수행 테스트 (제네릭,컬렉션,람다식)
4/5

이름 :　　닉네임 :

1. 제네릭클래스와 제네릭인터페이스는 내부에 여러 타입의 객체를 멤버필드에 가질 수 있습니다. 비슷한 역할을 할 수 있는 Object자료형으로 멤버필드를 정의하는 방법과 비교해서 어떤 장점이 있는지 적으세요. (2점)

1. 명시성 : 제네릭을 사용하면 코드의 가독성이 높아짐. 제네릭을 사용하면 코드가 더 명확하게 표현되며, 어떤 타입의 객체가 사용되는지 미리 알 수 있음. 또한, 코드 중복을 줄일 수 있어 유지 보수가 용이.
2. 타입 변환의 필요성 감소: Object 자료형을 사용하면 객체를 다룰 때 항상 타입 변환을 해야 함. 하지만 제네릭을 사용하면 컴파일러가 타입 체크를 수행하기 때문에 타입 변환을 할 필요가 없어짐.
3. 성능 향상: 제네릭을 사용하면 컴파일러가 타입 체크를 수행하므로 불필요한 타입 변환을 하지 않아도 됨. 이로 인해 런타임 오버헤드가 줄어들어 성능이 향상될 수 있음.

2. 아래는 제네릭 타입 변수 2개를 가진 제네릭 클래스의 선언식입니다. 빈칸 네모에 적당한 코드를 넣어 식을 완성하세요.

```
Class KeyValue < K, V >  {
        private K key;
        private V value;
}
```

3. 아래 코드는 제네릭 타입 제한 범위를 설정하여 특정한 클래스만 오도록 한정할 수 있고 그 사용하는 클래스의 메서드를 자유롭게 사용할 수 있습니다. 빈칸 네모에 적당한 코드를 넣어 제네릭 클래스 D는 A 클래스는 받을 수 없고 오직 B와 C클래스만 사용할 수 있도록 선언하세요.

```
Class A {}
Class B extends A {}
Class C extends B {}
Class D < T extends B > {  // B, C만 올 수 있음
        private T t;
}
```

4. 컬렉션(Collection)의 특성에 따라 구분하면 크게 List, Set, Map 으로 구분할 수 있습니다. 이중 Set가 List와 비교해서 어떤 **다른** 특성을 가지는지 아는대로 설명하세요.

1. 중복 요소 허용 여부 : List는 중복 허용하는 반면, Set은 중복된 요소를 허용하지 않음.
2. 요소의 순서 : List는 참조형 자료구조로서 요소들의 순서를 보존함. Set는 요소의 순서를 보장하지 않음.(집합형 자료구조)

5. ~ 7. 다음은 컬렉션(Collection)의 구현체에 대한 설명입니다. 아래 설명에 맞는 **구현체 클래스**의 이름을 적으세요.

5. "각 노드가 데이터를 저장하는 요소(element)와 다음 노드 및 이전 노드에 대한 참조(주소)를 가지고 있는 구조임. 각각의 노드는 연결 리스트의 시작부터 끝까지 순차적으로 데이터를 연결. 이는 데이터의 삽입과 삭제가 빈번한 경우에 유용하며, 데이터의 삽입 및 삭제 연산이 빠른 특징을 가짐."

LinkedList

6. “집합(Set)의 특성을 가지고 있어 중복된 요소를 허용하지 않음. 또한 요소들은 오름차순으로 정렬되어 저장함. 내부적으로 이진 검색 트리를 사용하여 요소들을 저장하고 관리함. 이진 검색 트리의 특성을 이용하여 요소의 삽입, 삭제, 검색 등의 연산을 수행하는데 장점을 가지고 있음.” TreeSet

7. “후입선출(LIFO, Last-In-First-Out)의 원칙을 따르는 자료 구조임. 데이터를 넣을 때는 맨 위에(push), 데이터를 빼낼 때도 맨 위에서(pop) 빼내는 형태로 동작함. 즉, 가장 최근에 추가된 데이터가 가장 먼저 제거되는 구조임. 이러한 특성 때문에 주로 재귀 알고리즘, 브라우저의 뒤로 가기 버튼 등에 사용됨.” Stack

8. 람다식을 사용하면 코드를 간결하게 표현할 수 있고, 익명 함수(anonymous function)를 만들어서 메서드의 인수로 전달하거나 반환할 수 있습니다. 이때 사용되는 인터페이스로서 **<u>딱 하나의 추상메서드만을 가지는 인터페이스</u>**를 무엇이라고 하는지 적으세요.

함수형 인터페이스

9. 아래 코드에서 위쪽은 익명이너클래스의 선언이고 아래쪽은 람다식 표현방식의 코드입니다. 동일한 동작이 되도록 람다식 표현코드를 완성하세요.

```
D d1 = new D() {
        @Override
        public double method(int a, double b) {
                return a + b;
        }
}
```

람다식 표현

```
D d1 = (a, b) -> a + b;
```

10. 아래 코드는 여러 학생의 이름과 점수를 저장하고, 특정 학생의 점수를 조회하는 예제코드입니다. 빈칸 네모부분에 적절한 코드를 추가하여 완성하세요. (2점)

```
public class StudentScores {
    public static void main(String[] args) {
        // HashMap을 사용하여 studentScores를 선언,생성합니다.
        Map<String, Integer> studentScores = new HashMap<>();
        // 학생의 이름과 점수를 HashMap에 추가합니다.
        studentScores.put("Alice", 85);
        studentScores.put("Bob", 90);
        studentScores.put("Charlie", 78);
        studentScores.put("David", 92);
        studentScores.put("Eve", 88);

        // 특정 학생의 이름을 사용하여 점수를 조회합니다.
        String studentName = "Bob"; // key
        //studentScores에서 value를 읽는 방법
        Integer score = studentScores.get(studentName);

        if (score != null) {
            System.out.println(studentName + "의 점수: " + score);
        }
    }
}
```