객체지향언어와 실습

실습 12주차

엄진영 교수님

담당조교 박선희
동국대학교 컴퓨터공학과
seonhuibag1228@gmail.com
010-4065-3024

실습문제 1-1 Student 클래스

- **Student 클래스**

```
import java.util.*;
class Student implements Comparable {
    String name;
    int ban;
    int no;
    int kor, eng, math;

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban = ban;
        this.no = no;
        this.kor = kor;
        this.eng = eng;
        this.math = math;
    }
    int getTotal() {
        return kor+eng+math;
    }
    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }
    public String toString() {
        return name +","+ban +","+no +","+kor +","+eng +","+math
                +","+getTotal() +","+getAverage();
    }
    public int compareTo(Object o) {
        if(o instanceof Student) {
                Student tmp = (Student)o;
                return name.compareTo(tmp.name); // String클래스의 compareTo()를 호출
        } else {
            return -1;
        }
    }
}
```

실습문제 1-2 Main 함수

- Main 함수

```
public class Source1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList list = new ArrayList();
        list.add(new Student("홍길동",1,1,100,100,100));
        list.add(new Student("남궁성",1,2,90,70,80));
        list.add(new Student("김자바",1,3,80,80,90));
        list.add(new Student("이자바",1,4,70,90,70));
        list.add(new Student("안자바",1,5,60,100,80));
        Collections.sort(list); // list에 저장된 요소들을 정렬한다.
        Iterator it = list.iterator();
        while(it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

- 결과 화면

```
김자바,1,3,80,80,90,250,83.3
남궁성,1,2,90,70,80,240,80.0
안자바,1,5,60,100,80,240,80.0
이자바,1,4,70,90,70,230,76.7
홍길동,1,1,100,100,100,300,100.0
```

실습문제 1 Generics

- 다음 장의 코드는 Comparable 인터페이스를 구현하도록 변경해서 Student클래스에서 이름(name)이 기본 정렬기준이 되도록 하는 compareTo 메소드를 지니고 있는 코드이다. 제네릭를 적용하시오. Comparable의 타입 매개변수로 Student 로 지정하여 결과화면과 같은 결과가 출력되도록 코드를 수정하시오.

```
class Student implements Comparable<Student>
```

  - ArrayList에 저장된 요소들은 모두 Comparable인터페이스를 구현한 것이어야 한다. 이 인터페이스에는 compareTo메소드가 정의되어 있는데, 이 메서드는 Collections.sort(List list)에 의해 호출되어 정렬기준을 제공하게 된다.
  - compareTo메서드는 매개변수로 주어진 객체(o)를 인스턴스 자신과 비교해서 자신이 작으면 음수를, 같으면 0을, 크면 양수를 반환하도록 구현되어야 한다.

  - 산출 파일 이름 : Source1.java

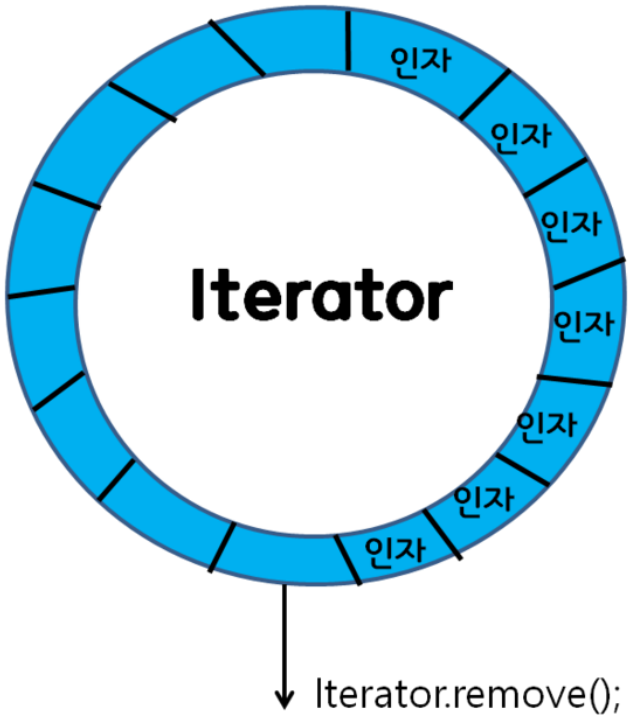
실습문제 1 Generics

- **Collections.sort**

```
public class Main {

    public static void main(String[] args){

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

        nameList.add("Heo");
        nameList.add("Choi");
        nameList.add("Lee");

        System.out.println("- 정렬 전" + nameList);

