(고급프로그래밍 과제4)

11/06/2021 (MM/DD/YYYY)

32212488 안태현 (고급프로그래밍 1분반)

Sections

Section 1. Description

『Java 명품 프로그래밍』(개정 4판) p.436의 5번 문제를 기반으로 한다.

Q1) 다음의 요구 사항을 반영하여 5번의 (1)의 내용을 구현하시오. ***(단, 학생들의 정렬과 검색은 Collections 클래스의 sort()와 binarySearch() 메소드를 이용하시오.)***

<요구사항>

(A) 키보드 입력 대신 텍스트 파일로부터 학생 데이터를 읽을 수 있도록 구현한다. 학생들의 정보는 콤마(,)로 구분하고, 파일에는 10명 이상의 학생 정보를 포함한다.

(B) 읽어들인 데이터를 출력할 때 학점 평균으로 정렬하여 출력한다. 정렬 결과를 보일 때는 한 화면에 많은 학생들 정보를 보이도록 한 줄에 한 학생의 정보를 제시한다.

Q2) 1의 요구사항들을 반영하도록 5번의 (2)의 내용을 구현하시오. ***(단, 검색은 HashMap의 키를 사용하여 구현하시오.)***

Q3) 수업 시간에 설명한 GenericList 클래스를 사용하여 1의 요구사항을 구현하시오. ***(단, 학생들의 정렬과 검색은 Arrays 클래스의 sort()와 binarySearch() 메소드를 이용하시오.)***

※문제 1, 2, 3에 대하여 각각 이름을 붙였는데, 다음과 같다.

문제 1: ArrayList Implementation Problem
문제 2: HashMap Implementation Problem
문제 3: GenericList Implementation Problem

5. 하나의 학생 정보를 나타내는 Student 클래스에는 이름, 학과, 학번, 학점 평균을 저장하는 필드가 있다.

(1) 학생마다 Student 객체를 생성하고 4명의 학생 정보를 ArrayList<Student> 컬렉션에 저장한 후에, ArrayList<Student>의 모든 학생(4명) 정보를 출력하고 학생 이름을 입력받아 해당 학생의 학점 평균을 출력하는 프로그램을 작성하라.

```
학생 이름, 학과, 학번, 학점평균 입력하세요.
>> 황기태, 모바일, 1, 4.1
>> 이재문, 안드로이드, 2, 3.9
>> 김남윤, 웹공학, 3, 3.5
>> 최찬미, 빅데이터, 4, 4.25
---------------------------
이름:황기태
학과:모바일
학번:1
학점평균:4.1
---------------------------
이름:이재문
학과:안드로이드
학번:2
학점평균:3.9
---------------------------
이름:김남윤
학과:공학
학번:3
학점평균:3.5
---------------------------
이름: 최찬미
학과:빅데이터
학번:4
학점평균:4.25
---------------------------
학생 이름 >> 최찬미
최찬미, 빅데이터, 4, 4.25
학생 이름 >> 이재문
이재문, 안드로이드, 2, 3.9
학생 이름 >> 그만
```

(2) ArrayList<Student> 대신, HashMap<String, Student> 해시맵을 이용하여 다시 작성하라. 해시맵에서 키는 학생 이름으로 한다.

▲ *『Java 명품 프로그래밍』(개정 4판) p.436*

Section 2. File Contents

모든 문제에 공통으로 쓰이는 파일이다.

info.txt (UTF-8 encoded)

