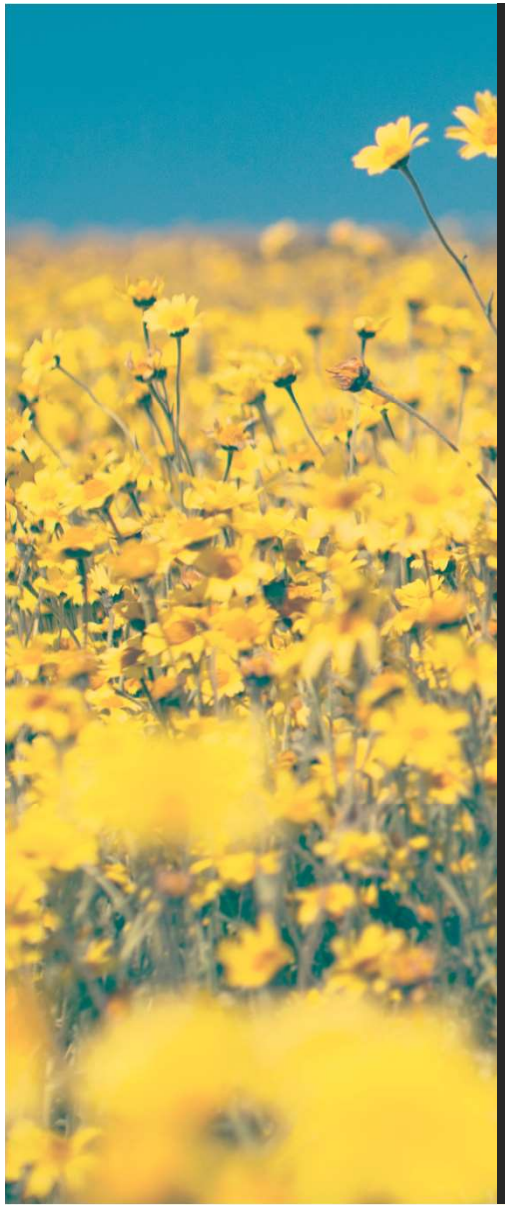


JAVA

GU, DA HAE


상 속
생 성 자
protected
오 버 라 이 딩
instanceof
abstract
interface
inner class
JAVA
객
체

JAVA

instanceof 연산자

참조변수가 참조하는 인스턴스의 자료형을 연산한다.

참조변수 instanceof 자료형

인스턴스의 자료형이 검사하는 자료형과 동일하다면 true, 그렇지 않다면 false를 반환한다.

JAVA

예제

```
class Box{
	public void noWrap(){
		System.out.println("NO WRAP!!");
	}
}

class PaperBox extends Box{
	public void paperWrap(){
		System.out.println("PAPER WRAP!!");
	}
}

class RedPaperBox extends PaperBox{
	public void redPaperWrap(){
		System.out.println("RED PAPER WRAP!!");
	}
}
```

```
public class Ex02{
	public static void howWrap(Box b){
		if(b instanceof Box)
			b.noWrap();
		else if(b instanceof PaperBox)
			((PaperBox)b).paperWrap();
		else
			((RedPaperBox)b).redPaperWrap();
	}
	public static void main(String[] args){
		Box b1 = new Box();
		PaperBox b2 = new PaperBox();
		RedPaperBox b3 = new RedPaperBox();

		howWrap(b1);
		howWrap(b2);
		howWrap(b3);
	}
}
```

JAVA

해보기

바로 앞의 예제를 instanceof 연산자를 사용하지 않은 형태로 변경해보자. 메서드 오버라이딩을 활용하면 쉽게 문제를 해결할 수 있다.

```
public static void howWrap(Box b){
    b.warp();
}
```

JAVA

상속을 위한 관계

상속을 사용하려면 IS-A 관계가 성립되어야 한다.

컴퓨터 --- (+이동성) ---→ 노트북

노트북은 컴퓨터의 일종이다.
노트북 is a 컴퓨터.

