컬렉션

목표:

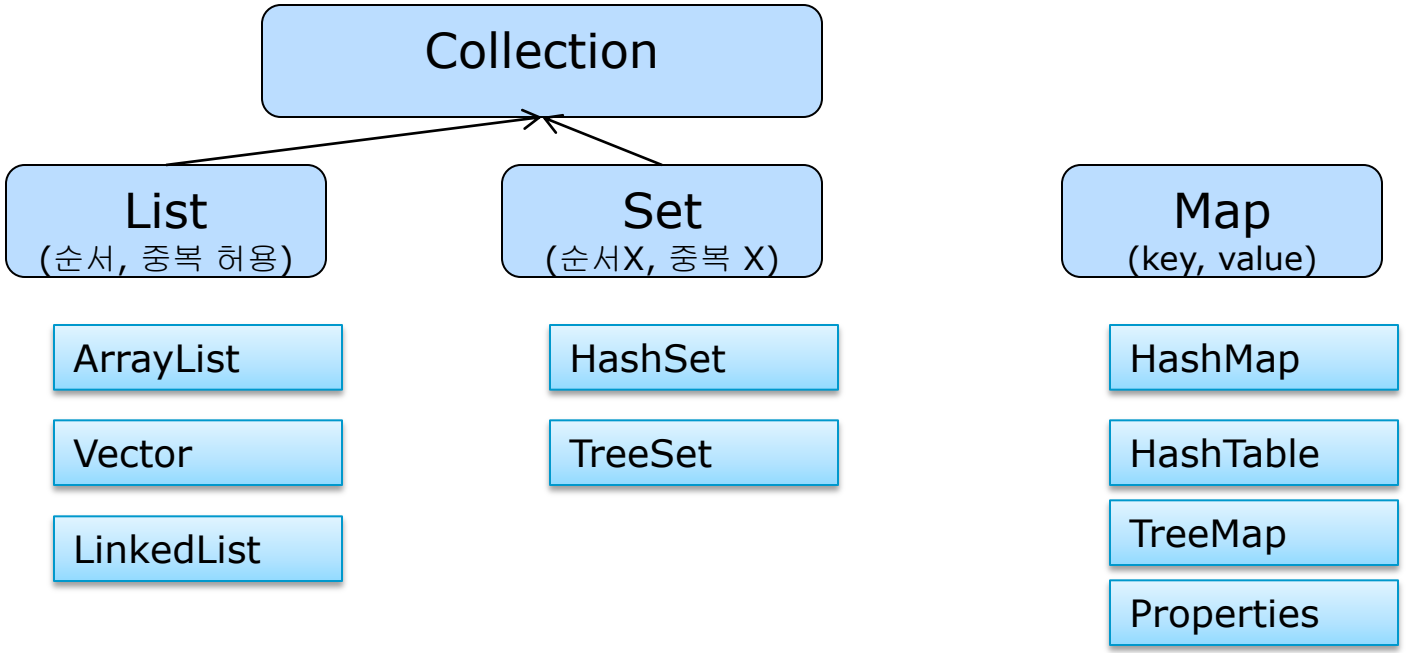
- 자바에서 메모리 저장에 활용되는 컬렉션 프레임워크에 대하여 이해하고 사용할 수 있다.
- 저장 시, 활용되는 **generic type**에 대하여 알고 파악한다.
- 객체 단위로 사용되는 **ArrayList<**객체**>**에 대하여 활용할 수 있다.
- 기타 컬렉션의 상속관계와 공통 메서드 및 개별 메서드의 기능에 대해 활용할 수 있다.

생각해봅시다:

- 배열 객체의 한계
  - String[] names={“홍길동”,”김길동”};
  - 배열 데이터의 크기가 변하는 내용인 동적 배열은 자바의 기본 배열type에서는 지원하지 않음.
- 데이터를 저장할 때, 어떤 자료구조의 형태가 필요로 하는가?
  - 순번, 중복, key-value, index, 계층구조
  - 필요에 따른 다른 자료구조

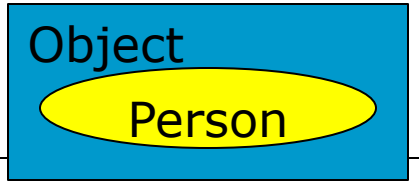
컬렉션이란?

- 데이터의 저장을 객체를 이용하여 효율적으로 추가, 삭제, 검색
- 자바 컬렉션은 객체를 수집해서 저장하는 역할을 하고 있다.



