find(List<T> items) {
        List<T> result = new ArrayList<>(); // 결과 리스트
        for (T item : items) {
            if (strategy.match(item)) { // 전략에 맞는 아이템 추가
                result.add(item); // 결과 리스트에 추가
            }
        }
        return result;
```

먼저 Finder<T>는 리스트에서 원하는 조건에 맞는 객체를 찾아주는 제네릭 찾기 도구이다. 생성

자를 통해 객체를 초기화 하며, find 메서드를 통해 조건에 맞는 item을 추가한다.

---

**TsunamiCauseCodeFinderStrategy.java**

```java
package finder;

import model.Tsunami;
import model.TsunamiCauseCode;

You, 그저께 | 1 author (You)
public class TsunamiCauseCodeFinderStrategy implements IFinderStrategy<Tsunami> {
    private TsunamiCauseCode code;

    public TsunamiCauseCodeFinderStrategy(TsunamiCauseCode code) {
        this.code = code;
    }
Ctrl+L to chat, Ctrl+K to generate
    @Override
    public boolean match(Tsunami item) {
        return item.getTsunamiCauseCode() == code;
    }
}
```

IfinderStrategy<Tsunami> 즉, Tsunami가 객체인 CauseCode에 해당하는지 판단하는 Strategy 역할을 한다. getTsunamiCauseCode(Tsunami.java에 있음)을 통해 쓰나미 원인 코드를 가지고와서 생성자에서 지정한 code랑 맞는지 비교한다. 나머지 FinderStrategy도 비슷한 방식이다.

---

**TsunamiCauseCodeComparator.java**

```java
package comparator;

import model.Tsunami;
import java.util.Comparator;

You, 그저께 | 1 author (You)
public class TsunamiCauseCodeComparator implements Comparator<Tsunami> {
    @Override
    public int compare(Tsunami t1, Tsunami t2) {
        return t1.getTsunamiCauseCode().getCode() - t2.getTsunamiCauseCode().getCode();
    }
}
```

Tsunami 객체 원인 코드의 값을 기준으로 비교하는 Comparator 클래스로, 나중에 Sort하기 위해 사용할 수 있다. 수업시간에 배웠듯이, Comparator를 쓰려면 직접 구현해야 함을 알 수 있다. 나머지 Comparator도 비슷한 방식이다.

## Comparator 인터페이스

- Comparator 인터페이스는 다른 두 개의 객체를 비교하기 위한 인터페이스

```
public interface Comparator {
      // o1가 o2보다 크면 1, 같으면 0, 작으면 -1을 반환한다.
      int compare(Object o1, Object o2);
}
```

```
class AgeComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        Person p1 = (Person)o1;
        Person p2 = (Person)o2;
        if (p1.age == p2.age) return 0;
        else if (p1.age > p2.age) return 1;
        else  return -1;
    }
}
```

---

**NaturalHazardSorter.java**

```
package sort;

import model.NaturalHazard;
import java.util.Comparator;
import java.util.List;
import java.util.Collections;

You, 그저께 | 1 author (You)
public class NaturalHazardSorter {
    public static <T extends NaturalHazard> void sort(List<T> hazards, Comparator<? super NaturalHazard> comparator) {
        // 제네릭 타입 T는 NaturalHazard 클래스를 상속받은 클래스만 가능
        Collections.sort(hazards, comparator); // 컬렉션 정렬
    }
}
```

제네릭 메서드이며, T는 NaturalHazard를 상속받은 클래스만 가능하다. Comparator<? super NaturalHazard> comparator은 정렬 기준을 정의하는 Comparator 객체 NaturalHazard 또는 그 상위 타입에 대한 Comparator를 받을 수 있다.

---

**TsunamiSorter.java**

```java
package sort;

import model.Tsunami;
import java.util.Comparator;
import java.util.List;
import java.util.Collections;

