Inheritance(@Override) Ⅱ - relationships for Inheritance

상속을 위한 관계

- 사람과 학생,교수가 있다면 무엇을 상위 Class로 둘까?
  → 사람이 상위 Class가 된다. **왜냐하면 학생은 사람이니까!**
  학생은 사람이 가져야될 기본적인 기능을 가지고 있어야 한다.

i. **상속을 위한 최소한의 기본조건은 IS - A 관계가 성립해야 한다.**

ii. **사람 → 학생 → 중학생, 고등학생, 대학생** 상속 관계를 집합의 관계에서 보면
  - 규모적 측면
    - 상위 Class로 올라갈수록 규모가 광범위하게 커지고, 범용적이게 된다.
    - 하위 Class로 내려갈수록 규모가 작아지고, 구체화된다.

  - 기능적 측면
    - 상위 Class로 올라갈수록 기능이 적어진다.
    - 하위 Class로 내려갈수록 기능이 확장된다.
      (하위 Class로 내려갈수록 기능을 덧붙히기 때문이다.)

IS - A 기반 상속의 예 : 노트북(하위)은 컴퓨터(상위)에 속한다

```
class Computer {
     String owner;
     public Computer(String name){ this.owner = name; }
     public void calculate(){ }
}

class NotebookComp extends Computer {
     int battery;

     public NotebookComp(String name, int initChag){
          super(name);
          this.battery = initChag;
     }

     public void charging(){ }
     public void moveCal(){ }
}

class TabletNotebook extends NotebookComp {
     String regstPenModel;

     public TabletNotebook(String name, int initChag, String pen){
          super(name, initChag);
          this.regstPenModel = pen;
     }

     public void write(String penInfo){ }
}
```

HAS - A 기반 상속의 예 : 경찰(하위)이 총(상위)을 소유하고 있다.

```
class Gun {
     int bullet;
     public Gun(int bnum){ this.bullet = bnum; }

     public void shut(){
          System.out.println("BBANG!");
          bullet--;
     }
}

class Police extends Gun {
     int handcuffs; // 소유한 수갑의 수

     public Police (int bnum, int bcuff){
          super(bnum);
          handcuffs = bcuff;
     }

     public void putHandCuff(){
          System.out.println("SNAP!");
          handcuffs--;
     }
}
```

- **상속은 강한 연결고리를 형성한다. (상속의 단점)**
  경찰은 총을 상속 받았으므로 무조건 총을 가지고 있어야 한다.
  그러나 총을 가지고 있지 않는 경찰도 있고, 경찰이 무슨 일을 하던 총을 항상 가지고 다녀야만 한다.

HAS - A 관계에 복합관계를 적용한 경우

- **HAS - A 관계의 복합관계는 강한 연결고리를 형성하지 않기 때문에, HAS - A 관계는 상속으로 표현하지 않는다.**

```
class Gun {
     int bullet;
     public Gun(int bnum){ this.bullet = bnum; }

     public void shut(){
          System.out.println("BBANG!");
          bullet--;
     }
}

class Police {
     int handcuffs; // 소유한 수갑의 수
     Gun pistol;

     public Police (int bnum, int bcuff){
          handcuffs = bcuff;

          if(bnum!=0)
               pistol = new Gun(bnum); // 총이 필요하면 총을 생성하면 된다.
          else
               pistol = null; // 총이 불필요하면 null만 전달해주면 된다!
     }

     public void putHandCuff(){
          System.out.println("SNAP!");
          handcuffs--;
     }

     public void shut(){
```

```
          if(pistol==null){
               System.out.println("hut BBANG!");
          }
          else
               pistol.shut();
     }

}
```