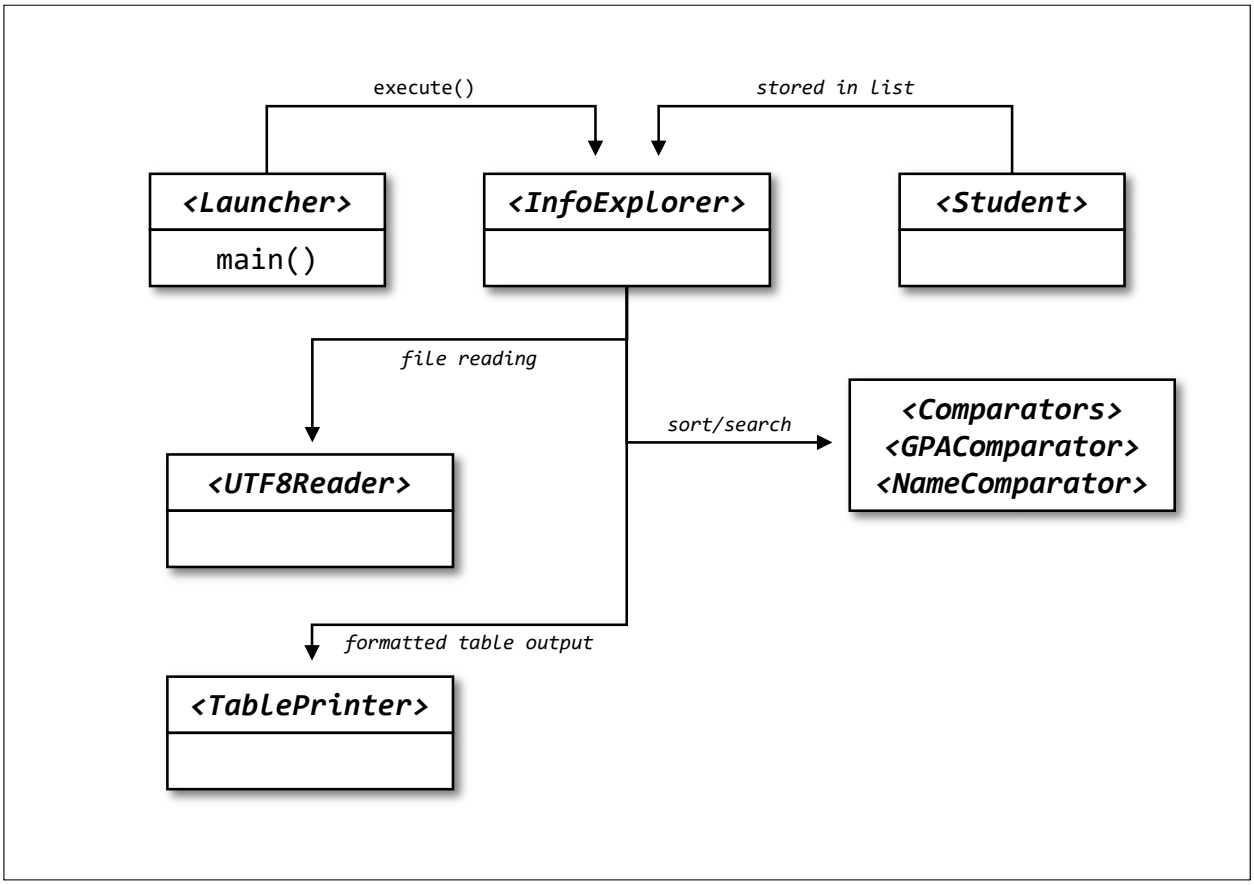
```
Unnamed Student 1,Web Development,40967547,1.01
Unnamed Student 2,Computer Engineering,40965324,1.98
Unnamed Student 3,Computer Engineering,40962249,1.13
Unnamed Student 4,Mobile Systems,40963562,2.12
Unnamed Student 5,Mobile Systems,40963723,0.81
Unnamed Student 6,Artificial Intelligence,40967425,4.49
Unnamed Student 7,Web Development,40969372,1.65
Unnamed Student 8,Web Development,40969012,2.52
Unnamed Student 9,Computer Engineering,40965118,4.03
Unnamed Student 10,Computer Engineering,40968641,1.34
Unnamed Student 11,Web Development,40962691,1.12
Unnamed Student 12,Mobile Systems,40963240,4.27
Unnamed Student 13,Mobile Systems,40963389,0.70
Unnamed Student 14,Artificial Intelligence,40961136,3.05
Unnamed Student 15,Artificial Intelligence,40964521,0.70
Unnamed Student 16,Mobile Systems,40968642,0.85
```

Section 3. ArrayList Implementation

구현할 프로그램은 학생들의 정보를 읽어들여 정렬하고, 검색하는 프로그램이다.

Section 3.1. Class Structure

사용하는 모든 클래스들의 관계를 아래 그림으로 살펴보자.



