제 12 장 제네릭 & 컬렉션(실습)

□ 개념 확인 학습

1. 제네릭을 사용해야 되는 이유는 무엇일까요?

* 제네릭을 이용하면 타입을 매개 변수처럼 간주할 수 있다.
* 자바 컴파일러는 컴파일 시간에 제네릭 코드에 대하여 타입을 검사할 수 있다.
* 제네릭을 이용하면 타입 변환으로 인한 에러를 사전에 방지할 수 있다.
* 제네릭을 사용해 작성된 자바 API 도큐먼트의 이해를 위해서도 필요하다.

2. double형을 저장하는 ArrayList를 생성하는 문장을 작성하세요.

```
ArrayList<Double> list = new ArrayList<Double>();
```

3. 어떤 정보를 저장하는데 저장 순서는 유지 되지 않아도 되지만 절대 중복이 발생해서는 안 될 때, 어떤 컬렉션을 사용해야 할까요?
   ① java.util.Map
   ② java.util.Set
   ③ java.util.List
   ④ java.util.Collection

4. 어떤 정보를 키-값의 쌍으로 저장하고자 한다면, 어떤 컬렉션을 사용해야 할까요?
   ① java.util.Map
   ② java.util.Set
   ③ java.util.List
   ④ java.util.Collection

5. ArrayList<Double> alist = new ArrayList<Double>();로 생성된 후, 객체들이 저장된 alist가 있다고 가정합니다. 이 alist의 모든 원소를 출력하는 문장을 아래에서 제시한 조건을 사용하여 작성하세요.

(1) 인덱스를 사용하는 보통의 for 루프

```
for(int i=0;i<alist.size();i++)
    System.out.println(alist.get(i));
```

(2) for-each 구문을 사용

```
for(Double data : alist)
        System.out.println(data);
```

(3) Iterator를 사용

```
Iterator e = alist.iterator();
while(e.hasNext()) {
   Double d = (Double)e.next();  // 반복자는 Object 타입을 반환!
   System.out.println(d);
}
```

(4) alist 객체를 바로 출력

```
System.out.println(alist); //출력 형태 [ 저장된 값, 저장된 값, ....]
```

□ 적용 확인 학습 & 응용 프로그래밍

6. 다음은 Stack 클래스의 일부분입니다.

(1) Stack에 저장되는 데이터의 타입을 int 대신에 제네릭 타입으로 표시하세요.

```
public class Stack {
   private int[] stack;
   public void push(int data) { ....}
   public int pop() { ....}
}
```

```
public class Stack <T> {
    private T[] stack;
    public void push(T data) { ... }
    public T pop() { ... }
}
```

(2) String 타입의 데이터를 가지는 Stack을 생성하는 문장을 작성하세요.

```
Stack<String> s = new Stack<String>();
```

7. 다음과 같은 코드가 컴파일 되지 않는다면 원인은 무엇일까요?

```
public class MyMax <T> {
    public T max(T x, T y) {
        return x > y ? x : y;
    }
}
```

8. 아래의 프로그램의 main() 메소드와 실행결과를 참고하여 swap()과 arrprint()를 제네릭 메소드로 완성하세요.

swap() : 전달 받은 배열에서 전달 받은 두 정수를 인덱스로 하는 원소를 서로 교환.
arrprint() : 전달 받은 배열의 내용을 출력.

```
import java.util.*;

public class Test {
    //swap() 메소드

    //arrprint() 메소드

    public static void main(String[] args) {
        //Double
        Double drr[] = {1.1, 2.2, 3.3};
        Test.<Double>arrprint(drr);

        Test.<Double>swap(drr, 1, 2);
        Test.<Double>arrprint(drr);

        //String
        String srr[] = {"cat", "dog", "ox"};
        Test.<String>arrprint(srr);

        Test.<String>swap(srr, 0, 1);
        Test.<String>arrprint(srr);
    }
}
```

```
1.1, 2.2, 3.3,
1.1, 3.3, 2.2,
cat, dog, ox,
dog, cat, ox,
```

```
public static <T> void swap(T[] arr, int i, int j) { //a[i]와 a[j]를 서로 교환
    T temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
public static <T> void arrprint(T[] arr) {
    for(T a : arr)
        System.out.print(a + ", ");
    System.out.println();
}
```

9. 아래의 프로그램은 MyAvg 클래스에서 제공하고 있는 기능을 사용하는 main() 메소드와 실행 예 입니다. MyAvg 클래스의 타입 매개변수 T는 Number 클래스의 하위 클래스만 사용 가능하도록 지정합니다. 제네릭을 사용해 MyAvg 클래스를 작성해 보세요. MyAvg 에서 구현해야 하는 메소드 다음과 같습니다.

   getAverage() : T 타입의 배열 요소들의 평균을 반환.

```
public class Test {
   public static void main(String[] args) {

        MyAvg<Integer> mai = new MyAvg<Integer>();

        Integer[] iarr = { 1, 2, 3, 4, 5, 6 };
        System.out.println("mai.getAverage() = " + mai.getAverage(iarr));

        MyAvg<Double> mad = new MyAvg<Double>();

        Double[] darr = { 1.5, 2.5, 3.5};
        System.out.println("mad.getAverage() = " + mad.getAverage(darr));
   }
}
```

```
<terminated> Test (6) [Java Application] C
mai.getAverage() = 3.5
mad.getAverage() = 2.5
```

```
public class MyAvg <T extends Number> {

   double v=0.0;

   public double getAverage(T[] a){
        for (int i = 0; i < a.length; i++)
             v = v + a[i].doubleValue();
        return v/a.length;
   }
}
```

10. 제네릭을 사용하여 문자열(String)을 저장할 수 있는 ArrayList 객체 alist를 생성한 후, alist에 “apple”, “banana”, “candy”를 저장하고 저장된 내용을 출력하는 프로그램을 작성하세요.

