중첩클래스(내부클래스)

특정 클래스와 관계를 맺는 경우에는 관계 클래스를 내부에 선언해 주는게 좋음

이를 중첩클래스(Nested class)라고 함.

인터페이스 또한내부에 선언할 수 있음 - 주로 UI 프로그래밍에 이용

인스턴스 내부 클래스

static 키워드 없이 선언된 내부 클래스

따라서 외부 클래스가 생성 되어야 사용할 수 있음

static 필드와 메서드를 선언할 수 없음

private으로 선언을 권장

정적 내부 클래스

static 키워드로 선언된 내부 클래스

모든 필드와 메서드를 선언 가능

외부 클래스 밖에서 정적 내부 클래스를 선언하기 위해서 외부클래스를 생성할 필요X

A.C c = new A.C();

지역 내부 클래스

메서드 내에 선언된 클래스

접근제한자(public, private), static 사용 불가

클래스 내부에 인스턴스만 선언 가능

내부클래스에서 사용하는 메서드의 지역변수나 매개변수는 final로 자동 컴파일됨

익명 내부 클래스

1. 익명 자식 객체

클래스를 상속하거나 인터페이스를 구현해야만 생성할 수 있음

자식클래스가 재사용되지 않고 해당 필드와 변수의 초기값으로만 사용되는 경우

익명 자식 객체를 생성하여 초기값으로 대입. 꼭 끝에 ; 붙여야 함

부모 클래스 필드or변수 = new 부모클래스(매개값){

//필드 or 메서드

};

익명 자식 객체에 사용된 필드와 메서드는 자식 객체 내부에서만 사용됨

따라서 외부클래스 밖에선 오버라이드된 메서드만 호출 가능

1. 익명 구현 객체

인터페이스 필드or변수 = new 인터페이스(매개값){

//필드 or 메서드

추상 메소드 구현

};

람다식

자바는 객체지향 프로그래밍이 패러다임이었더 90년대 개발됨

최근에 병렬 처리와 이벤트 지향 프로그래밍에 적합한 함수적 프로그래밍이 부각됨

자바8 부터 람다식 지원

람다식의 장점

코드가 간결해짐

대용량 데이터를 필터링 또는 매핑해서 쉽게 집계할 수 있음

람다식의 형태

매개변수를 가진 코드 블록이지만, 런타임 시에는 익명 구현 객체를 생성한다

람다식 => 매개변수를 가진 코드 블럭 => 익명 내부 객체

외부 자료를 사용하지 않음

여러 자료가 동시에 수행되는 병렬처리가 가능함

람다식의 기본 문법

타입 매개변수 = () ->{실행문};

ex) (int a) ->{sysout(a);}

매개변수 타입은 런타임시에 자동으로 인식되므로 보통 언급하지 않음

ex) (a) ->{sysout(a);}

매개 변수가 하나라면 괄호() 생략 가능, 실행문 하나라면 중괄호{} 생략 가능

ex) a ->sysout(a)

만약 매개 변수가 없다면 람다식에서 매개 변수 자리가 없어지므로 반드시

빈 괄호()를 사용해야 함

실행문이 반환문이 포함되어 있으면 반드시 중괄호{}를 포함해야 함

(x,y)->{return x+y;} —O

(x,y)->return x+y —---X

함수형 인터페이스

람다식을 선언하기 위한 인터페이스

@FunctionalInterface 사용

익명함수와 매개변수만으로 구현되므로 인터페이스는 단 하나의 메서드만을 선언해야 함

스트림

자바8부터 추가된 컬렉션의 저장 요소를 하나씩 참조하여 람다식으로 처리할 수 있도록 해주는 반복자

외부반복 : 개발자가 코드로 직접 컬렉션의 요소를 반복하여 가져옴

내부반복 : 개발자는 처리코드만 제공하여 컬렉션 내부에서 반복시킴

장점

개발자는 처리코드에만 집중

요소들의 반복순서를 변경

데이터 병렬 처리 가능

중간처리와 최종처리를 할 수 있음 p.798참조