프로그램의 실행 순서는 가장 높은 단계에서 다음과 같다.

1. Launcher의 main()에서 InfoExplorer의 인스턴스를 생성한다.
2. 생성한 인스턴스에 대하여 execute() 메소드를 호출한다.

InfoExplorer의 execute() 구현은 아래와 같다.

```
public void execute() {
    printVersion();
    loadStudents();
    sortStudents(Comparators.GPA_COMPARATOR);
    printAllStudents();
```

```
    sortStudents(Comparators.NAME_COMPARATOR);
    runFinder();
    exit();
}
```

메소드의 이름으로부터 execute()의 과정을 짐작할 수 있는데, 좀 더 풀어쓰면 다음과 같다.

1. 학생 탐색기의 버전을 출력한다.
2. 파일로부터 학생들의 정보를 불러와 저장한다.
3. 저장된 학생들을 학점 순서대로 정렬한다.
4. 모든 학생들을 출력한다.
5. 저장된 학생들을 이름 순서대로 정렬한다.
6. 검색 기능을 실행한다.
7. 탐색기를 종료한다.

Section 3.2. Execution

```
Student Info Explorer: ARRAYLIST_IMPLEMENTATION

Name               | Major                   | Student ID | Grade Point Average
Unnamed Student 13 | Mobile Systems          | 40963389   | 0.7
Unnamed Student 15 | Artificial Intelligence | 40964521   | 0.7
Unnamed Student 5  | Mobile Systems          | 40963723   | 0.81
Unnamed Student 16 | Mobile Systems          | 40968642   | 0.85
Unnamed Student 1  | Web Development         | 40967547   | 1.01
Unnamed Student 11 | Web Development         | 40962691   | 1.12
Unnamed Student 3  | Computer Engineering    | 40962249   | 1.13
Unnamed Student 10 | Computer Engineering    | 40968641   | 1.34
Unnamed Student 7  | Web Development         | 40969372   | 1.65
Unnamed Student 2  | Computer Engineering    | 40965324   | 1.98
Unnamed Student 4  | Mobile Systems          | 40963562   | 2.12
Unnamed Student 8  | Web Development         | 40969012   | 2.52
Unnamed Student 14 | Artificial Intelligence | 40961136   | 3.05
Unnamed Student 9  | Computer Engineering    | 40965118   | 4.03
Unnamed Student 12 | Mobile Systems          | 40963240   | 4.27
Unnamed Student 6  | Artificial Intelligence | 40967425   | 4.49

Student name: Unnamed Student 9
Name                : Unnamed Student 9
Major               : Computer Engineering
Student ID          : 40965118
Grade Point Average : 4.03

Student name: Unnamed Student 100
'Unnamed Student 100' does not exist in the list.

Student name: exit
```

Section 4. HashMap Implementation

ArrayList Implementation Problem에서 List 대신에 HashMap을 사용하여 구현하는 문제이다.

Section 4.1. Problem

ArrayList Implementation Problem의 요구사항을 그대로 반영해야 하므로 HashMap을 정렬해야 한다. 그런데 HashMap은 List 타입이 아니기 때문에 Collections.sort(List<T>, …) 메소드를 사용할 수 없다. 게다가 HashMap API를 참조해보면, 다음과 같은 문장을 확인할 수 있다.

> This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time.

즉, HashMap에 저장되어 있는 원소들의 순서가 일정하게 유지된다는 보장이 없다. 따라서 HashMap의 정렬은 의미가 없는 작업이다.

