자료구조

Interface, Java Collection Framework

학습내용1

- Interface 이해하고 활용하기

- 자료구조와 Java Collection Framework

인터페이스 이용 예: 온도 변환기

- package org.cscnu.example1
  - Temperature.java : 인터페이스
  - MyTemperature.java: 상속구현
  - Convert.java: 테스트 드라이버

- Temperature interface
  - getCelsius()
  - getFahrenheit()
  - setCelsius()
  - setFahrenheit()

java-data-structure > temp-converter > src > main > java > org > cscnu > example1 > Temperature.j

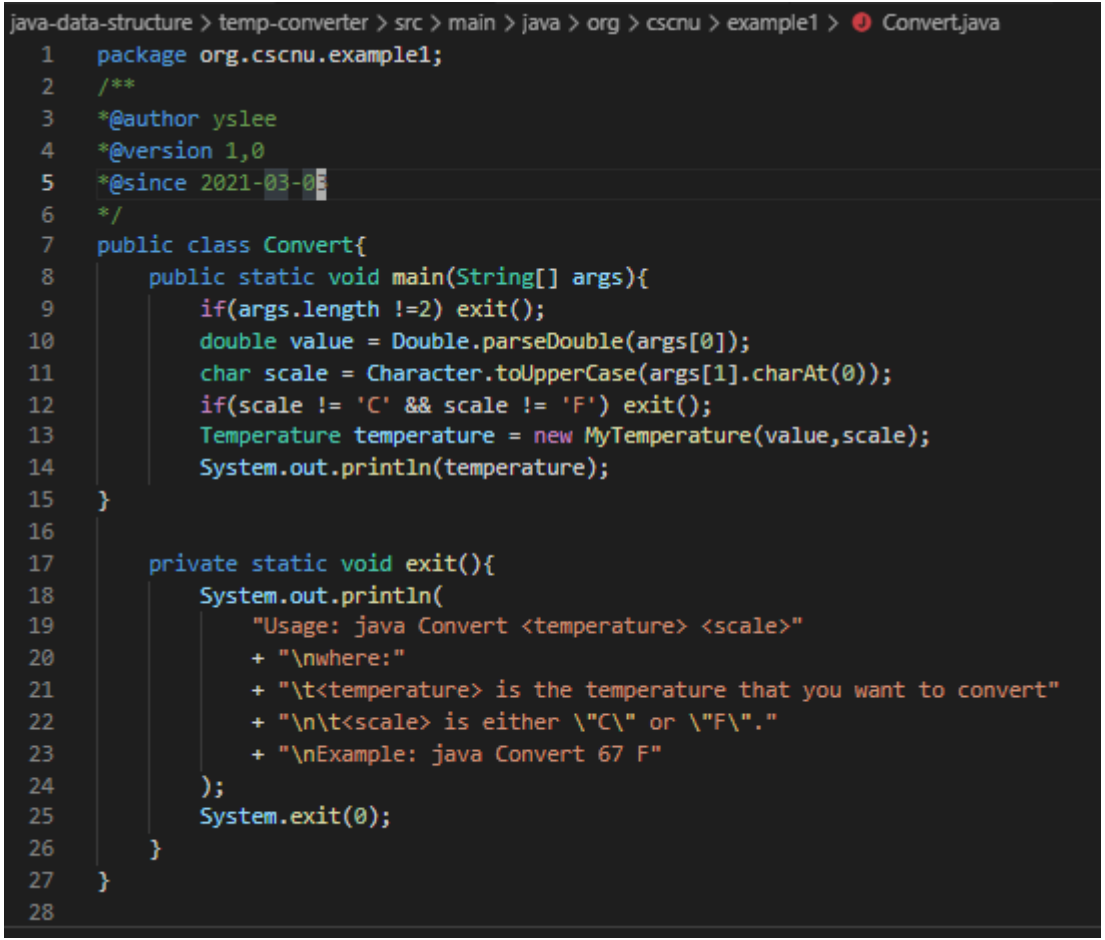
```java
package org.cscnu.example1;
/**
*@author yslee
*@see MyTemperature
*@since 2021-03-0
*/
public interface Temperature{
    /**@return Celsius */
    public double getCelsius();

    /**@return Fahrenheit **/
    public double getFahrenheit();

    /**@param celsius  **/
    public void setCelsius(double celsius);

    /**@param fahrenheit to celsius **/
    public void setFahrenheit(double fahrenheit);
}

```

인터페이스 이용 예: 온도 변환기 상속

- MyTemperature 클래스
  - Temperature 상속 구현
  - 생성자
  - get, set 메소드
  - toString()