You, 그저께 | 1 author (You)
public class TsunamiSorter {
    public static void sortByEventValidity(List<Tsunami> tsunamis) {
        You, 그저께 | 1 author (You)
        Collections.sort(tsunamis, new Comparator<Tsunami>() {
            @Override
            public int compare(Tsunami t1, Tsunami t2) {
                return t1.getTsunamiEventValidity().getCode() - t2.getTsunamiEventValidity().getCode();
            }
        });       You, 그저께 • Lab5
```

TsunamiSorter는 쓰나미 리스트를 다양한 기준으로 정렬할 수 있는 정렬 유틸리티 클래스이다.

자바 표준 라이브러리의 정렬 메서드로, 리스트를 주어진 기준으로 정렬한다. 그리고 compare 메서드를 오버라이드하여, 두 쓰나미 객체의 비교 기준을 정의한다. 나머지 sortBy도 비슷한 방식이다.

---

**파일 저장하는 메서드 (Your Code) LAB5.java 코드**

```java
/**
     * 쓰나미 데이터를 CSV 파일로 저장하는 메서드
     * @param tsunamis 쓰나미 데이터 리스트
     * @param filename 저장할 파일 이름
     */
    private static void saveToCSV(List<Tsunami> tsunamis, String filename) {
        try (FileWriter writer = new FileWriter(filename)) {
            // CSV 헤더 작성
            writer.write("Year,Month,Day,Event Validity,Cause Code,Country,Location,Latitude,Longitude,Max Water Height,Number of Runup\n");

            // 데이터 작성
            for (Tsunami t : tsunamis) {
                writer.write(String.format("%s,%s,%s,%s,%s,%s,%s,%.3f,%.3f,%.2f,%d\n",
                    t.getYear(),
                    t.getMonth(),
                    t.getDay(),
                    t.getEventValidity(),
                    t.getCauseCode(),
                    t.getCountry(),
                    t.getLocation(),
                    t.getLatitude(),
                    t.getLongitude(),
                    t.getMaxWaterHeight(),
                    t.getNumberOfRunup()
                ));
            }

            System.out.println("\n결과가 " + filename + " 파일로 저장되었습니다.");
        } catch (IOException e) {
            System.err.println("CSV 파일 저장 중 오류가 발생했습니다: " + e.getMessage());
        }
    }
}
```

saveCSV 메서드는 Tsunami 객체 리스트를 받아, 지정한 파일명으로 csv 파일을 생성한다. 파일 쓰기를 위해 FileWriter랑 IOExceoption을 import 했다. 먼저 Header를 작성한 후, 각 객체의 데이터를 작성한다.

LAB5.java(main)

```
String[][] tsunamiData = {
    {"2020", "5", "29", "2", "4", "INDONESIA", "LESSER SUNDA: BALI: IJEN VOLCANO", "-8.058", "114.242", "3", "1"},
    {"2020", "6", "23", "4", "1", "MEXICO", "OFF COAST OF OAXACA", "16.029", "-95.901", "1.57", "9"},
    {"2020", "10", "30", "4", "1", "TURKEY", "AEGEAN SEA", "37.918", "26.790", "5.3", "268"},
    {"2021", "3", "4", "4", "1", "NEW ZEALAND", "S OF RAOUL ISLAND, KERMADEC ISLANDS", "-29.740", "-177.267", "1", "174"},
    {"2022", "1", "15", "4", "4", "TONGA", "TONGA ISLANDS", "-20.536", "-175.382", "22", "783"},
    {"2022", "9", "19", "4", "1", "MEXICO", "MEXICO", "18.367", "-103.252", "0.79", "14"},
    {"2022", "11", "30", "4", "1", "NEW ZEALAND", "LAKE TAUPO, NORTH ISLAND", "-38.808", "175.906", "1", "4"},
    {"2023", "9", "16", "4", "8", "GREENLAND", "DICKSON FJORD, GREENLAND", "72.810", "-26.950", "200", "3"},
    {"2024", "1", "1", "4", "3", "JAPAN", "HONSHU: W COAST", "37.498", "137.242", "7.23", "396"},
    {"2024", "4", "2", "4", "1", "TAIWAN", "E. TAIWAN-RYUKYU ISLANDS", "23.819", "121.562", "1", "12"},
    {"2024", "8", "7", "4", "8", "USA", "PEDERSEN LAGOON, AK", "59.904", "-149.825", "17", "3"},
    {"2025", "2", "", "3", "8", "ANTARCTICA", "DALK GLACIER", "-69.417", "76.450", "1", "1"}
    };
```

준비해둔 Tsunami 데이터를 입력한다.