Section 4.2. Solution

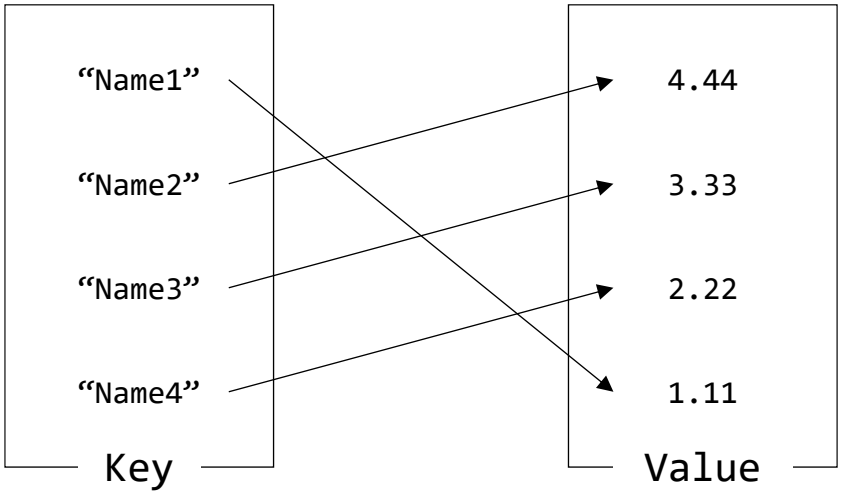
자바 상속 트리를 타고 한 단계 내려가보자. HashMap을 확장하는 클래스 중 'LinkedHashMap'이라는 클래스가 있다. LinkedHashMap API를 참조해보면, 다음과 같은 문장을 확인할 수 있다.

> Hash table and linked list implementation of the Map interface, with predictable iteration order. This implementation differs from HashMap in that it maintains a doubly-linked list running through all of its entries. This linked list defines the iteration ordering, which is normally the order in which keys were inserted into the map (*insertion-order*). Note that insertion order is not affected if a key is *re-inserted* into the map.

즉, LinkedHashMap은 이중 연결 리스트 형식으로 구현되어 있으므로 원소들의 순서가 보장되며, 일반적으로 삽입 순서와 동일하다. 또한, 같은 키를 재삽입해도 원소들의 순서는 변함이

없다. 따라서 HashMap보다 LinkedHashMap을 사용하는 것이 정렬에 더 적합함을 알 수 있다.

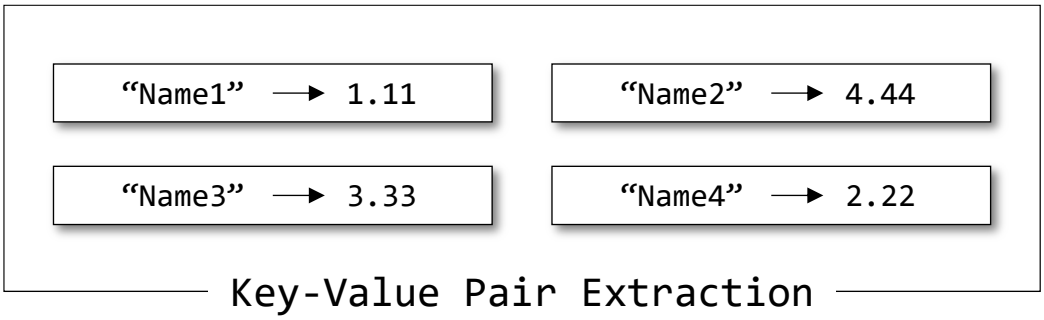
어떤 LinkedHashMap이 아래와 같은 매핑 관계를 가진다고 가정하고, 정렬의 과정을 살펴보자.



먼저 맵의 key-value 쌍을 추출해야 한다. Map 인터페이스의 API를 살펴보면, 다음과 같은 메소드가 있음을 확인할 수 있다.

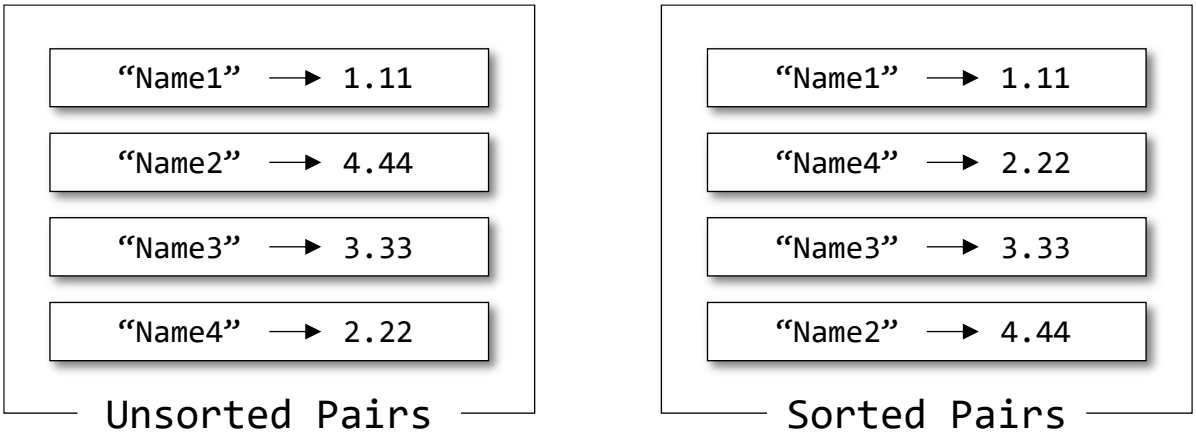
```
Set<Map.Entry<K,V>> entrySet()
Returns a Set view of the mappings contained in this map. The set is backed by the map,
so changes to the map are reflected in the set, and vice-versa.
```