제네릭(Generic)

- 컬렉션 구조의 특정 클래스 type에 대한 선언을 하지 않으면 잘못된 타입이 사용될 수 있는 문제를 컴파일 과정에서 나올 수 있기에 java5부터 지원하고 있다.
- 컴파일 시 강한 type checking을 할 수 있다.
- 컬렉션의 활용
  - ArrayList li = new ArrayList();
    - 모든 클래스의 최상위 클래스인 Object를 넣을 수 있는 list형태의 동적 배열인 ArrayList를 객체생성
    - li.add("홍길동"); Object o = "홍길동";
    - li.add(new Person()); Object o = new Person();
    - (Person)li.get(1);
      - 사용할려면 typecasting이 필요하다.



- ArrayList(컬렉션)에는 여러 객체Type이 들어갈 수 있다?
  - ➔ 문제 발생
    - 어떤 객체 Type 들어갈 수 있다.
    - 해당 내용을 가져올 때, 들어간 객체Type으로 casting 해서 사용해야 한다.
- 하나의 객체Type만 들어오게끔 처리 generic
  - 컬렉션클래스**<지정한객체type> li = new 컬렉션클래스 <지정한객체type>();**
    - 다른 TYPE을 입력시, 에러➔ 지정한 TYPE만 추가가능
    - 가져올 때, TYPE CASTING을 할 필요가 없다.

컬렉션 객체의 공통 메서드

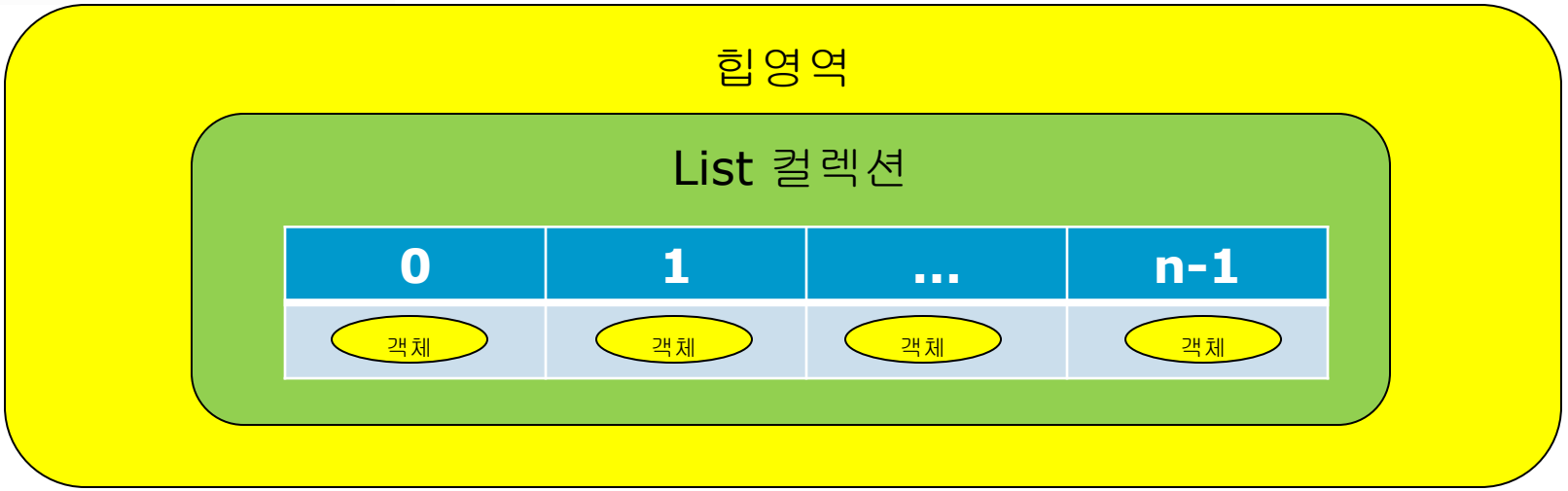
- add(추가할객체) : 기본적인 추가 처리
- get(index) : 해당 index위치에 있는 객체 가져오기
- ArrayList<String> fruits = new ArrayList<String>();
- fruits.add("사과");
- fruits.add("바나나");
- fruits.get(0); ➔ "사과"

