이번 상속 챕터에서는 부모 클래스와 자식클래스가 뭔지, 그리고 오버라이딩과 파이널에 대해 알아보도록 하겠습니다.

먼저 상속이란 기존 클래스. 즉, 부모클래스를 재사용하여 새로운 자식 클래스를 작성하는 것을 말합니다. 이를 통해서 얻을 수 있는 장점은 부모클래스의 필드와 메소드를 재사용할 수 있고요. 또, 기존 클래스를 변경할 수 있습니다. 우리는 이를 오버라이딩이라고 부릅니다. 이러한 재사용을 통해 코드의 중복을 줄일 수 있는 장점도 있습니다. 특징으로는 상속하려는 대상이 class냐 interface냐에 따라 나뉘는데요. class의 경우 extends를 사용하여 단일 상속만을 받을 수 있고 interface의 경우 implements를 통해 다중으로 상속이 가능합니다.

Class가 다중상속이 불가능한 이유는. 예를 들어, A라는 클래스가 print(); 라는 메소드를 가지고 있는 B와 C클래스를 상속받는다고 가정해보면, A에서 print(); 라는 메소드를 호출하였을 경우 이 print()는 B의 것인지 C의 것인지 알 수가 없기 때문입니다. 이러한 혼란을 방지하기위해 자바는 클래스의 다중상속을 지원하지 않습니다.

상속의 형식으론 자식클래스를 앞에 쓰고 extends 뒤에 부모클래스를 작성합니다. 밑에 예시를 보시면 Super클래스에는 int타입 x와 y 필드가 있고 Sub클래스에는 int타입 x,y,z가 있습니다. 만약 Sub클래스가 Super클래스를 상속시에 부모가 가진 필드나 메소드는 모두 자식의 것으로 다음과 같이 중복을 제거할 수 있습니다.

다음으로 오버라이딩이란 부모클래스로부터 상속받은 메소드의 내용을 자식클래스에서 변경하는 것을 말합니다. 이를 ‘재정의한다’ 라고도 합니다. Super클래스에서 print() 라는 메소드를 물려받은 Sub클래스는 이를 자기 마음대로 다시 작성할 수 있습니다. 오버라이딩을 할 때 필요한 조건이 있는데 우선, 부모클래스의 메소드와 이름이 같아야 합니다. 그리고 매개변수의 타입과 개수, 순서가 같아야하고 리턴타입 역시 같아야 합니다.

끝으로 알아볼 개념은 final 입니다. final이 붙어있는 클래스와 메소드 그리고 멤버필드는 각각상속금지, 오버라이딩 금지, 상수값 변경 금지의 특징을 가지고 있습니다. 코드로 표현할 땐 맨 우측처럼 표현합니다. 대표적인 예로 리턴타입이 final인 String 클래스는 상속이 불가능합니다.

자, 컬렉션 프레임워크에 대해 알아보기 전에 컬렉션과 프레임워크의 개념에 대해 알아보도록 하겠습니다. 첫번째로 컬렉션이란. 여러 데이터를 모아 놓은 것을 말합니다. 뭐 예를 들어서 가방 컬렉션, 가전제품 컬렉션 이런 것들처럼 컴퓨터 상에서는 여러 데이터를 모아 놓은 것. 이를 컬렉션이라고 합니다. 그 다음 프레임워크란 표준화된 프로그래밍 방식을 얘기하는데요. 예를 들어 통계프레임워크라고 하면 통계에 관련된 계산 방법들이 모여서 표준화 돼있는 것을 말하겠지요. 그래서 컬렉션 프레임워크란 컬렉션을 편리하게 다루기 위해 다양한 클래스를 표준화된 방식으로 모아둔 것입니다. 그리고 이 컬렉션 프레임워크의 종류에는 크게 3가지가 있습니다. 바로 List와 Set 그리고 Map이 있습니다.

계층도를 보시면 List와 Set은 Collection을 상속받고 있고 Map의 경우 상속받고 있지 않습니다. 그리고 List 아래에는 ArrayList와 Vector가 상속받고 Set 아래에는 HashSet, Map 아래에는 HashMap이 상속받고 있습니다.

list와 set 그리고 map에서 중요한 점은 순서와 중복이 있느냐 없느냐 인데요. 우선 전체적인 틀을 한번 보시고 하나씩 자세히 알아보도록 하겠습니다.

