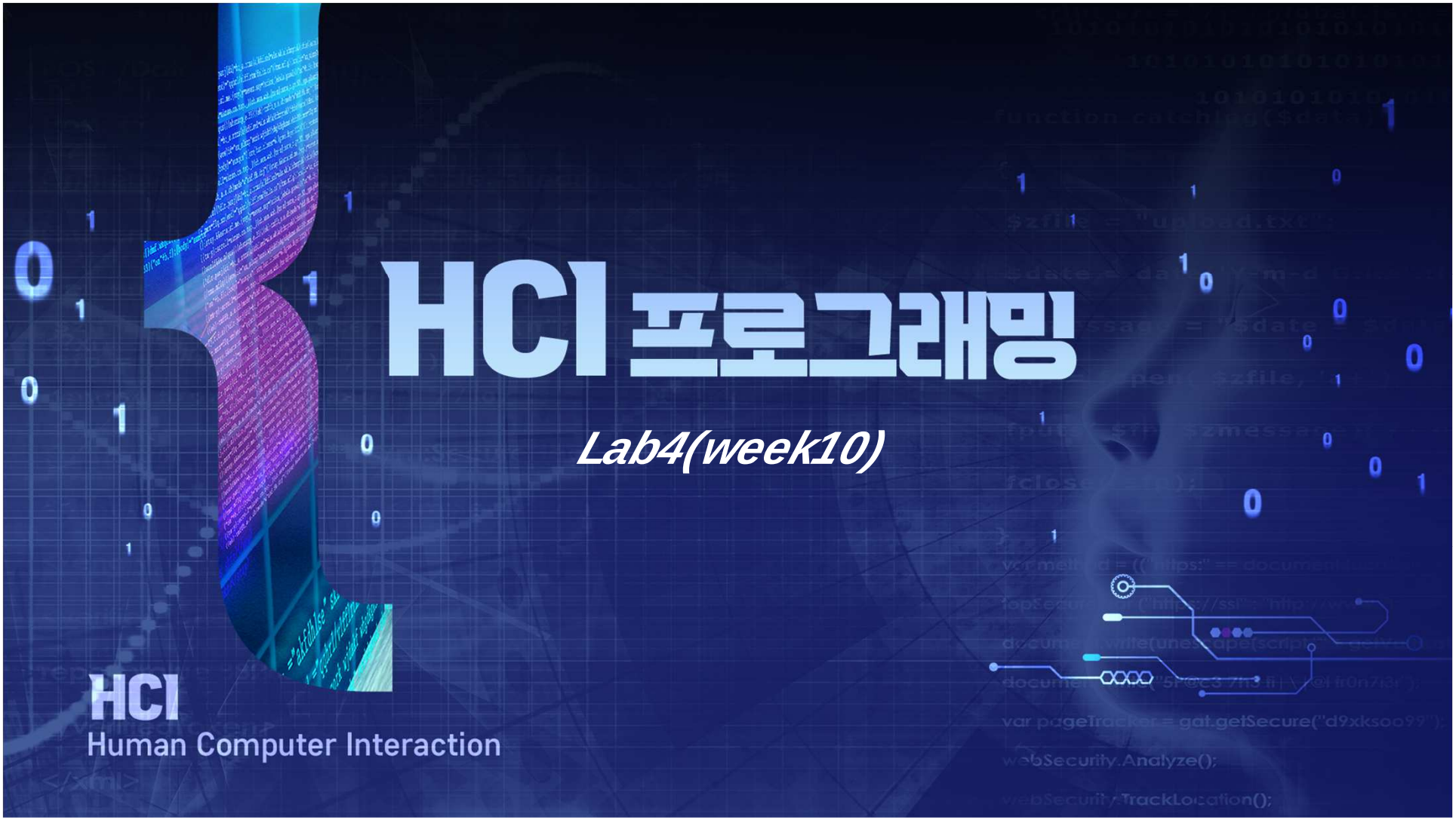
HCI 프로그래밍
Lab4(week10)
HCI
Human Computer Interaction

과제 Lab4

Interface, Collection

Lab4에서는 interface와 collection을 사용한다.

- WeatherData 클래스
- IWeatherCalculator 인터페이스
- WeatherCalculator 추상클래스
- DewPointCalculator, WindChillTemperatureCalculator, HeatIndexCalculator, DiscomfortIndexCalcualtor, 클래스
- WeatherCalculatorListManager 클래스
  - List〈WeatherCalculator〉 사용
- WeatherCalculatorSetManager 클래스
  - HashSet〈WeatherCalculator〉 사용
- Program 클래스

과제 Lab4

Interface, Collection

IEquatable<WeatherData>를 상속한 WeatherData 클래스를 작성한다.