11. 다음과 같이 리스트가 생성되었다고 할 때, 다음의 각 문장을 실행한 후의 결과를 쓰세요.

```
String[] s = { "사과", "배", "바나나" };
ArrayList list = new ArrayList(Arrays.asList(s));
```

(1) list.add("포도"); System.out.println(list);

(2) list.add(2, "자몽"); System.out.println(list);

(3) System.out.println(list.get(3));

(4) list.remove(1); System.out.println(list);

(5) System.out.println(list.contains("사과"));

(6) System.out.println(list.indexOf("사과"));

12. 아래의 프로그램은 RandList 클래스에서 제공하는 메소드를 사용하는 main() 메소드와 실행 예 입니다. 모든 타입의 객체가 저장되고 사용될 수 있도록 제네릭을 사용해 RandList 클래스를 작성해 보세요. RandList 클래스에서 구현해야 하는 메소드는 다음과 같습니다.

add() : LinkedList 타입의 list에 원소를 추가.
getlist() : 저장된 전체 list를 반환.
select() : 난수를 이용해 list의 원소 중에서 하나를 랜덤하게 선택하여 반환.

```
import java.util.*;

public class Test {
  public static void main(String[] args) {

    RandList<String> rlist = new RandList<String>();

    String[] sample = {"I", "love", "the", "coffee."};
    for(int i=0; i<sample.length; i++)
       rlist.add(sample[i]);

    System.out.println("==== 전체 데이터 ====");
    List<String> alldata = rlist.getlist();
    for(String s : alldata)
       System.out.println(s);

    System.out.println("==== 랜덤하게 선택된 데이터 ====");
    for(int i=0; i<5; i++)
    System.out.println(i + ")" + rlist.select());
  }
}
```

```
<terminated> Test (6) [Java Application] C:₩Pro
==== 전체 데이터 ====
I
love
the
coffee.
==== 랜덤하게 선택된 데이터 ====
0)love
1)I
2)love
3)coffee.
4)the
```

13. Student 클래스는 학생의 이름(private String name), 전화번호(private String phone)를 필드로 가집니다. 적절한 생성자, 접근자(getter) 메소드, toString() 메소드를 작성하세요.

```
class Student {
    private String name;
    private String phone;

    public Student() {
        this("None", "None");
    }
    public Student(String name, String phone) {
        this.name = name;
        this.phone = phone;
    }
    public String getName() {
        return name;
    }
    public String getPhone() {
        return phone;
    }
    @Override
    public String toString() {
        return "Student [name=" + name + ", phone=" + phone + "]";
    }
}
```

14. 아래의 실행화면을 참고하여 13번에서 작성한 학생(Student 객체)들의 정보를 ArrayList에 저장하고 검색할 수 있는 Test 클래스를 작성하세요.

```
public class Test {
    public static void main(String[] args) {

        //학생 정보를 저장하는 alist 생성


        //실행화면과 같이 alist에 학생 정보를 저장


        //alist의 모든 요소를 출력


        //사용자에게 이름을 입력 받아 전화번호 검색 후 해당 내용 출력


        //사용자에게 이름을 입력 받아 데이터 삭제 후 alist의 모든 요소를 출력
```

```
<terminated> Test (6) [Java Application] C:₩Program Files₩Java₩
Student [name=name0, phone=phone0]
Student [name=name1, phone=phone1]
Student [name=name2, phone=phone2]
Student [name=name3, phone=phone3]
Student [name=name4, phone=phone4]

이름을 입력하시면 전화번호를 드립니다 : name3
name3의 전화번호 = phone3

이름을 입력하시면 정보를 삭제합니다 : name1
Student [name=name0, phone=phone0]
Student [name=name2, phone=phone2]
Student [name=name3, phone=phone3]
Student [name=name4, phone=phone4]
```

```
    }
}
```

15. 아래의 실행화면을 참고하여 Map에, 생성 순서 번호를 key로, 13번에서 작성한 학생(Student) 객체를 value로 저장하고 출력하는 Test 클래스를 작성하세요.

```
<terminated> Test (6) [Java Application] C:₩Program Files₩Java₩jre1.8.0_261₩bin₩java
key=0, value=Student [name=name0, phone=phone0]
key=1, value=Student [name=name1, phone=phone1]
key=2, value=Student [name=name2, phone=phone2]
key=3, value=Student [name=name3, phone=phone3]
key=4, value=Student [name=name4, phone=phone4]
```