List로 구현된 클래스인 ArrayList는 순서와 중복을 허용합니다. 코드를 보시면 String타입의 ArrayList를 선언 후 이름을 하나씩 집어놓고 있는데요. 오른쪽에 있는 그림과 같이 내가 넣은 이름이 순서대로 들어가는 모습을 볼 수 있습니다.

List로 구현된 클래스인 Vector 역시 순서와 중복을 허용합니다. ArrayList와 다른점으론 초기값과 용량의 증가량을 설정해 줄 수 있다는 점입니다. 코드를 보시면 (10, 5)가 적혀있는데 이는 총 10칸짜리 공간에 이를 초과할 시 5칸씩 증가한다는 것을 의미합니다. 괄호 안에 아무값도 넣어주지 않으면 Vector의 초기값은 기본적으로 10칸입니다.

Set으로 구현된 클래스인 HashSet은 순서와 중복이 없습니다. 코드를 보시면 String 타입의 HashSet을 선언 후 1,1,3,2,5,4를 넣어주었습니다. 그리고 나서 이를 출력하면 입력한 순서에 상관없이 [1,2,3,4,5]가 출력되는 모습을 볼 수 있습니다. 우측에 있는 그림처럼 Set은 넣는 순서와 중복이 상관없다는 것을 알 수 있습니다.

Map으로 구현된 클래스인 HashMap은 순서는 없지만 중복의 경우 Key는 허용하지 않고 값의 중복을 허용하고 있습니다. 그리고 Map의 특징은 key와 value를 하나로 묶어서 관리된다는 점입니다. 코드를 통해 예를 들면, Integer 타입의 key값과 String타입의 Value를 갖는 HashMap 객체에 다음과 같이 데이터를 넣고 출력하면 이순신이란 값이 두개가 있는 것을 확인할 수 있지만 장보고의 데이터는 찾을 수 없습니다. 이처럼 Map은 key값의 중복은 허용하지 않지만 value는 중복을 허용하는 것을 알 수 있습니다.

Map에서 key와 value를 따로 관리하는 방법으로 enteySet이 있습니다. 기존에 선언했던 HashMap객체를 entrySet 메소드를 통해 key와 value를 따로 묶어서 원하는 key값 또는 value를 단독으로 호출할 수 있습니다.

컬렉션에 있는 데이터를 표준화하여 하나씩 읽어오는 Iterator는 값이 있는지 확인하여 참, 거짓으로 반환하는 hasNext 메소드와 해당값을 리턴해주는 next 메소드를 이용하여 컬렉션에 담긴 내용을 읽어올 수 있습니다. 코드를 보시면 entrySet을 통해 생성된 mySet을 iterator에 담아줍니다. 그리고나서 hasNext를 통해 ir인스턴스 안에 데이터가 있는지 확인하여 ir.next 메소드를 통해 하나씩 출력해줍니다.

Comparable과 Comparator는 객체를 정렬할 때 사용하는 인터페이스입니다. 이 둘의 차이점은 comparable은 나와 다른 객체를 비교하는 것이고 comparator는 다른 객체가 두 객체를 비교한다는 점입니다.

우선 comparable에 대해 알아보면 comparable을 상속받는 Score 클래스를 만들어줍니다. 그리고 compareTo 메소드를 override 하여 국어점수를 비교하는데, 내 국어점수가 다른 국어점수보다 크면 1, 같으면 0, 작으면 -1을 리턴해줍니다. 그리고 메인메소드에서 정렬시켜주고 출력하면 국어점수를 오름차순으로 정렬시켜 줍니다.

comparator를 사용하기 위해선 comparator를 상속받는 객체를 생성해줘야 합니다. MySortTest라는 클래스를 생성 후 Comparator를 상속받아 내가 정렬하고 싶은 대로 리턴값을 compare메소드에서 오버라이드 해줍니다. 그리고나서 메인에서 정렬할 때 sort메소드 안에 정렬할 객체와 기준이 될 객체를 입력해줍니다. 이렇게 되면 오름차순으로 출력되던 list가 내림차순으로 출력되는 모습을 볼 수 있습니다.