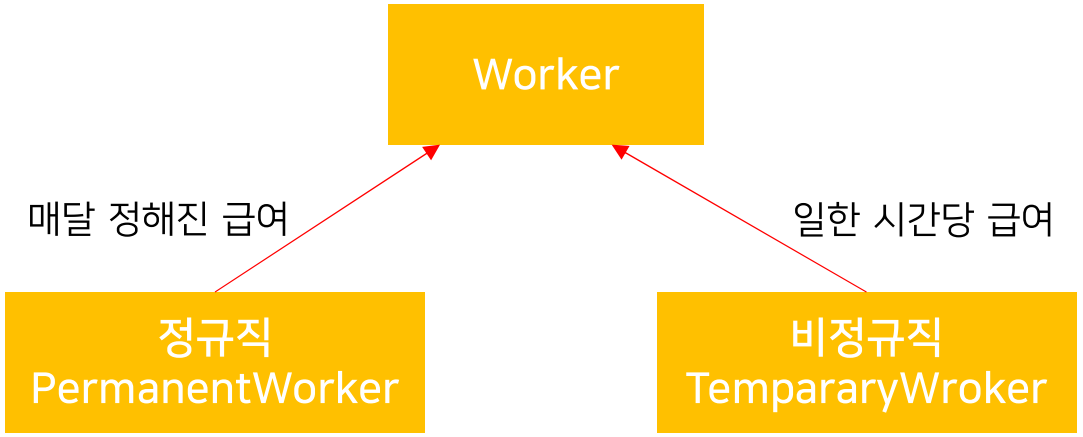
하위 클래스는 상위 클래스의 멤버에, 하위 클래스만의 특성이 추가된다. 이런 관계를 관계를 IS-A관계라고 표현한다.

JAVA

예제

```
class Worker{// 데이터 클래스
    private String name;
    private int salary;
    public Worker(String name, int salary){
        this.name = name;
        this.salary = salary;
    }
    public int getPay(){ return salary; }
    public void showSalaryInfo(){
        System.out.println("name : " + name);
        System.out.println("salary : " + getPay());
    }
}
class EmployeeHandler{// 핸들러 클래스, UI(User Interface)
    private Worker[] workerList;
    private int empNum;
    public EmployeeHandler(){
        workerList = new Worker[50];
        empNum = 0;
    }
    public void addEmployee(Worker w){workerList[empNum++]=w;}
    public void showAllSalaryInfo(){
        for(int i=0; i< empNum ; i++){
            e.showSalaryInfo();
            System.out.println("================");
        }
    }
    public void showTotalSalary(){
        int sum = 0;
        for(int i=0; i< empNum ; i++) sum += e.getPay();
        System.out.println("Salary Sum : " + sum);
    }
}
public class Ex01{
    public static void main(String[] args){
        // 직원 관리를 목적으로 설계된 컨트롤 클래스 인스턴스 생성
        EmployeeHandler handler = new EmployeeHandler();
        // 직원 등록
        handler.addEmployee(new Worker("Jain", 1000));
        handler.addEmployee(new Worker("Louis", 1500));
        handler.addEmployee(new Worker("Keneth", 2000));

        handler.showAllSalaryInfo();
        handler.showTotalSalary();
    }
}
```