확인예제

- Food 클래스
  - 음식명, 가격 (필드, 메서드 선언)
- ArrayList<제너릭>
  - add(), get() 활용하여 데이터 3건을 입력하고.
  - 출력형식
    - NO　음식명　가격
    - 1　@@@　@@@
    - 2　@@@　@@@
    - 3　@@@　@@@

List 컬렉션

- 객체를 일렬로 늘어놓은 구조



- 기능 메서드
  - add(객체) : 객체를 맨 끝에 추가
  - add(index, 객체) : 주어진 인덱스에 객체를 추가
  - set(index, 객체) : 주어진 인덱스에 객체로 바꿈

List 컬렉션

- 기능 메서드
  - get(index) : 주어진 인덱스에 저장된 객체 가져옴.
  - size() : 저장되어 있는 전체 객체 수를 리턴
  - contains(객체): 주어진 객체가 저장되어 있는지 여부(boolean)
  - isEmpty() : 컬렉션이 비어 있는지 여부(boolean)
  - clear() : 저장된 모든 객체를 삭제
  - remove(index) : 주어진 인덱스에 저장된 객체 삭제
  - remove(객체) : 주어진 객체를 삭제

ArrayList 확인예제(숙제)

- 과일가게 ArrayList sellList
- 장보는사람 ArrayList buyList
- 시나리오
  - 과일가게에 사과, 바나나, 딸기, 오렌지, 수박을 팔고 있습니다. add
  - 장보는 사람이 사과와 딸기를 구매했습니다.
    - remove, add
  - 과일가게에서 바나나를 사과로 변경했습니다.
  - 장보는 사람이 사과와 딸기 다 먹었습니다.
  - 장보는 사람이 가지고 있는 과일이 없다면 과일가게에서 남은 과일 중 하나를 장보는 사람에게 사은품으로 주기로 했습니다.

객체와 ArrayList

- 단위의 객체를 List형태로 사용하는 것은 일반 활용도가 높은 데이터 구조 형태이다.
- 아래와 같은 테이블 구조의 데이터를 ArrayList형태의 컬렉션 구조에 담는 것을 해보도록 하자.

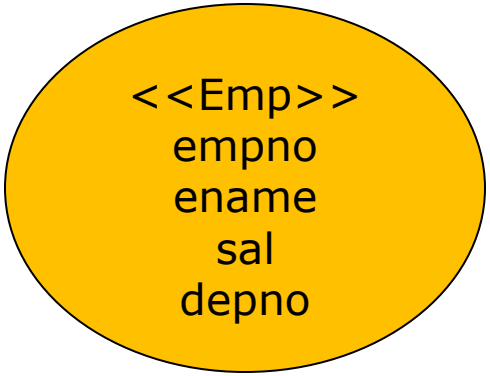
empno	ename	sal	deptno
7001	하이맨	**2000**	**10**
7002	홍길동	3000	20
7003	슈퍼맨	4000	30

- 단위 객체를 만들기 위해 한 row의 데이터를 담을 수 있는 클래스 설계부터 한다.

객체와 ArrayList

empno	ename	sal	deptno
7001	하이맨	**2000**	**10**
7002	홍길동	3000	20
7003	슈퍼맨	4000	30

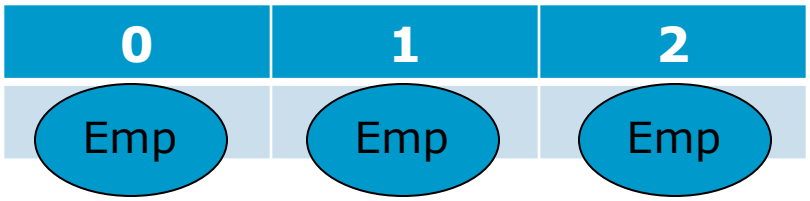
- 단위 클래스 설계
  - class Emp{
  - private int empno;
  - private String ename;
  - private int sal;
  - private int deptno;
  - }



객체와 ArrayList

empno	ename	sal	deptno
7001	하이맨	**2000**	**10**
7002	홍길동	3000	20
7003	슈퍼맨	4000	30

- 단위 객체 생성 및 ArrayList에 담기
  - ArrayList<Emp> elist = new ArrayList<Emp>();
  - Emp p = new Emp(7001, "하이맨", 2000, 10);
  - elist.add(p);
  - elist.add(new Emp(7002, "홍길동", 3000, 20));
  - elist.add(new Emp(7003, "슈퍼맨", 4000, 30));