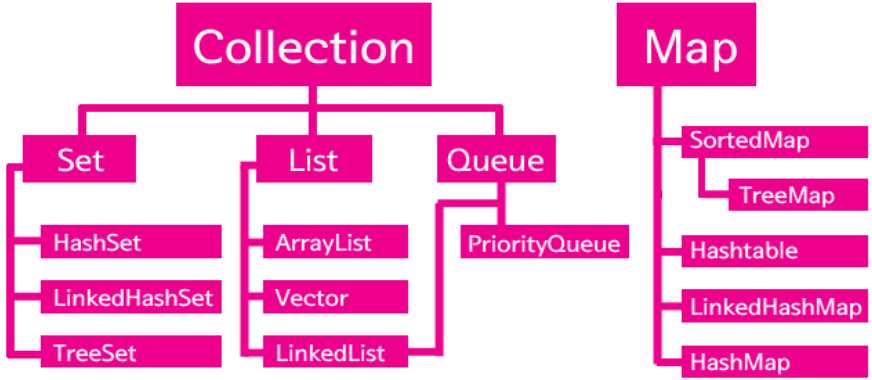
java-data-structure > temp-converter > src > main > java > org > cscnu > example1 > MyTemperature.java

```java
package org.cscnu.example1;
/**
*@author yslee
*@version 1.0
*@since 2021-03-04
*/

public class MyTemperature implements Temperature {
    private double celsius;  // storing values in celsius

    public MyTemperature(double value, char scale){
        if(scale=='C') setCelsius(value);
        else setFahrenheit(value);
    }

    public double getCelsius(){ /* return celius */
        return celsius;
    }

    public double getFahrenheit(){  /* return Farenheit */
        return 9*celsius/5 + 32.0;
    }

    public void setCelsius(double celsius){
        this.celsius = celsius;
    }

    public void setFahrenheit(double fahrenheit){ /* saving values in celsius */
```


java-data-structure > temp-converter > src > main > java > org > cscnu > example1 > Convert.java
1 package org.cscnu.example1;
2 /**
3 *@author yslee
4 *@version 1,0
5 *@since 2021-03-05
6 */
7 public class Convert{
8 public static void main(String[] args){
9 if(args.length !=2) exit();
10 double value = Double.parseDouble(args[0]);
11 char scale = Character.toUpperCase(args[1].charAt(0));
12 if(scale != 'C' && scale != 'F') exit();
13 Temperature temperature = new MyTemperature(value,scale);
14 System.out.println(temperature);
15 }
16
17 private static void exit(){
18 System.out.println(
19 "Usage: java Convert <temperature> <scale>"
20 + "\nwhere:"
21 + "\t<temperature> is the temperature that you want to convert"
22 + "\n\t<scale> is either \"C\" or \"F\"."
23 + "\nExample: java Convert 67 F"
24);
25 System.exit(0);
26 }
27 }
28

학습내용2

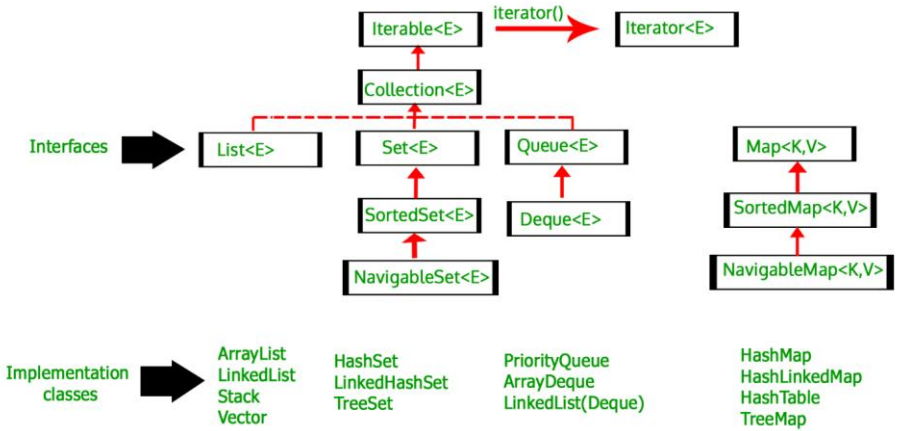
- Interface 이해하고 활용하기
  - 추상화: 온도 변환기
- 자료구조와 Collection Framework



https://www.opentutorials.org/course/1223/6446

Java Collection Framework 주요 인터페이스 간 상속관계

- <E> <K,V> -> 제네릭
- List<E>: 순서, 중복
- Set<E>
- Map<K, V>: key, value 쌍

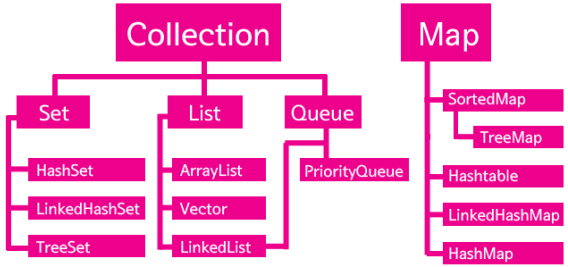


제네릭(Generic)