Map.Entry*(key-value 쌍을 나타내는 인터페이스)* 타입의 집합을 반환하는 메소드이므로, LinkedHashMap에 매핑된 모든 쌍들을 추출할 수 있다. 각 쌍들을 추출한 결과는 아래와 같다.



집합은 정렬할 수 없기 때문에 이 집합의 원소들을 리스트에 추가한 후 각 쌍의 value에 대하여 정렬한다. *(이때 리스트에*

저장된 쌍들을 비교하기 위한 Comparator는 Map.Entry의 comparingByValue() 메소드의 반환 인스턴스를 이용한다.)



정렬된 리스트의 쌍들을 새로운 LinkedHashMap에 순서대로 추가하면, 정렬된 HashMap을 얻는다.

Section 4.2.1. Solution Code Implementation

4.2절에서 설명한 방법을 자바 제네릭 코드로 구현하면 아래와 같다.

```
public class Maps {
    //This class consists exclusively of static methods that operate on or return
maps

    public static <K, V> List<Map.Entry<K, V>> getMappingsAsList(Map<K, V> map) {
        List<Map.Entry<K, V>> mappings = new ArrayList<>();
        mappings.addAll(map.entrySet());
        return mappings;
    }

    public static <K, V> Map<K, V> getListAsOrderedMap(List<Map.Entry<K, V>>
mappings) {
        Map<K, V> map = new LinkedHashMap<>();
        for (Map.Entry<K, V> mapping : mappings) {
            map.put(mapping.getKey(), mapping.getValue());
        }

        return map;
    }

    public static <K, V> Map<K, V> sortByValues(Map<K, V> unsorted, Comparator<?
super V> comparator) {
        List<Map.Entry<K, V>> mappings = getMappingsAsList(unsorted);
        mappings.sort(Map.Entry.comparingByValue(comparator));
```

```
        return getListAsOrderedMap(mappings);
    }
}
```

Maps*(java.util.Collections와 비슷하게 지은 이름이다.)* 클래스의 sortByValues() 메소드를 이용하여 정렬된 Map 인스턴스를 반환받을 수 있다.

sortByValues() 메소드를 들여다보자. 우선 2개의 매개변수를 확인할 수 있다. *(정렬되지 않은 Map, 그리고 맵의 value를 비교하는 Comparator이다.)*

```
Map<K, V> unsorted, Comparator<? super V> comparator
```

(key-value쌍을 추출하여 리스트에 추가하는 부분)

```
List<Map.Entry<K, V>> mappings = getMappingsAsList(unsorted);
```

(리스트를 쌍의 value에 대하여 정렬하는 부분)

```
mappings.sort(Map.Entry.comparingByValue(comparator));
```

(리스트의 쌍을 새로운 LinkedHashMap에 추가하여 반환하는 부분)

```
return getListAsOrderedMap(mappings);
```

Section 4.2.2. Time Complexity

HashMap은 정렬에 적합한 클래스가 아니기 때문에 4.2.1절에서 설명한 코드의 시간 복잡도를 계산해볼 필요가 있다.