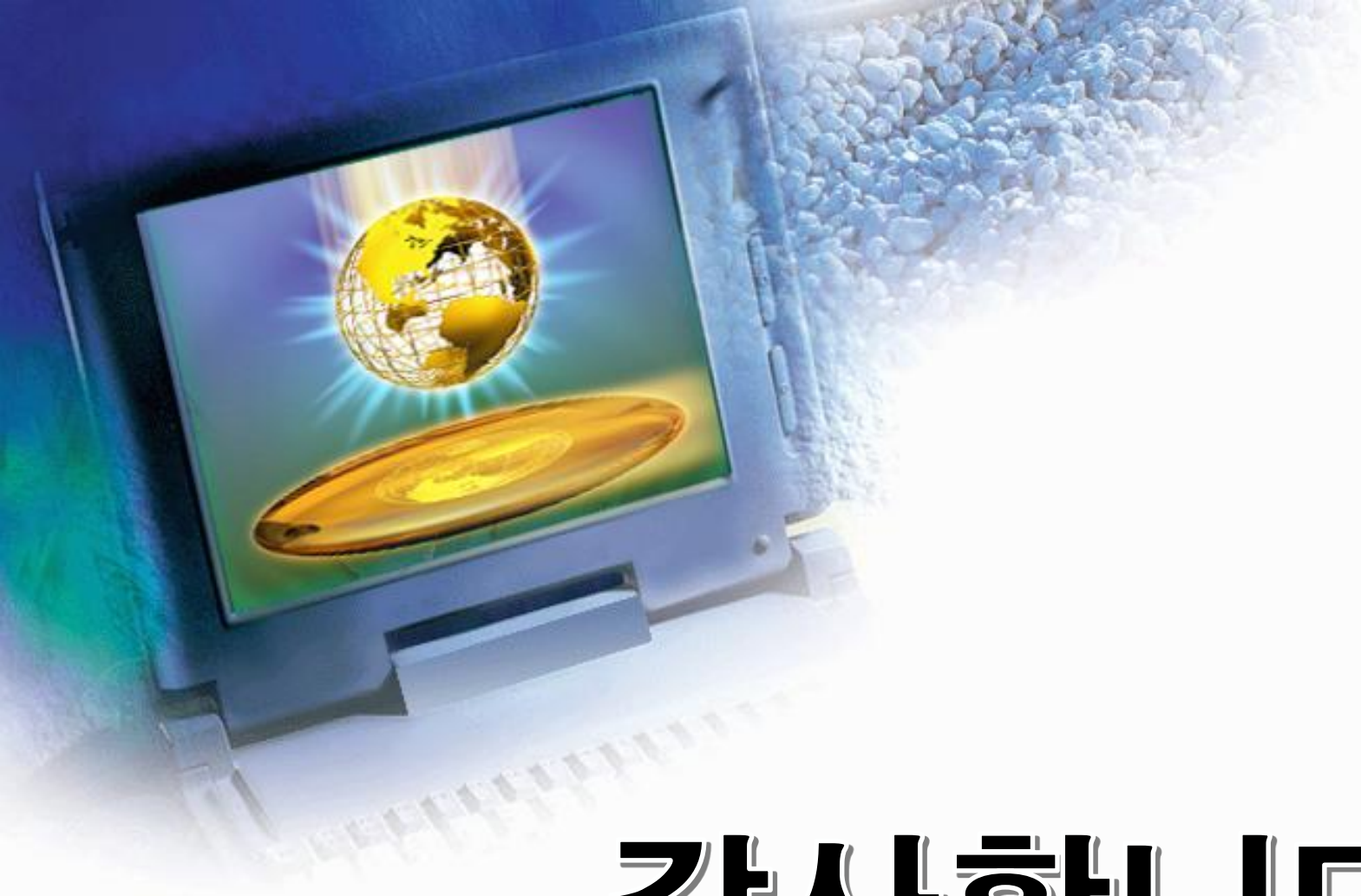

```
elist.get(index).getEmpno()
elist.get(index).getEname()
elist.get(index).getSal()
elist.get(index).getDeptno()
```

확인 예제:

- 단위 클래스 설계 class
- ArrayList<클래스>
- list 할당..
- 할당된 데이터 가져오기

deptno	dname	loc
10	회계	뉴욕
20	감사	달라스
30	영업	시카고

정리 및 확인하기



감사합니다 !