```
TsunamiParser tsunamiParser = new TsunamiParser();
```

파서 생성을 먼저 한다.

```
List<Tsunami> tsunamiList = tsunamiParser.parse(tsunamiData);
```

파싱 해준다.

```
Tsunami[] tsunamis = tsunamiList.toArray(new Tsunami[0]);
```

배열로 변환 한다.

```
// Tsunami Finder 테스트
PrintUtil.printTitle(title:"7. Tsunami Event Validity Finder 테스트");
TsunamiEventValidityFinderStrategy validityStrategy = new TsunamiEventValidityFinderStrategy(TsunamiEventValidity.DEFINITE_TSUNAMI);
Finder<Tsunami> validityFinder = new Finder<>(validityStrategy);
List<Tsunami> validTsunamis = validityFinder.find(tsunamiList);
System.out.println(x:"확실한 쓰나미 이벤트:");
for (Tsunami t : validTsunamis) {
    System.out.println(t);
```

먼저 validityStrategy 찾기 전략 객체를 생성하고, 전략 객체를 이용하기 위해 Finder라는 validityFinder 찾기 도구를 만든다. 그리고 조건에 맞는 쓰나미만 골라서 validTsunamis 리스트로 만든다. 그리고 출력한다. 다른 테스트들도 비슷한 방식이다.

```
// Tsunami Sorter 테스트
PrintUtil.printTitle(title:"3. Tsunami Sorter 테스트");
System.out.println(x:"이벤트 유효성 기준 정렬:");
TsunamiSorter.sortByEventValidity(tsunamiList);
for (Tsunami t : tsunamiList) {
    System.out.println(t);
}
saveToCSV(tsunamiList, filename:"sorted_by_validity.csv");
```

Tsunami Sorter 테스트는 sortByEventValidity 메서드를 실행하므로써 이루어진다. TsunamiSorter의 sortByEventValidity가 호출되므로써, 아래의 코드가 실행된다.

```
Collections.sort(tsunamis, new Comparator<Tsunami>() {
    @Override
    public int compare(Tsunami t1, Tsunami t2) {
        return t1.getTsunamiEventValidity().getCode() - t2.getTsunamiEventValidity().getCode();
    }
});
```

---

## 3. 실행 결과창

```
7. Tsunami Event Validity Finder 테스트
----------------------------------------
확실한 쓰나미 이벤트:
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='1.57', numberOfRunup='9'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TURKEY', maximumWaterHeight='5.3', numberOfRunup='268'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='174'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='VOLCANO_AND_EARTHQUAKE', country='TONGA', maximumWaterHeight='22.0', numberOfRunup='7
83'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='0.79', numberOfRunup='14'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='4'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='GREENLAND', maximumWaterHeight='200.0', numberOfRunup='3'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE_AND_LANDSLIDE', country='JAPAN', maximumWaterHeight='7.23', numberOfRunup=
'396'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TAIWAN', maximumWaterHeight='1.0', numberOfRunup='12'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='USA', maximumWaterHeight='17.0', numberOfRunup='3'}

