Java Mission #6

Collection API

개 요

지난 Mission 까지 우리는 개체의 순서를 유지하면서 관리하기 위한 방법으로 배열(array)를 사용하였다. 본 Mission 에서는 새로운 개체를 추가하거나 삭제할 때 배열보다 효율적인 Collection 클래스를 이용해서 비슷한 기능을 수행하는 프로그램을 작성하게 된다.

학습목표

배열의 장점과 단점을 설명할 수 있다.
ArrayList 를 이해하고 프로그램에 응용할 수 있다.
Iterator 를 이용한 코드를 작성할 수 있다.
객체지향 문법을 적용하여 Singleton 패턴을 적용할 수 있다.

예상 소요시간

2 시간

Coding Process

제공되는 프로젝트(JavaMission6)를 이클립스 환경 내로 import 한다.

1. 코드를 실행하여 결과를 확인하고, 프로그램의 흐름을 분석한다.

2. VehicleManager 클래스 내의 생성자(Constructor) 코드를 수정한다.
   A. Vehicle 개체를 담을 수 있는 ArrayList 개체를 생성한다.
   B. Airplane, Car, Ship 개체들을 생성하면서 생성자를 이용해 값을 초기화하고 ArrayList 에 추가한다.

3. sortByModelName() 메서드를 작성한다.
   A. 전체적인 로직은 이전에 사용한 정렬 로직과 동일하다. 단지 개체 참조값을 얻어내거나 수정하는 방법이 배열(array)와 다를 뿐이다.
   B. ArrayList 의 메서드를 활용해서 개체의 modelName 을 기준으로 오름차순으로 정렬하도록 메서드의 기능을 완성한다.

4. displayVehicles1() 메서드를 작성한다.
   A. 새로운 for 문법을 이용하여 ArrayList 로부터 저장된 개체를 순차적으로 얻어낸다.
   B. 얻어낸 개체가 가지고 있는 displayInfo() 메서드를 호출한다.

5. displayVehicles2() 메서드를 작성한다.
   A. ArrayList 로부터 Iterator 에 대한 참조값을 얻어내서 저장한다.
   B. while 안에서 iterator 를 이용해 순차적으로 저장된 개체에 대한 참조값을 얻어온 다음, 얻어낸 개체가 가지고 있는 displayInfo() 메서드를 호출한다.

6. displayVehicles(String title, int type) 메서드를 완성한다. type 에 따른 동작방식은 아래와 같다.
   (숫자 리터럴 대신 Vehicle 클래스에서 제공되는 상수를 활용할 것)
   type = 0 인 경우: 전체 목록 출력
   type = 1 인 경우: 비행기 목록 출력
   type = 2 인 경우: 자동차 목록 출력

`type = 3` 인 경우: 선박 목록 출력

7. `VehicleManager` 클래스를 `Singleton` 패턴을 적용하도록 수정하여 직접 인스턴스를 만들 수 없도록 처리하고, `main()` 메서드에서 하나의 객체를 얻어와 사용하도록 기존 코드를 수정한다.

8. 작성된 프로그램을 충분히 테스트 한 후, 문제가 없는 경우 export 하여 제출한다.