- DateTime DateTime { get; set; }
- double Temperature { get; set; } // fahrenheit
- double RelativeHumidity { get; set; } // %
- double WindVelocity { get; set; }
- double Value { get; set; } // 계산값
- 생성자
- public override string ToString() 메소드
- public bool Equals(WeatherData other) 메소드
- public static bool operator ==(WeatherData x, WeatherData y) 메소드
- public static bool operator !=(WeatherData x, WeatherData y) 메소드
- public override int GetHashCode() 메소드

과제 Lab4

Interface, Collection

IWeatherCalculator 인터페이스 추가

- void PrintTable(); // 추상메소드
- void GetUserInput(); // 추상메소드
- void Calculate(); // 추상메소드

과제 Lab4

Interface, Collection

- WeatherCalculator 추상클래스는 IWeatherCalculator 인터페이스를 상속
  - protected WeatherData WeatherData { get; set; } // 날씨데이터 사용
  - protected double Value { get; set; } // 계산 값
  - protected WeatherCalculator(WeatherData weatherData) {
    WeatherData = weatherData;
    Calculate(); // 동적바인딩으로 자식클래스의 calculate() 호출
    Console.WriteLine(this);
    }
  - 나머지 동일

과제 Lab4

Interface, Collection

DewPointCalculator, WindChillTemperatureCalculator, HeatIndexCalculator, DiscomfortIndexCalculator 클래스는 WeatherCalculator 추상클래스를 상속받아 weatherData를 사용하여 추상메소드 내부 구현

- 각 클래스의 멤버필드는 weatherData 객체 사용

과제 Lab4

Interface, Collection

```
public WindChillTemperatureCalculator(WeatherData weatherData) : base(weatherData) {    }
public override void Calculate() {
    Value = Calculate(WeatherData.getTemperature(), WeatherData.getWindVelocity());
}
public override void PrintTable() { WCT 테이블 출력 }
public override void GetUserInput() { weatherData 사용자 입력 }
public override string ToString() {
    return "WindChillTemperatureCalculator [Temperature=" + WeatherData.Temperature + ",
    WindVelocity=" + WeatherData.WindVelocity+ ", Value=" + Value + ", Index=" + GetIndex(Value) + "]";
}
```

과제 Lab4

Interface, Collection

DiscomfortIndexCalculator 클래스를 작성한다.

- 생성자
- public override string ToString() 메소드
- public static double Calculate(double F, double RH) // 불쾌지수 계산 공식
  DI = T - 0.55*(1 - 0.01*RH)*(T - 14.5) [T: celsius]
- public void Calculate() // 멤버 필드 온도와 상대습도로 불쾌지수 계산
- public void PrintTable() // 불쾌지수 테이블 출력은 다음 데이터를 사용한다.
  - ✓int[] fahrenheit = {68, 71, 74, 77, 80, 83, 86, 89, 92, 95, 98, 101, 104, 107, 110};
  - ✓int[] humidities = {25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100};
- public void GetUserInput() // 불쾌지수 계산을 위한 온도(F), 상대습도(%) 사용자 입력

Discomfort Index

DiscomfortIndex 열거형

- EXTREME_DISCOMFORT 89.6 F (32 C ~)
- VERY_HIGH_DISCOMFORT 86 ~ 89.6 F (30 ~ 32 C)
- HIGH_DISCOMFORT 82.4 ~ 86 F (28 ~ 30 C)
- MODERATE_DISCOMFORT 77 ~ 82.4 F (25 ~ 28 C)
- LOW_DISCOMFORT 69.8 ~ 77 F (21 ~ 25 C)
- NO_DISCOMFORT ~ 69.8 F (~ 21 C)
- public static DiscomfortIndex GetIndex(double value) { 내부 구현 }

과제 Lab4

Interface, Collection

WeatherCalculatorFactory 클래스를 작성한다.

- public static WeatherCalculator GetInstance(int mode); // mode에 따른 DewPointCalculator, WindChillTemperatureCalculator, HeatIndexCalculator, DiscomfortIndexCalculator 객체 반환 메소드
- public static WeatherCalculator GetInstance(int mode, WeatherData data); // mode에 따른 DewPointCalculator(data), WindChillTemperatureCalculator(data), HeatIndexCalculator(data), DiscomfortIndexCalculator(data) 객체 반환 메소드

과제 Lab4

Interface, Collection

- WeatherCalculatorListManager,
  WeatherCalculatorSetManager 클래스

```
public class WeatherCalculatorListManager {
    private List<WeatherCalculator> calculators = null;
    public WeatherCalculatorListManager() : this(new List<WeatherCalculator>()) { }
    public WeatherCalculatorListManager(List<WeatherCalculator> calculators) {
        this.calculators = calculators;
    }
}

public class WeatherCalculatorSetManager {
    private ISet<WeatherCalculator> calculators = null;
    public WeatherCalculatorSetManager() : this(new HashSet<WeatherCalculator>()) { }
    public WeatherCalculatorSetManager(ISet<WeatherCalculator> calculators) {
        this.calculators = calculators;
    }
}
```

