현실에서 상속은 부모가 자식에게 물려주는 행위를 말한다. 객체 지향 프로그램에서도 부모 클래스의 멤버를 자식 클래스에게 물려줄 수 있다.

상속은 이미 개발된 클래스를 재사용 해서 새로운 클래스를 만들기 때문에 코드의 중복을 줄여준다.

public class A{

int field1;

void method1(){}

}

public class B extedns A{

String field2;

void method2(){}

}

B b = new B();

b.field1 = 10l

b.method1();

b.field2 = “홍길동";

b.method2();

field1과 method1은 A에서 물려받은 애트리뷰트와 메소드이며 field2와 method2는 클래스 B에서 추가한 애트리뷰트와 메소드이다.

부모 생성자 호출

모든 객체는 클래스를 호출해야만 생성된다. 부모 객체도 예외는 아니다. 이를 위해 자식 객체에서는 super();를 통해 부모의 기본생성자를 호출한다.

super(매개값)는 매개값의 타입과 일치하는 부모 생성자를 호출한다. 만약 매개값의 타입과 일치하는 부모 생성자가 없을 경우 컴파일 오류가 발생한다.

public class People{

public String name;

public Stirng ssn;

public People(String name, String ssn){

this.name = name;

this.sssn = ssn;

}

}

public clsss Student extends People{

public int studentNo;

public Student(String name, String ssn, int studentNo){

super(name,ssn);

this.studentNo = studentNo;

}

}

위처럼 super(name, ssn)을 통해 부모 클래스인 People 클래스의 People(String name, String ssn) 생성자를 호출했다고 볼 수 있다.

메소드 재정의(@override)

상속된 메솓의 내용이 자식 클래스와 맞지 않을 경우, 자식 클래스에서 동일한 메소드를 재정의하는 것을 말한다. 메소드가 오버라이딩 되었다면 부모 객체의 메소드는 숨겨지기 때문에, 자식 객체에서 메소드를 호출하면 오버라이딩된 자식 메소드가 호출횐다.

예시

public class Calculator{

double areaCircle(double r){

System.out.println(”Calculator 객체의 areaCircle() 실행");

return 3.14159 \* r \* r;

}

}

public class Computer extedns Calculator {

double areaCircle(double r){

System.out.println(”Computer 객체의 areaCircle() 실행");

return math.PI\*r\*r;

}

}

Calculator의 areaCircle은 3.14159로 파이값을 계산하지만 Computer 의 areaCircle은 좀 더 정밀한 계산을 위해 Math.PI API를 사용해서 측정한다.

이처럼 메소드를 상속하면서 수정해야 되는 사항이 있을 때 오버라이딩을 통해 메소드를 재정의 가능하다.

연습 문제

다음의 출력결과를 작성하시오.

public class A{

void method1() {

System.out.println("메소드 1");

}

void method2() {

System.out.println("메소드 2");

}

void method3() {

System.out.println("메소드 3");

}

}

class B extends A{

void method2() {

System.out.println("변경된 메소드 2");

}

void method4() {

System.out.println("새로운 메소드 4");

}

}

답:

메소드 1

변경된 메소드 2

메소드 3

새로운 메소드 4

해설:

method1, method3 은 부모의 메소드를 실행한다.

method2는 오버라이딩 되어 자식의 메소드를 실행한다.

method4는 자식의 메소드를 실행한다.