        Collections.sort(nameList);

        System.out.println("- 정렬 후" + nameList);


    }// main()
}// class
```

```
- 정렬 전[Heo, Choi, Lee]
- 정렬 후[Choi, Heo, Lee]
```

- **Iterator**



hasNext() : 남은 값이 있으면 참 (1)

실습문제 2 와일드 카드

- 과일박스에 과일을 담고 주스를 만들어보는 프로그램을 만들어 본다. 아래와 같은 메인 함수를 작성하였을 때, 결과화면과 동일하게 출력하도록 조건에 맞게 클래스를 작성하시오.

  - 메인함수

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    FruitBox<Fruit> fruitBox = new FruitBox<Fruit>();
    FruitBox<Apple> appleBox = new FruitBox<Apple>();


    fruitBox.add(new Apple());
    fruitBox.add(new Grape());
    appleBox.add(new Apple());
    appleBox.add(new Apple());

    System.out.println(Juicer.makeJuice(fruitBox));
    System.out.println(Juicer.makeJuice(appleBox));
}
```

  - 결과화면

```
Apple Grape Juice
Apple Apple Juice
```

  - 산출 파일 이름 : Source2.java

실습문제 2-1 Fruit, Apple, Grape 클래스

- Fruit 클래스를 부모클래스로 갖는 자식 클래스 Apple, Grape를 설계하시오.

- 세 개의 클래스는 각각 “Fruit”, “Apple”, “Grape”를 반환하는 toString() 메소드를 지니고 있다.

실습문제 2-2 Juice, Juicer 클래스

- Juice 클래스

```
class Juice{
    String name;
    Juice(String name) {this.name=name+"Juice";}
    public String toString() {return name;}
}
```

- Juicer 클래스는 매개변수에 과일박스를 대입하면 주스를 만들어서 반환하는 makeJuicer() 라는 static 메소드가 다음과 같이 정의되어 있다. 와일드 카드를 사용하여 매 개 변 수 로 FruitBox<Fruit>, FruitBox<Apple>, FruitBox<Grape>가 가능하게끔 빈칸을 완성하시오.

```
static Juice makeJuice(FruitBox 빈칸을 채우시오 box) {
    String tmp="";
    for(Fruit f:box.getList())
        tmp+=f+" ";
    return new Juice(tmp);
}
```

실습문제 2-3 Box, FruitBox 클래스

- Box 클래스

```
class Box<T>{
    ArrayList <T> list= new ArrayList<T>();
    빈칸을 채우시오
    public String toString() {return list.toString();}
}
```

- 다음 네개의 메소드를 빈칸에 추가하시오.
  - add : list 에 매개변수를 저장한다.
  - get : list의 i번째 요소를 반환한다.
  - getList : list를 반환한다.
  - size : list 사이즈를 반환한다.

- FruitBox 클래스는 제한된 제레닉 클래스 이다. Fruit의 자손인 클래스만 타입 매개변수로 받게끔 제한한다. 또한 Box 클래스를 상속받고 있다.

```
class FruitBox 빈칸을 채우시오 {}
```

실습문제 3 Annotation

- 1. 다음 장의 코드를 컴파일 하였을 때 아래와 같은 메시지가 나타나게끔 수정하시오.

```
Note: DeprecatedAnnotation.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

- 2. 1번에서 수정한 코드에서 위 메시지가 나타나지 않게끔 수정하시오. 또한 ArrayList 객체를 생성한 곳에 제네릭 관련 경고를 억제하시오.

실습문제 3 코드

- 수정할 코드

```
import java.util.ArrayList;

class NewClass{
    int newField;
    int getNewField() {
        return newField;
    }

    int oldField;
    int getOldField() {
        return oldField;
    }
}
public class Source3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        NewClass nc=new NewClass();
        nc.oldField=10;
        System.out.println(nc.getOldField());

        ArrayList<NewClass> list =new ArrayList();
        list.add(nc);
    }

}
```

- 산출 파일 이름 : Source3.java

제출 시 유의사항

- 기한 : 2019년 5월 26일 (일) 23:59까지
- 제출 파일 형식
  - 제출 시, *.java파일과 보고서를 압축하여
    [n주차]_[학번]_[이름].zip 파일로 압축하여 제출
    ex > 1주차_2016xxxxxx_박선희.zip
  - 보고서는 소스코드와 실행화면을 캡쳐하고, 간단히 분석하여 제출
  - 코드 시작 부분에 주석을 이용하여 과, 학번, 이름 및 문제번호를 적을 것
- 기타 문의: seonhuibag1228@gmail.com 으로 문의