과제3) 자바 람다식과 스트림

1. 객체지향 언어에서 인터페이스는 그 시그너처와 선언이 변하지 않는다는 전제로 하여 다형성을 정의하는 객체간의 규약입니다. 자바8의 인터페이스는 인터페이스를 업그레이드하는 개념을 도입하였습니다. 새롭게 변경된 인터페이스에 대한 내용을 각각 기술하세요. (30점)
   1. 인터페이스에서 구현 메서드를 작성할 수 있는 방법을 기술하세요.

인터페이스에서 구현 메서드를 작성하기 위해서는 인터페이스를 implements한 일반클래스들에서 구현해야 합니다.

그리고 인터페이스를 implements한 클래스들은 인터페이스의 규격에 맞도록 재정의하는 과정을 거칩니다.

* 1. 자바9 에서 새롭게 추가된 구현 메서드를 작성할 수 있는 방법을 기술하세요.

자바 9에서 새롭게 추가된 구현 메서드를 작성할 수 있는 방법은 인터페이스에 private 메소드를 사용하는 것 입니다. 이전에는 인터페이스를 수정하는 것에 어려움이 있었지만 private 메소드를 통해 인터페이스에 대한 캡술화를 유지할 수 있게 되어 인터페이스 내부의 코드 재사용성을 증가시켰습니다.

* 1. 함수형 인터페이스에 대해서 설명하세요.

함수형 인터페이스는 1개의 추상메소드를 가지고 있는 인터페이스로 FunctionalInterface 애노테이션을 가지고 있는 인터페이스 입니다. 이 인터페이스는 함수를 메소드로 전달하기 위해 java8부터 새롭게 정의되었으며 이를 람다 표현식이라고 부르기도 합니다.

함수형 인터페이스를 통해 람다식으로 생성한 객체에 접근 할 수 있고, 함수형 클래스와 람다식으로 익명 클래스를 간단하게 표기하면서 작성하는 코드의 수를 줄일 수 있었습니다.

1. 문제로 제공되는 stream-data.txt 파일에 대해서 다음을 각각 수행하세요. (예외는 별도로 처리하지 않습니다.) (50점)
   1. stream-data.txt 파일은 하나의 숫자를 행으로 가지고 있는 텍스트 파일입니다. 각 행을 읽어서 Stream<String> 타입으로 작성하세요.

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Stream;

public class Test2 {

public static void main(String[] args) throws IOException {

// 파일 불러오기

BufferedReader reader = new BufferedReader(

new FileReader("C:/intellij/IdeaProjects/java\_study/src/Homework/files/stream-data.txt")

);

String str;

// list

List<String> strArr = new ArrayList<>();

while ((str = reader.readLine()) != null){

strArr.add(str);

}

reader.close();

// stream

Stream<String> stream = strArr.stream();

stream.forEach(s -> System.out.println(s));

}

}

* 1. A를 IntStream 타입으로 변환하는 코드를 작성하세요.

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class Test2 {

public static void main(String[] args) throws IOException {

// 파일 불러오기

BufferedReader reader = new BufferedReader(

new FileReader("C:/intellij/IdeaProjects/java\_study/src/Homework/files/stream-data.txt")

);

String str;

// list

List<String> strArr = new ArrayList<>();

while ((str = reader.readLine()) != null){

strArr.add(str);

}

reader.close();

// stream

Stream<String> stream = strArr.stream();

// stream.forEach(s -> System.out.println(s));

List<Integer> integerList = stream.mapToInt(Integer::parseInt)

.boxed().collect(Collectors.toList());

System.out.println(integerList);

}

}

* 1. B 에서 작성된 코드에서 map 중간연산을 사용해서 각 요소를 2배로 해서 새롭게 스트림을 생성합니다.
  2. C에서 생성된 스트림에서 50미만의 숫자는 제외하고 새롭게 스트림을 생성합니다.
  3. 최종적으로 모든 요소의 합을 구해서 출력합니다. (System.out.println)

1. 2번 문제를 스트림을 사용하지 않고 for 문을 사용해서 결과를 도출하는 방식을 Imperative Programming(명령형 프로그래밍, 스트림을 사용하여 처리하는 방식을 선언형이라고 함) 이라고 한다. 다음을 각각 수행할 수 있도록 코드를 작성합니다. (예외는 별도로 처리하지 않습니다.) (20점)
   1. stream-data.txt 파일을 읽어 List<Integer> 혹은 ArrayList<Integer> 타입으로 데이터를 읽어옵니다.

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class Test2 {

public static void main(String[] args) throws IOException {

// 파일 불러오기

BufferedReader reader = new BufferedReader(

new FileReader("C:/intellij/IdeaProjects/java\_study/src/Homework/files/stream-data.txt")

);

String str;

// list

List<Integer> integerList = new ArrayList<>();

while ((str = reader.readLine()) != null){

// System.out.println(str);

int a = Integer.parseInt(str);

integerList.add(a);

}

System.out.println(integerList);

reader.close();

}

}

* 1. for 문을 사용하여 요소 중 50을 초과하는 요소의 총 합을 구해서 출력합니다. (각 요소를 2배로 곱하지 않습니다.)

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class Test2 {

public static void main(String[] args) throws IOException {

// 파일 불러오기

BufferedReader reader = new BufferedReader(

new FileReader("C:/intellij/IdeaProjects/java\_study/src/Homework/files/stream-data.txt")

);

String str;

// list

List<Integer> integerList = new ArrayList<>();

while ((str = reader.readLine()) != null){

int a = Integer.parseInt(str);

integerList.add(a);

}

System.out.println(integerList);

reader.close();

int sum = 0;

for (int i = 0; i < integerList.size(); i++) {

if (integerList.get(i) > 50){

sum += integerList.get(i);

}

}

System.out.println(sum);

}

}