결과가 definite_tsunamis.csv 파일로 저장되었습니다.
```

definite_tsunamis.csv > data

```
Year,Month,Day,Event Validity,Cause Code,Country,Location,Latitude,Longitude,Max Water Height,Number of Runup
2020,6,23,DEFINITE_TSUNAMI,EARTHQUAKE,MEXICO,OFF COAST OF OAXACA,16.029,-95.901,1.57,9
2020,10,30,DEFINITE_TSUNAMI,EARTHQUAKE,TURKEY,AEGEAN SEA,37.918,26.790,5.30,268
2021,3,4,DEFINITE_TSUNAMI,EARTHQUAKE,NEW ZEALAND,S OF RAOUL ISLAND, KERMADEC ISLANDS,-29.740,-177.267,1.00,174
2022,1,15,DEFINITE_TSUNAMI,VOLCANO_AND_EARTHQUAKE,TONGA,TONGA ISLANDS,-20.536,-175.382,22.00,783
2022,9,19,DEFINITE_TSUNAMI,EARTHQUAKE,MEXICO,MEXICO,18.367,-103.252,0.79,14
2022,11,30,DEFINITE_TSUNAMI,EARTHQUAKE,NEW ZEALAND,LAKE TAUPO, NORTH ISLAND,-38.808,175.906,1.00,4
2023,9,16,DEFINITE_TSUNAMI,LANDSLIDE,GREENLAND,DICKSON FJORD, GREENLAND,72.810,-26.950,200.00,3
2024,1,1,DEFINITE_TSUNAMI,EARTHQUAKE_AND_LANDSLIDE,JAPAN,HONSHU: W COAST,37.498,137.242,7.23,396
2024,4,2,DEFINITE_TSUNAMI,EARTHQUAKE,TAIWAN,E. TAIWAN-RYUKYU ISLANDS,23.819,121.562,1.00,12
2024,8,7,DEFINITE_TSUNAMI,LANDSLIDE,USA,PEDERSEN LAGOON, AK,59.904,-149.825,17.00,3
```

EventValidity Fidner 테스트가 잘 이루어졌고 파일로 저장되었다.

```
8. Tsunami Cause Code Finder 테스트
----------------------------------------
지진으로 인한 쓰나미:
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='1.57', numberOfRunup='9'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TURKEY', maximumWaterHeight='5.3', numberOfRunup='268'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='174'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='0.79', numberOfRunup='14'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='4'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TAIWAN', maximumWaterHeight='1.0', numberOfRunup='12'}

결과가 earthquake_tsunamis.csv 파일로 저장되었습니다.
```

earthquake_tsunamis.csv > data

```
Year,Month,Day,Event Validity,Cause Code,Country,Location,Latitude,Longitude,Max Water Height,Number of Runup
2020,6,23,DEFINITE_TSUNAMI,EARTHQUAKE,MEXICO,OFF COAST OF OAXACA,16.029,-95.901,1.57,9
2020,10,30,DEFINITE_TSUNAMI,EARTHQUAKE,TURKEY,AEGEAN SEA,37.918,26.790,5.30,268
2021,3,4,DEFINITE_TSUNAMI,EARTHQUAKE,NEW ZEALAND,S OF RAOUL ISLAND, KERMADEC ISLANDS,-29.740,-177.267,1.00,174
2022,9,19,DEFINITE_TSUNAMI,EARTHQUAKE,MEXICO,MEXICO,18.367,-103.252,0.79,14
2022,11,30,DEFINITE_TSUNAMI,EARTHQUAKE,NEW ZEALAND,LAKE TAUPO, NORTH ISLAND,-38.808,175.906,1.00,4
2024,4,2,DEFINITE_TSUNAMI,EARTHQUAKE,TAIWAN,E. TAIWAN-RYUKYU ISLANDS,23.819,121.562,1.00,12
```

Tsunami Cause Code Finder 테스트가 잘 이루어지고 파일로 저장되었다.

```
9. Tsunami Country Finder 테스트
----------------------------------------
일본의 쓰나미:
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE_AND_LANDSLIDE', country='JAPAN', maximumWaterHeight='7.23', numberOfRunup=
'396'}

결과가 japan_tsunamis.csv 파일로 저장되었습니다.
----------------------------------------
10. Tsunami Maximum Water Height Finder 테스트
----------------------------------------
최대 수위 5.0-20.0m 쓰나미:
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TURKEY', maximumWaterHeight='5.3', numberOfRunup='268'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE_AND_LANDSLIDE', country='JAPAN', maximumWaterHeight='7.23', numberOfRunup=
'396'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='USA', maximumWaterHeight='17.0', numberOfRunup='3'}

결과가 high_water_tsunamis.csv 파일로 저장되었습니다.
----------------------------------------
11. Tsunami Number of Runup Finder 테스트
----------------------------------------
런업 횟수 100-500회 쓰나미:
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TURKEY', maximumWaterHeight='5.3', numberOfRunup='268'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='174'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE_AND_LANDSLIDE', country='JAPAN', maximumWaterHeight='7.23', numberOfRunup=
'396'}