정렬해야 할 Map에 N개의 쌍이 존재한다고 가정하자. 4.2.1절에서 구현한 코드는 4단계의 과정을 거쳐 Map을 정렬한다. 각 단계의 시간 복잡도를 $O(N_1)$, $O(N_2)$, $O(N_3)$, $O(N_4)$라 하고 계산해보자.

첫번째 단계에서는 N개의 key-value쌍을 추출한다.

$$\therefore N_1 \leq N$$

두번째 단계에서는 리스트에 N개의 쌍을 추가한다.

$$\therefore N_2 = N$$

세번째 단계에서는 리스트를 정렬한다. List API를 참고해보면 sort() 메소드에 대하여 아래와 같은 내용을 확인할 수 있다.

```
This implementation is a stable, adaptive, iterative mergesort that requires far fewer than n lg(n) comparisons when the input array is partially sorted, while offering the performance of a traditional mergesort when the input array is randomly ordered.
```

일종의 개선된 합병 정렬 알고리즘을 사용한다고 명시되있다.

$$\therefore N_3 \leq NlogN$$

네번째 단계에서는 리스트의 자료를 LinkedHashMap에 추가한다.

$$\therefore N_4 = N$$

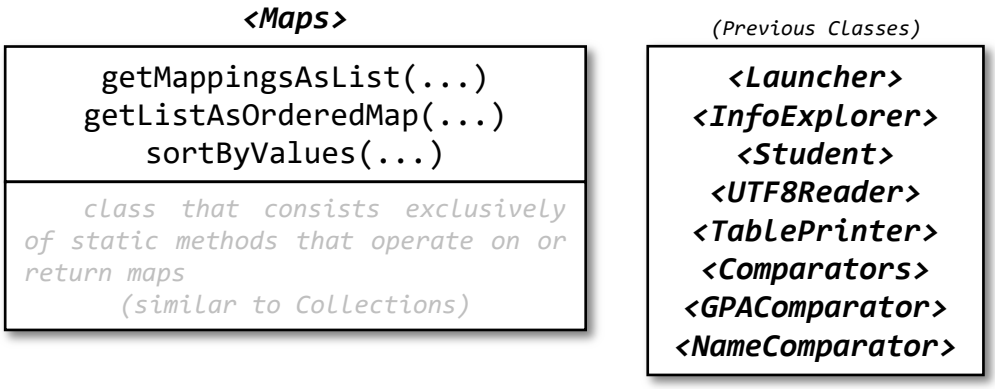
4가지 과정의 시간 복잡도를 모두 합하면, 아래와 같다.

$$O(N_1 + N_2 + N_3 + N_4) \leq O(N + N + NlogN + N)$$
$$= O(3N + NlogN)$$
$$= O\big(N(logN + 3)\big)$$
$$= O(Nlog8N)$$

따라서 4.2.1절에서 설명한 코드의 시간 복잡도는 최대 O(Nlog8N)이다. 리스트 정렬의 시간 복잡도인 O(NlogN)보다 비효율적이지만, 비율의 관점에서 고려해볼 때 큰 차이는 없다.

Section 4.3. Class Structure

ArrayList Implementation Problem과 거의 동일한 구조이다.



