
JDK8로 배우는 자바 프로그래밍
어서와
Java는
처음이지!
천인국 지음
False
True
INFINITY BOOKS

TEST BANK

CHAPTER 15

1. 다음의 질문에 간단히 답하시오.

(1) double형을 저장하는 ArrayList를 생성하는 문장을 작성하시오.

(2) Iterator 인터페이스는 어떤 목적으로 사용되는가?

(3) 리스트(list)와 집합(set)의 차이점은 무엇인가?

(4) 키와 값의 매핑을 나타내는 인터페이스는 무엇인가?

2. 여러분이 어떤 정보를 저장하는데 절대 중복이 발생하면 안된다고 가정하자. 그리고 모든 요소들은 삽입된 순서대로 출력되어야 한다. 어떤 컬렉션을 사용하여야 하는가?

①java.util.Map ②java.util.Set ③java.util.List ④java.util.Collection

3. 어떤 정보를 키-값의 쌍으로 저장하고자 한다. 어떤 컬렉션을 사용하여야 하는가?

①java.util.Map ②java.util.Set ③java.util.List ④java.util.Collection

4. 다음 프로그램의 출력은?

```
import java.util.Iterator;
import java.util.TreeSet;

public class Test
{
    int s;
    public static void main(String [] args)
    {
        TreeSet map = new TreeSet();
        map.add("one");
        map.add("two");
        map.add("three");
        map.add("four");
        map.add("one");
        Iterator it = map.iterator();
        while (it.hasNext() )
        {
            System.out.print( it.next() + " " );
        }
   }
}
```

5. 실제로 컴파일러는 컴파일 시에 모든 타입 매개 변수들을 삭제한다. 그럼에도 불구하고 우리가 제네릭을 사용해야 되는 이유는 무엇일까?

6. 다음과 같은 코드가 컴파일되는가? 만약 컴파일되지 않는다면 원인은 무엇인가? 실제로 컴파일해보아도 좋다.

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

(힌트) > 연산자는 기초 자료형에만 적용된다.

7. 배열에서 원소 2개를 서로 교환하는 제네릭 메소드를 작성하시오.

```
public final class MyAlgorithm {
    public static <T> void swap(T[] a, int i, int j) {
        ____________________;
        ____________________;
        ____________________;
    }
}
```

8. 다음은 Stack 클래스의 일부분이다.

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

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

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

9. 다음과 같이 리스트가 생성되었다고 하자. 다음의 각 문장을 실행한 후의 결과를 쓰시오.

```
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"));

10. list가 ArrayList<Double>의 객체를 참조하고 있다고 하자. list의 모든 원소를 출력하는 문장을 다음과 같이 작성하라.

(1) 인덱스 변수를 사용하는 보통의 for 루프
(2) for-each 구문을 사용
(3) Iterator를 사용

11. 본문에서 ArrayList를 설명하였다. 문자열을 저장할 수 있는 ArrayList 객체를 생성하고 여기에 "a", "b", "c", "d", "e"를 저장하고 이것을 출력하는 프로그램을 작성해보자.

```
[a, b, c, d, e]
```

12. 랜덤 리스트(random list)를 작성하여 보자. 랜덤 리스트란 원소들을 가지고 있다가 get()이라는 메소드를 호출하면 랜덤하게 하나의 원소를 선택하여서 반환한다. 모든 타입의 객체를 저장하도록 제네릭을 사용하라. 원소들은 ArrayList를 이용하여서 저장하고 add() 메소드는 원소를 추가한다. select()가 호출되면 난수 생성기를 이용하서 ArrayList 원소 중에서 하나를 선택하여 반환하라. 다음 코드를 참조하라.

```
public class RandomList<T> {
   _____________________________________;
   public void add(T  item) { _____________________;}
   public T select() { ___________________________;}
}
```

13. 타입 매개 변수 T를 가지는 클래스 MyMath를 작성하여 보자. MyMath에는 평균을 구하는 getAverage() 메소드를 추가하여 보자. Integer나 Double과 같은 다양한 타입의 데이터에 대하여 평균을 구할 수 있도록 하라.

14. 제네릭을 사용하여서 똑같은 타입의 객체 두개를 저장하는 Pair 클래스를 작성하여 보자. 생성자와 접근자, 설정자, toString() 메소드를 정의하라. String을 저장하여 다음과 같이 테스트하여 보라.

```
MyPair<String> fruit = new MyPair<String>("사과", "포도");
```

15. 클래스 안에서 하나의 메소드만 제네릭으로 만들어보자. 제네릭 메소드 a()를 가지는 클래스 Test를 정의하여 보자. 메소드 a()는 매개 변수의 클래스 이름을 출력한다. 객체 obj의 클래스 이름은 obj.getClass().getName()으로 출력할 수 있다. int나 float와 같은 기초형 값을 전달하여서 호출해보자. 어떤 값이 출력되는가?

16. 장기 자랑 프로그램에 사용될 수 있는 심사 위원들의 점수를 집계하는 프로그램을 작성하라. 점수는 0.0에서 10.0까지 가능하다. 10명의 점수 중에서 최저 점수와 최고 점수는 제외된다. Double 타입의 ArrayList를 사용하라.

실행결과

```
심사위원의 점수: 1
심사위원의 점수: 2
심사위원의 점수: 3
심사위원의 점수: 4
심사위원의 점수: 5
심사위원의 점수: 6
심사위원의 점수: 7
심사위원의 점수: 8
심사위원의 점수: 9
심사위원의 점수: 10
점수의 합: 44.0
```

17. 학생들의 정보를 ArrayList에 저장하고 검색할 수 있는 프로그램을 작성하라. 학생들의 정보는 Student라는 클래스로 나타낸다. Student는 학생의 이름, 주소, 전화번호 등의 필드로 가진다. 적절한 접근자와 설정자를 작성하라. 학생들의 정보를 추가하고 검색하고 삭제하는 간단한 메뉴를 제공한다. ArrayList의 원소들을 처리할 때 for-each 루프를 사용하라.

18. 로또 번호를 생성하는 프로그램을 작성하여 보자. 로또는 1부터 45까지의 숫자 중에서 6개를 선택한다. 로또 번호는 중복되면 안 된다. 따라서 집합을 나타내는 HashSet을 사용하여서 중복을 검사하여 보자. Math.random()을 사용하면 0부터 1사이의 난수를 생성할 수 있다. 0부터 1사이의 난수가 생성되면 여기에 44를 곱하고 1을 더하면 1부터 45사이의 정수를 생성할 수 있다. 생성된 정수는 HashSet의 contains() 메소드를 이용하여서 이미 선택된 정수인지를 검사한다.

실행결과

```
Lotto [set=[16, 23, 7, 8, 26, 30]]
```