- 데이터타입의 일반화
  - 클래스나 메소드에서 사용할 내부 데이터타입을 컴파일할 때 지정

```
class MyArray<T> {
   T element;
   void setElement(T element) { this.element = element; }
   T getElement() { return element; }
}

MyArray<Integer> myArr = new MyArray<Integer>();
```

List, Set



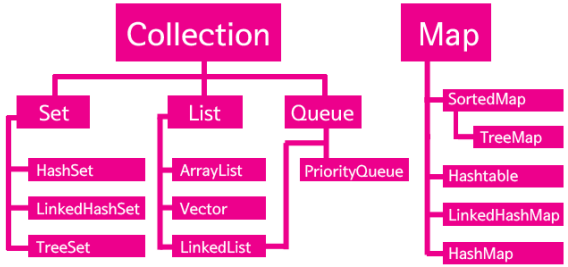
- List
  - import java.util.ArrayList;
  - 중복 O
- Set
  - import java.util.HashSet;
  - 중복 X

```
ArrayList<String> al = new ArrayList<String>();
al.add("one");
al.add("two");
al.add("two");
```

```
HashSet<Integer> A = new HashSet<Integer>();
A.add(1);
A.add(2);
A.add(3);
```

Map

- HashMap
  - (key, value)




```
HashMap<String, Integer> a = new HashMap<String, Integer>();
a.put("one", 1);
a.put("two", 2);
a.put("three", 3);
a.put("four", 4);
System.out.println(a.get("one"));
System.out.println(a.get("two"));
System.out.println(a.get("three"));
```

자바 객체 정렬하기: Comparable<T>

- 객체 정렬 방법: Comparable vs. Comparator interface 차이점은?
  - package: java.lang.Comparable (기본 적용), compareTo() 메소드 구현
  - package: java.util.Comparator (기본과 다른 정렬), compare() 메소드 구현

- package: java.lang.Comparable
  - 기본(default) 정렬 메소드 정의

    public final class Integer extends Number implements Comparable<Integer> { ... }
  - 구현 방법
    - 객체에 Comparable interface를 implements 후, compareTo() 메서드를 오버라이드
  - 사용 방법
    - Arrays.sort(array)
    - Collections.sort(list)

자바 객체 정렬하기: Comparator<T>

- package java.util.Comparator< T>
  - default가 아닌 다른 순서로 정렬할 때
  - 메소드 2번째 인자로 비교연산을 사용자가 지정할 수 있음
  - 구현 방법
    - Comparator interface를 implements 후 compare() 메서드를 오버라이드한 myComparator class를 작성
    - 익명 클래스로 작성
  - 사용 방법
    - Arrays.sort(array, myComparator)
    - Collections.sort(list, myComparator)

Wrapper 클래스의 객체 정렬

- Integer 클래스 정렬 -> Collections.sort()
  - https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html

```
public class testSimpleSort {
    Run | Debug
    public static void main (String[] args) throws java.lang.Exception
    {
        List<Integer> list = new ArrayList<Integer>();

        list.add(3);
        list.add(5);
        list.add(4);
        list.add(8);
        list.add(1);

        Collections.sort(list);

        System.out.println(list.toString());
    }
}
```

학생 클래스 정렬(아이디 순서)

```java
class Student implements Comparable<Student> {
    String name;
    int id;
    int score;

    public Student(String name, int id, int score) {
        this.name = name;
        this.id = id;
        this.score = score;
    }
    public String getName() {
        return this.name;
    }
    public int getId() {
        return this.id;
    }
    public int getScore() {
        return this.score;
    }
    @Override
    public int compareTo(Student s) {
        if (this.id < s.getId()) {
            return -1;
        } else if (this.id > s.getId()) {
            return 1;
        }
        return 0;
    }
```

```java
public class ListSort {
    Run | Debug
    public static void main (String[] args) throws java.lang.Exception
    {
        List<Student> list = new ArrayList<Student>();

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine());
        for (int i = 0; i < n; i++) {
            String [] str = br.readLine().split(" ");
            String name   = str[0];
            int year      = Integer.parseInt(str[1]);
            int score     = Integer.parseInt(str[2]);
            list.add(new Student(name, year, score));
        }
        br.close();

        Collections.sort(list);

        Iterator iter = list.iterator();
        while (iter.hasNext()) {
            Student str = (Student)iter.next();
            System.out.println(str.name +" "+str.id+ " " + str.score);
        }

        Comparator<Student> myComparator = new Comparator<Student>() {
            @Override
            public int compare(Student s1, Student s2) {
                return s2.getScore() - s1.getScore();
            }
        };
        Collections.sort(list, myComparator);

        iter = list.iterator();

        while (iter.hasNext()) {
            Student str = (Student)iter.next();
            System.out.println(str.name +" "+str.id+ " " + str.score);
        }

    }
}
```

정리

- 자바 interface 이용 객체지향 프로그래밍
- Java collection framework 활용