컬렉션 프레임워크 - 데이터 집합을 효율적으로 관리하기 위해 표준화 시킨 것

list

- 컬렉션을 상속받아 하나의 데이터를 순서대로 저장하면 중복을 허용

set

- 컬렉션을 상속받아 하나의 데이터를 저장하지만 순서를 보장하지 않고 중복 불가

map

- 컬렉션을 상속받지 않으며 중복 불가인 key, 중복 가능인 value의 두가지 값을 가짐

-> key = 주소, ID, 전화번호

-> value = 우편번호, password, 지역번호

ArrayList

- list 인터페이스를 구현한 클래스

- ArrayList의 객체를 생성할 땐 사용할 클래스를 입력해야 한다(wrapper클래스인 Integer등 사용 가능)

장점 - 배열은 연속되기 때문에 탐색이 빠르다

단점 - 삽입과 삭제가 불편하다(add를 이용할 시 해당 번지 이후의 값을 뒤로 밀어줘야 한다)

LinkedList

장점 - 삽입과 삭제가 빠르다(다른 값을 뒤로 밀지 않고 새로 객체를 만든 뒤 연결만 바꿔준다)

단점 - 탐색이 불편하다

Stack : 클래스가 있고 메서드를 호출할 때 사용된다.

마지막 입력이 먼저 출력된다(Last In First Out)

Queue : 인터페이스로서 입력된 순서대로 출력된다(First In First Out)

해당 인터페이스를 구현한 클래스를 통해서 객체를 생성해야 한다

컬렉션 프레임워크 - 데이터들을 효율적으로 관리하기 위해 표준화시킨 것

collection 인터페이스

- 하나의 객체를 저장한다(일반 변수를 저장할 수 없고 wrapper클래스의 객체로 변환된다)

- Iterator를 제공

-> list - 중복 허용 -> 순서대로 저장

- get()을 통해 원하는 번지에 직접 접근이 가능

-ArrayList, LinkedList등의 클래스를 통해 구현

-> set - 중복 불가 -> 순서를 보장하지 않음(중복이 없으면 순서대로 저장될 수도 있음)

- 직접 접근이 불가능 -> Iterator를 이용해 접근

- HashSet클래스를 통해 구현

map - 두 개의 객체를 저장한다

-> key - 중복 불가(ID, 주소, 전화번호 등)

-> value - 중복 가능(비밀번호, 우편번호, 지역번호 등)

- HashMap클래스를 통해 구현

자료 구조

Stack - Last In First Out(마지막에 저장된 것이 먼저 호출된다)

Queue - First In First Out(저장된 순서대로 호출된다)

지네릭스(Generics) - 타입을 체크해 타입 안정성을 높인다

- 컬렉션 프레임워크에서 제공하는 다양한 지네릭 클래스들을 사용하기 위해 학습

- 지네릭 클래스 용어

- 지네릭 클래스 타입변수를 제한하는 방법

- 와일드카드

열거형

- 제한된 값을 가지는 상수들을 효율적으로 관리하기 위해