결과가 many_runup_tsunamis.csv 파일로 저장되었습니다.
```

japan_tsunamis.csv > data

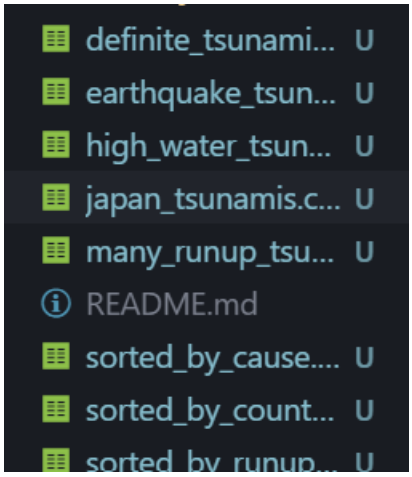
```
1  Year,Month,Day,Event Validity,Cause Code,Country,Location,Latitude,Longitude,Max Water Height,Number of Runup
2  2024,1,1,DEFINITE_TSUNAMI,EARTHQUAKE_AND_LANDSLIDE,JAPAN,HONSHU: W COAST,37.498,137.242,7.23,396
```

```
3. Tsunami Sorter 테스트
----------------------------------------
이벤트 유효성 기준 정렬:
{ tsunamiEventValidity='QUESTIONABLE_TSUNAMI', tsunamiCauseCode='VOLCANO_AND_EARTHQUAKE', country='INDONESIA', maximumWaterHeight='3.0', numberOfRunup='1'}
{ tsunamiEventValidity='PROBABLE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='ANTARCTICA', maximumWaterHeight='1.0', numberOfRunup='1'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='1.57', numberOfRunup='9'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TURKEY', maximumWaterHeight='5.3', numberOfRunup='268'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='174'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='VOLCANO_AND_EARTHQUAKE', country='TONGA', maximumWaterHeight='22.0', numberOfRunup='783'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='0.79', numberOfRunup='14'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='4'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='GREENLAND', maximumWaterHeight='200.0', numberOfRunup='3'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE_AND_LANDSLIDE', country='JAPAN', maximumWaterHeight='7.23', numberOfRunup='396'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TAIWAN', maximumWaterHeight='1.0', numberOfRunup='12'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='USA', maximumWaterHeight='17.0', numberOfRunup='3'}

결과가 sorted_by_validity.csv 파일로 저장되었습니다.
----------------------------------------
원인 코드 기준 정렬
----------------------------------------
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='1.57', numberOfRunup='9'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TURKEY', maximumWaterHeight='5.3', numberOfRunup='268'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='174'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='MEXICO', maximumWaterHeight='0.79', numberOfRunup='14'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='NEW ZEALAND', maximumWaterHeight='1.0', numberOfRunup='4'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE', country='TAIWAN', maximumWaterHeight='1.0', numberOfRunup='12'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='EARTHQUAKE_AND_LANDSLIDE', country='JAPAN', maximumWaterHeight='7.23', numberOfRunup='396'}
{ tsunamiEventValidity='QUESTIONABLE_TSUNAMI', tsunamiCauseCode='VOLCANO_AND_EARTHQUAKE', country='INDONESIA', maximumWaterHeight='3.0', numberOfRunup='1'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='VOLCANO_AND_EARTHQUAKE', country='TONGA', maximumWaterHeight='22.0', numberOfRunup='783'}
{ tsunamiEventValidity='PROBABLE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='ANTARCTICA', maximumWaterHeight='1.0', numberOfRunup='1'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='GREENLAND', maximumWaterHeight='200.0', numberOfRunup='3'}
{ tsunamiEventValidity='DEFINITE_TSUNAMI', tsunamiCauseCode='LANDSLIDE', country='USA', maximumWaterHeight='17.0', numberOfRunup='3'}

결과가 sorted_by_cause.csv 파일로 저장되었습니다.
```

Sort 테스트도 문제없이 진행된 것을 알 수 있다.




파일도 정상적으로 저장되었다.