Section 4.4. Execution

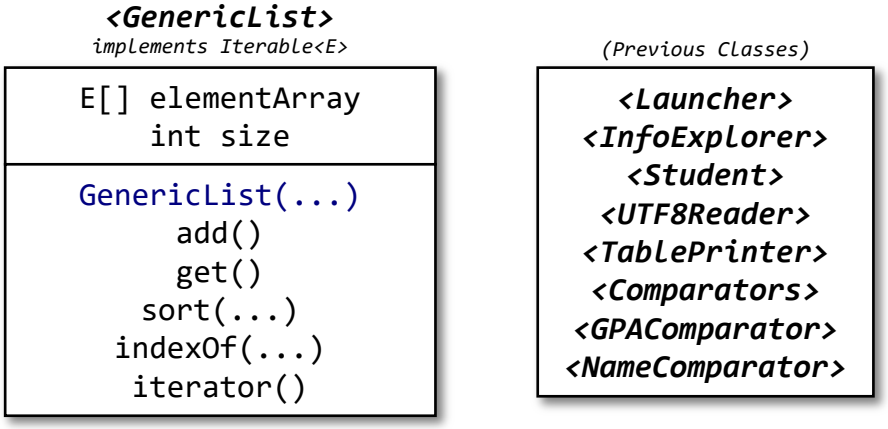
```
Student Info Explorer: HASHMAP_IMPLEMENTATION

Name               | Major                   | Student ID | Grade Point Average
Unnamed Student 13 | Mobile Systems          | 40963389   | 0.7
Unnamed Student 15 | Artificial Intelligence | 40964521   | 0.7
Unnamed Student 5  | Mobile Systems          | 40963723   | 0.81
Unnamed Student 16 | Mobile Systems          | 40968642   | 0.85
Unnamed Student 1  | Web Development         | 40967547   | 1.01
Unnamed Student 11 | Web Development         | 40962691   | 1.12
Unnamed Student 3  | Computer Engineering    | 40962249   | 1.13
Unnamed Student 10 | Computer Engineering    | 40968641   | 1.34
Unnamed Student 7  | Web Development         | 40969372   | 1.65
Unnamed Student 2  | Computer Engineering    | 40965324   | 1.98
Unnamed Student 4  | Mobile Systems          | 40963562   | 2.12
Unnamed Student 8  | Web Development         | 40969012   | 2.52
Unnamed Student 14 | Artificial Intelligence | 40961136   | 3.05
Unnamed Student 9  | Computer Engineering    | 40965118   | 4.03
Unnamed Student 12 | Mobile Systems          | 40963240   | 4.27
Unnamed Student 6  | Artificial Intelligence | 40967425   | 4.49

Student name: Unnamed Student 11
Name                : Unnamed Student 11
Major               : Web Development
Student ID          : 40962691
Grade Point Average : 1.12

Student name: Unnamed Student 500
'Unnamed Student 500' does not exist in the list.

Student name: exit
```

Section 5. GenericList Implementation

ArrayList Implementation Problem에서 ArrayList 대신에 GenericList를 사용하여 구현하는 문제이다. HashMap Implementation Problem과 마찬가지로 구현이 매우 비슷하다.

Section 5.1. Class Structure

ArrayList Implementation Problem과 거의 동일한 구조이다.



GenericList 클래스는 foreach 반복문에 사용할 수 있도록 하기 위하여 Iterable을 구현한다. 따라서 iterator() 메소드를 오버라이딩하는데, iterator()에서는 Iterator<E>를 구현하는 익명 내부 클래스의 인스턴스를 반환한다.

Section 5.2. Execution

```
Student Info Explorer: GENERIC_ARRAY_IMPLEMENTATION

Name               | Major                    | Student ID | Grade Point Average
Unnamed Student 13 | Mobile Systems           | 40963389   | 0.7
Unnamed Student 15 | Artificial Intelligence  | 40964521   | 0.7
Unnamed Student 5  | Mobile Systems           | 40963723   | 0.81
Unnamed Student 16 | Mobile Systems           | 40968642   | 0.85
Unnamed Student 1  | Web Development          | 40967547   | 1.01
Unnamed Student 11 | Web Development          | 40962691   | 1.12
Unnamed Student 3  | Computer Engineering     | 40962249   | 1.13
Unnamed Student 10 | Computer Engineering     | 40968641   | 1.34
Unnamed Student 7  | Web Development          | 40969372   | 1.65
Unnamed Student 2  | Computer Engineering     | 40965324   | 1.98
Unnamed Student 4  | Mobile Systems           | 40963562   | 2.12
```

```
Unnamed Student 8  | Web Development         | 40969012   | 2.52
Unnamed Student 14 | Artificial Intelligence | 40961136   | 3.05
Unnamed Student 9  | Computer Engineering    | 40965118   | 4.03
Unnamed Student 12 | Mobile Systems          | 40963240   | 4.27
Unnamed Student 6  | Artificial Intelligence | 40967425   | 4.49

Student name: Unnamed Student 10
Name                : Unnamed Student 10
Major               : Computer Engineering
Student ID          : 40968641
Grade Point Average : 1.34

Student name: Unnamed Student 6
Name                : Unnamed Student 6
Major               : Artificial Intelligence
Student ID          : 40967425
Grade Point Average : 4.49

Student name: exit
```