과제 Lab4

Interface, Collection

WeatherCalculatorListManager 클래스

- public void Add(WeatherCalculator c) // 리스트에 새로 추가
- public void Remove(WeatherCalculator c) // 리스트에서 해당 calculator 삭제
- public void Set(int index, WeatherCalculator c) // 리스트에서 index에 calculator로 교체
- Public WeatherCalculator Get(int index) // 리스트에서 index의 calculator 반환
- public int IndexOf(WeatherCalculator c) // 리스트에서 해당 calculator의 index 반환

과제 Lab4

Interface, Collection

WeatherCalculatorListManager 클래스

- public int Count { …. } // 리스트의 크기 반환
- public void Print() { …. . } // 전체 리스트 프린트
- public IList<WeatherCalculator> Select(Predicate<WeatherCalculator> predicate) // 리스트에서 해당 predicate 조건에 맞는 모든 calculator 리스트 반환
- public static WeatherCalculator GetRandom(IList<WeatherCalculator> list) // 인자로 넘겨준 리스트에서 랜덤하게 하나 선택해서 반환

과제 Lab4

Interface, Collection

WeatherCalculatorSetManager 클래스

- public void Add(WeatherCalculator c) // 세트에 새로 추가
- public void Remove(WeatherCalculator c) // 세트에서 해당 c 삭제
- public int Count { … } // 세트의 크기 반환
- public void Print() // 전체 세트 프린트
- public ISet<WeatherCalculator> select(Predicate<WeatherCalculator> predicate) // 리스트에서 해당 predicate 조건에 맞는 모든 calculator 세트 반환
- public static WeatherCalculator GetRandom(ISet<WeatherCalculator> set) // 인자로 보내준 셋에서 랜덤하게 하나 선택해서 반환

과제 Lab4

Interface, Collection

- HashSet을 위해 Equals와 GetHashCode override하여 같은 데이터임 명시함

```
public class WeatherCalculator : IWeatherCalculator, IEquatable<WeatherCalculator> {
    public bool Equals(WeatherCalculator other) {
        return WeatherData.Equals(other.WeatherData) && Value == other.Value;
    }
    public override int GetHashCode() {
        return Hashcode.Combine(WeatherData.Temperature, WeatherData.RelativeHumidity,
                                WeatherData.WindVelocity, Value);
    }
    public static bool operator ==(WeatherCalculator x, WeatherCalculator y) {
        return x.Equals(y);
    }
    public static bool operator !=(WeatherCalculator x, WeatherCalculator y) {
        return !x.Equals(y);
    }
...}
```

과제 Lab4

- Lab4 클래스를 테스트하고 더 많은 테스트를

```
static WeatherData[] weatherData = {
new WeatherData(new DateTime(2019, 1, 1), 30.38, 46, 4.0265), new WeatherData(new DateTime(2019, 2, 1), 33.8, 47, 4.0265),
new WeatherData(new DateTime(2019, 3, 1), 44.78, 51, 4.6976), new WeatherData(new DateTime(2019, 4, 1), 53.78, 51, 4.2502),
new WeatherData(new DateTime(2019, 5, 1), 66.92, 47, 4.6976), new WeatherData(new DateTime(2019, 6, 1), 72.5, 61, 3.8028),
new WeatherData(new DateTime(2019, 7, 1), 78.62, 69, 4.0265), new WeatherData(new DateTime(2019, 8, 1), 80.96, 69, 3.5791),
new WeatherData(new DateTime(2019, 9, 1), 72.68, 65, 4.9213), new WeatherData(new DateTime(2019, 10, 1), 61.52, 62, 4.6976),
new WeatherData(new DateTime(2019, 11, 1), 45.68, 56, 4.9213), new WeatherData(new DateTime(2019, 12, 1), 34.52, 58, 4.6976),
new WeatherData(new DateTime(2020, 1, 1), 34.88, 56, 4.6976), new WeatherData(new DateTime(2020, 2, 1), 36.5, 58, 5.1450),
new WeatherData(new DateTime(2020, 3, 1), 45.86, 46, 5.5924), new WeatherData(new DateTime(2020, 4, 1), 51.98, 50, 6.7108),
new WeatherData(new DateTime(2020, 5, 1), 64.4, 67, 5.3687), new WeatherData(new DateTime(2020, 6, 1), 75.02, 68, 5.1450),
new WeatherData(new DateTime(2020, 7, 1), 75.38, 77, 5.3687), new WeatherData(new DateTime(2020, 8, 1), 79.7, 85, 5.1450),
new WeatherData(new DateTime(2020, 9, 1), 70.52, 71, 5.5924), new WeatherData(new DateTime(2020, 10, 1), 57.74, 61, 4.6976),
new WeatherData(new DateTime(2020, 11, 1), 46.4, 64, 4.9213), new WeatherData(new DateTime(2020, 12, 1), 31.46, 58, 4.9213)
};
```

과제 Lab4

Lab4과 보고서 전체를 묶어서 e- learning에 과제 제출

- Lab3와 Lab4의 방식을 비교 분석한다.
- 배열과 List와 HashSet의 차이점을 비교 분석한다.