다형성 – 객체의 타입 변환 : 업캐스팅과 다운캐스팅

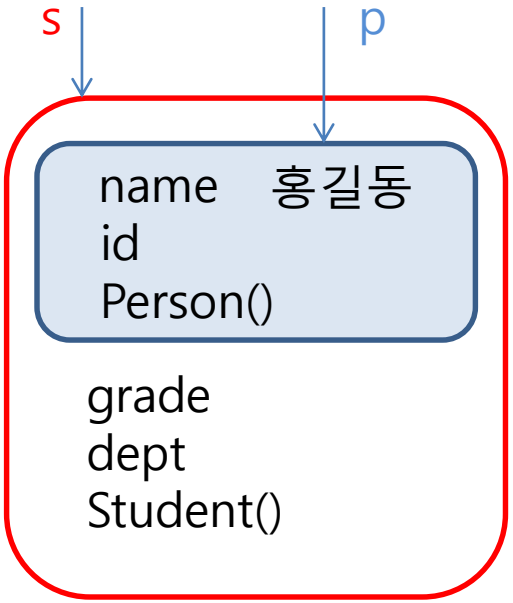
```
class Person {
        String name;
        String id;
        public  Person(){}
        public  Person(String name){
                this.name = name;
        }
}
```

```
class Student extends Person{
        String grade;
        String dept;

        public  Student(String name){
                super(name);
        }
}
```

```
public class UpcastingEx {
    public static void main(String[] args) {
        Person p;
        Student s = new Student("홍길동");
        p=s; //업캐스팅 발생
        System.out.println(p.name);

        p.grade="A";     //error
        p.dept="com";   //error
        }
}
```



```
public class DowncastingEx {

    public static void main(String[] args) {

            Person p= new Student("홍길동"); // 업캐스팅 발생
            Student s;
            s = (Student)p; //다운캐스팅 발생

            System.out.println(s.name);

            s.grade="A";     //오류 없음
            s.dept="com";   //오류 없음
            }
}
```

* 다형성이 필요한 이유를 생각해 본다. – 도형 객체의 삽입과 삭제를 통하여 이해해 보자.

Method Overriding

도형

```
class DObject {


  public  void draw() {
          System.out.println("DObject  draw");
  }
}
```

선

```
class Line extends DObjet {

  public  void draw() {
            System.out.println("Line");
  }
}
```

사각형

```
class Rect extends DObjet {

  public  void draw() {
            System.out.println("Rect");
  }
}
```

원

```
class Circle extends DObjet {

  public  void draw() {
            System.out.println("Circle");
  }
}
```

```
Public class MethodOverridingEx {

    public static void main(String[] args) {

            DObject obj = new DObject();
            DObject l = new Line();
            DObject r = new Rect();
            DObject c = new Circle();

            obj.draw() //?

            l.draw();  // ?
            r.draw();  // ?
            c.draw();  //?

            Line line = new Line();
            Dobject p = line;

            p.draw(); //?

    }
}
```

```
Public class MethodOverridingEx {

    public static void main(String[] args) {

        DObject obj = new DObject();
        DObject l = new Line();
        DObject r = new Rect();
        DObject c = new Circle();

        obj.draw() // Dobject Draw

        l.draw();  //  Line     메소드 오버라이딩
        r.draw();  //  Rect     메소드 오버라이딩
        c.draw();  // Circle    메소드 오버라이딩

        Line line = new Line();
        Dobject p = line;

        p.draw(); // Line      메소드 오버라이딩

    }
}
```

메소드 오버라이딩의 활용 – 다형성의 필요성 이해

도형을 상속받은 각각의 Line , Rect, Circle을 하나의 타입으로 묶어 검색, 수정 , 삭제 하려고 할때 부모 클래스 타입으로 묶어 정리 할 수있다.

```
DObject p = new Line();
```

또한 필요한 부분들을 메소드 오버라이딩 할 수도 있다.