JAVA


Worker
매달 정해진 급여
일한 시간당 급여
정규직
PermanentWorker
비정규직
TempararyWroker

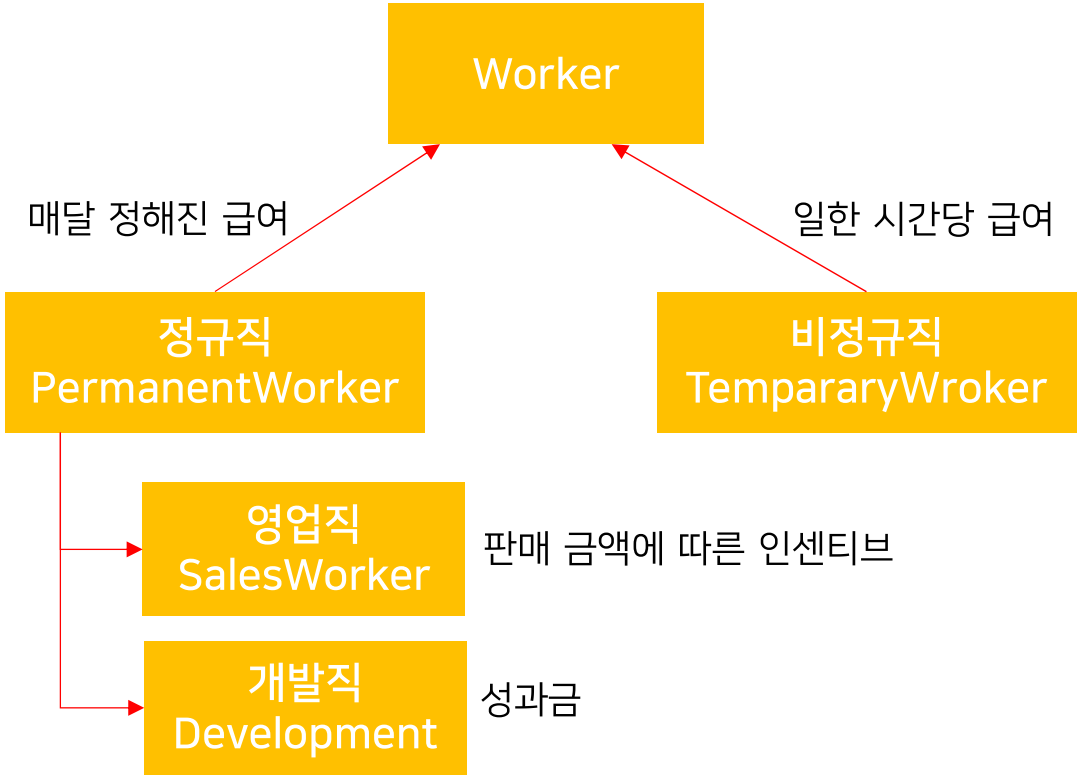
JAVA

예제

```
class Worker{

    private String name

    public Worker(String name) {this.name = name;}

    public void showName() {

        System.out.println("name : " + name);

    }}


class PermanentWorker extends Worker{
    private int salary;
    public PermanentWorker(String name, int salary){
        super(name);
        this.salary = salary;
    }
    public int getPay(){ return salary; }
    public void showSalaryInfo(){
        showName();
        System.out.println("Salary : " + getPay());
    }
}
```

```
class TemporaryWorker extends Worker{
    private int worktime; // 일한 시간
    private int payPerHour; // 일한 시간 당 금액
    public TemporaryWorker(String name, int pay){
        super(name);
        worktime = 0;
        payPerHour = pay;
    }
    public void addWorkTime(int time) { workTime += time; }
    public int getPay(){ return workTime * payPerHour; }
    public void showSalaryInfo(){
        showName();
        System.out.println("Salary : " + getPay());
    }
}
```

JAVA


Worker
매달 정해진 급여
일한 시간당 급여
정규직
PermanentWorker
비정규직
TempararyWroker
영업직
SalesWorker
판매 금액에 따른 인센티브
개발직
Development
성과금

JAVA

예제

```
class SalesWorker extends PermanentWorker{
    private int salesResult; // 판매금
    private double bonusRatio; // 인센티브 비율

    public SalesWorker(String name, int pay, double bonus){
        super(name, pay);
        salesResult = 0;
        bonusRatio = bonus;
    }
    public void addSalesReulst(int result){ salesResult += result;}
    public int getPay(){return (int) (super.getPay() + salesResult * bonusRatio);}
    public void showSalaryInfo(){
        showName();
        System.out.println("Salary : " + getPay());
    }
}

class EmployeeHandler{ // 앞과 같으므로 생략 }
```

```
public class Ex01{
    public static void main(String[] args){
      // 직원 관리를 목적으로 설계된 컨트롤 클래스 인스턴스 생성
      EmployeeHandler handler = new EmployeeHandler();

      // 직원 등록
      handler.addEmployee(new PermanentWorker("Jain", 1000));
      handler.addEmployee(new PermanentWorker("Louis", 1500));
      handler.addEmployee(new PermanentWorker("Keneth", 2000));
      handler.addEmployee(new TemporaryWorker("Lu", 100));
      handler.addEmployee(new TemporaryWorker("Grace", 110));
      handler.addEmployee(new SalesWorker("Smith", 500, 0.1));
      handler.showAllSalaryInfo();
      handler.showTotalSalary();
    }

}

// 실행 ERROR!
```

JAVA

예제

```
class Worker{
	private String name;

	public Worker(String name){
		this.name = name;
	}
	public void showName(){
		System.out.println("Name : " + name);
	}
	public int getPay(){return 0;}
	public void showSalaryInfo(){}
}
```

```
class PermanentWorker extends Worker{…}

class TemporaryWorker extends Worker{…}

class SalesWorker extends TemporaryWorker{…}

class EmployeeHandler{…}

public class Ex01{…}
```

// EmployeeHandler 클래스는 모든 정규직, 비정규직, 영업직 클래스를 Worker 클래스로 생각하고 관리한다. 이 점은 절차지향에서는 절대 볼 수 없는, 객체지향만의 장점(다형성)이다. 또한 상속을 통해 여러 클래스의 공통적인 특징을 정의할 수 있다.

JAVA

해보기

앞의 예제를 확장하여 다음 특성에 해당하는 ForeignSalesWorker 클래스를 추가로 정의해보자.

영업직 직원 중 일부는 해외로 장기출장을 간다. 위험한 지역으로 출장을 가는 영업직들에게 위험수당을 지급하고자 한다. 위험수당의 지급방식은 '위험 노출도'에 따라서 다음과 같이 결정된다.

-리스크A : 영업직의 기본급여와 인센티브 합계 총액의 30%를 추가로 지급한다.

-리스크B : 영업직의 기본급여와 인센티브 합계 총액의 20%를 추가로 지급한다.

-리스크C : 영업직의 기본급여와 인센티브 합계 총액의 10%를 추가로 지급한다.

[실행 결과]

name : hong
salary : 1700
risk pay : 510
sum : 2210

name : han
salary : 1700
risk pay : 340
sum : 2040

name : gu
salary : 1700
risk pay : 170
sum : 1870