**객체의 정의**

객체는 효율적으로 정보를 관리하기 위해서 사람들이 의미를 부여하고 분류하는 논리적 단위

객체는 속성과 기능의 집합이며 속성은 변수로 기능은 메서드로 정의한다.

**클래스의 정의**

클래스는 객체를 정의해 놓은 것이고 객체를 생성하는 데 사용된다.

**객체지향 언어의 주요 개념(상속, 캡슐화, 다형성)**

- 상속

이미 만든 객체와 비슷하지만 속성과 메서드가 약간 다른 객체를 생성할 때 기존의 클래스에서 속성과 메서드를 상속하고 더 필요한 속성과 메서드를 추가 및 오버라이딩한다. 상속을 통해 코드를 간경하게 할 수 있고 코드의 재사용성을 높일 수 있다.

- 캡슐화

객체를 사용하는 쪽에서 객체의 내부 처리를 자세히 알 필요가 없다. 따라서 객체 단위로 기능이나 정보를 캡슐화한다. 또 숨겨야 하는 정보와 공개하는 정보를 구분할 수 있다. 정보를 은닉할 수 있다.

- 다형성

메서드 오버로딩, 오버라이딩을 통해 같은 메서드에 다른 구현이 여러 개 가능하다.

오버로딩은 동일한 이름에 다른 시그니처를 가진 메서드를 생성하는 것을 의미한다. 규칙은 매개변수의 타입이나 개수가 다르고 이름이 같은 메서드를 정의할 때 오버로딩된 메서드를 생성했다고 할 수 있다.

반면 오버라이딩은 부모 클래스의 메서드를 자식클래스에서 재정의하는 것을 의미한다. 따라서 상속 관계를 갖는 클래스에서만 정의할 수 있고 조건으로는 동일한 메서드 이름, 동일한 매개변수 개수 및 타입이어야 한다. 오버로딩 오버라이딩 모두 반환값은 중요하지 않은데 같은 이름, 같은 매개변수이고 반환값만 다른 메서드를 선언한 경우 컴파일 오류가 된다.

* 예시 코드 : class ShapeTest.main();

**public** **class** ShapeTest {

**public** **static** **void** main(String[] args) {

// 1. 상속

// Point 클래스를 상속한 ColorPoint 클래스로 만들어진 인스턴스는

// 별도로 정의하지 않은 부모 클래스의 draw() 메서드를 사용할 수 있다.

// visible()메서드는 부모 클래스의 메서드를 오버라이딩하여 다르게 정의하였다.

ColorPoint colorPoint = **new** ColorPoint(50, 50, "빨강");

colorPoint.visible(**true**);

colorPoint.draw();

// 2. 캡슐화

// Point 클래스에 정의된 변수 x와 y는 접근제한자가 private로 정의되어

// Point point = new Point();

// point.x = 100;

// 위와 같이 x의 갑을 설정할 수 없다.

// 이는 캡슐화되어 정보 은닉된 케이스라 할 수 있다.

// 또한 Point(int x, int y) 생성자는 매개변수와 반환타입을 통해

// 내부 처리를 모르더라도 사용할 수 있다.

Point point = **new** Point(10, 20);

// 3. 다형성 - 오버로딩

// Circle 클래스의 calculateArea() 메서드는 매개변수가 없는 경우와

// calculateArea(int radius) 와 같이 매개변수가 설정된 경우의 두 가지로 오버로딩되었다.

Circle circle = **new** Circle();

circle.setRadius(20);

System.***out***.println("circle.calculateArea():"+circle.calculateArea());

System.***out***.println("circle.calculateArea(30):"+circle.calculateArea(30));

// 3. 다형성 - 오버라이딩

// ColorPoint 클래스의 visible(boolean isVisible) 메서드는 Point 클래스의 매서드르 오버라이딩하여

// 다르게 정의하였다.

Point pointOverriding = **new** ColorPoint(30, 50, "파랑");

pointOverriding.visible(**true**);

pointOverriding.visible(**false**);

}

}