**\*이전 6장 예제 20번 BankApplication**

**private** **static** **void** findAccount(String ano) {

// Account aabc = null;

**for** (**int** i = 0; i < *accountArray*.length; i++) {

**if** (*accountArray*[i] != **null**) {

**if** (ano == *accountArray*[i].getAno()) {

*findano* = i;

// String a = accountArray[i].getAno();

// String b = accountArray[i].getOwner();

// int c = accountArray[i].getBalance();

// aabc = new Account(a, b ,c);

**break**;

} **else** {

**continue**;

}

}

}

// return aabc;

}

**private** **static** Account[] *accountArray* = **new** Account[100];

처음 accountArray객체를 선언시 객체 배열의 크기만 설정해주었지 초기값은 설정하지않았기 때문에 null인지 확이해야한다

**if** (*accountArray*[i] != **null**) 이부분에서 nullpointException 오류가 발생하였는데 값을 비교하기전 해당 배열의값이 null인지 확인하는 문구를 넣어줘야 비교를 할수있다

1.findAccount메소드를 void형으로도 만들수있고 ->

**private** **static** **int** *dd* = 0;

하여 변수를 메모리에 상시 저장하여 공유하기 때문에 2 객체를 만들어 반환시키는 것이

더 좋은방법

2. Account객체를 반환하여 만들수도있다

-Account[] *accountArray* = **new** Account[100];

Account형식의 배열 100개를만드는데 이름이 accountArray->Account형식의 accountArray[0]~accountArray[99]배열이 만들어짐

**PPt 7장**

\*상속 extends : 자식클래스에서 상속받을 부모클래스를 선택하여 클래스의 멤버(필드, 메소드)받음

- 재사용, 반복코드줄임, 유지보수 편리성, 객체 다형성

- private. default접근을 갖는 필드와 메소드제외

- java는 단일상속만 가능

-자식객체 생성시 부모객체부터 생성후 자식객체 생성

-부모클래스를 상속받은 자식클래스가 생성자를 호출할 때 기본값 부모객체를 생성할때는 super()를 사용할필요없지만 super(매개값)- 매개값과 동일한 타입,개수,순서 맞는 부모생성자 호출

-상속받은 자식클래스에서 부모클래스의 메소드를 재정의 할 때 : 동일한 시그니처. 접근제한을 더강하게 오버라이딩 불가 -> 어떤 자식클래스는 해당 메소드가 재정의될필요가있기 때문에 오버라이드를 사용한다

-@Override 어노테이션 선언

-재정의시 부모메소드는 숨겨지는 효과발생->자식클래스 내부에서 오버라이딩된 부모클래스의 메소드를 호풀해야할경우 super.부모메소드();를통해 직접 접근할수있다

\*final class, method

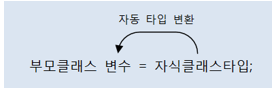
-클래스를 선언할 때 final 키워드를 class 앞에 붙이게되면 최종적인 클래스가되어 상속할수 없는 클래스가 된다(자식클래스를 만들수없다)

-메소드를 선언할 때 final 키워드를 붙이게 되면 오버라이딩 할 수 없는 메소드가 된다(자식클래스에서 재정의(Override 할수없음)

-protected

다른패키지의 클래스에서 직접 접근 할수없으나 다른패키지의 클래스가 해당 클래스를 상속받고있으면 spuer();생성자로 해당클래스의 생성자를 호출할수있으며 필드 메소드에 접근할수있다

\*다형성 : 같은타입이지만 실행결과가 다양한 객체대입 가능한성질



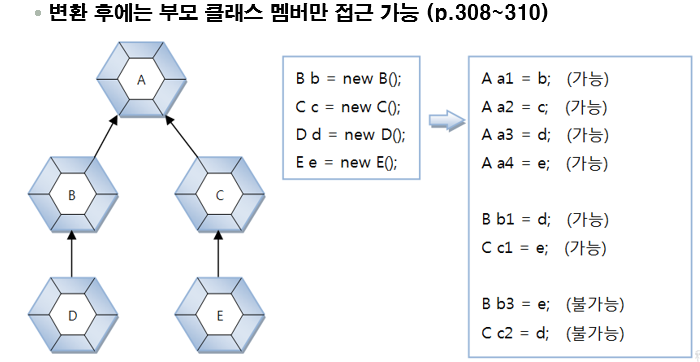
자식클래스 객체로 부모클래스의 메소드에 접근할수있다

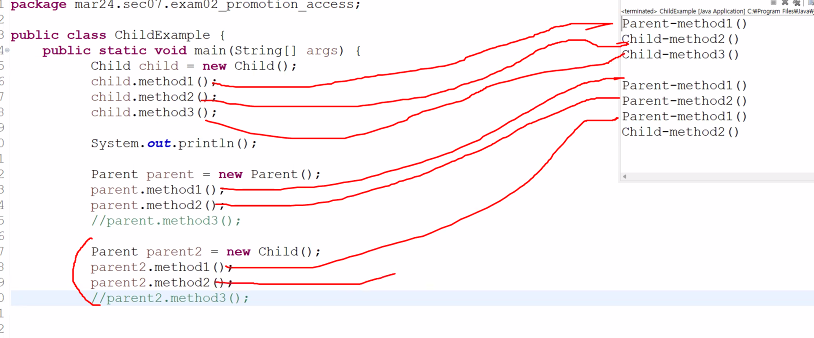
-다형성 이해 Child클래스는 Parent클래스를 상속 받았고

Parent클래스에는 method1, method2가 정의되어있으며

Child클래스에는 method2가 오버라이딩, method3가 정의되어있다

--자동 타입변환





-Child클래스의 객체를 생성자 Child()로 생성할때: Child클래스에 정의된 메소드 2,3과 상속받은 Parent클래스의 메소드1을 접근하여 사용할수있다

객체child는 Child클래스에 정의된 메소드 2,3을 그대로가져와 사용한다

-Parent클래스 의 객체 parent를 생성자 Parent()로생성하게되면

기존의 Parent클래스에 정의된 메소드 method1,과 method2를 그대로 사용하게되며 Parent클래스는 Child클래스를 상속 받지 않아 method3를 사용할수없다

-Parent클래스의 객체 parent2를 생성자 Child()로 생성할 때

Child()생성자로 생성하기 때문에 Parent클래스의 method2가 숨겨지게된다(Child클래스에 method2가 오버라이딩 되어있기때문

따라서 parent2에 method1은 기존 parent method1로 접근하게되고

method2는 parent method2가 숨겨져 child method2 가 실행되게되며

Parent클래스의 객체를 생성하였기 때문에 child method3에는 접근하지 못한다

필요할경우 강제타입변환(Casting)을통해서 자식클래스의 method3에 접근할수있다

((Child) parent2).method3();

\*필드의 다형성

자동타입변환이 왜필요할까 다형성을 구현하는 기술적 방법때문

다형성이란 동일한 타입을 사용하지만 다양한 결과가 나오는 성질을 말한다

\*매개변수의 다형성

-객체를 호출하면 to String이 자동호출된다

to String을 @Override하면 해당 객체의값이 Override한 값이 출력된다

-jav.util,Vector

Vector를 사용하면 배열등 추가 제거 찾는 것을 쉽게할수있음

\*모든클래스는 Object클래스를 상속받는다 -> to String() hashCode()

Object로 만든 객체는 모든 객체를 받아들일수있다