상속

상속(inheritance)

상속(inheritance)의 정의와 장점

▸ 상속이란?

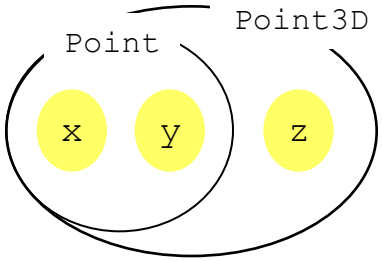
- 기존의 클래스를 재사용해서 새로운 클래스를 작성하는 것.
- 두 클래스를 조상과 자손으로 관계를 맺어주는 것.
- 자손은 조상의 모든 멤버를 상속받는다.(생성자, 초기화블럭 제외)
- 자손의 멤버개수는 조상보다 적을 수 없다.(같거나 많다.)

```
class Point {
    int x;
    int y;
}
```

```
class 자손클래스 extends 조상클래스 {
    // ...
}
```

```
class Point3D {
    int x;
    int y;
    int z;
}
```

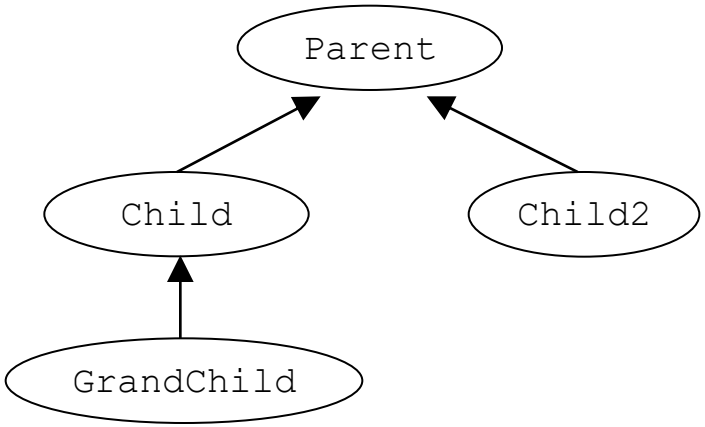
```
class Point3D extends Point {
    int z;
}
```

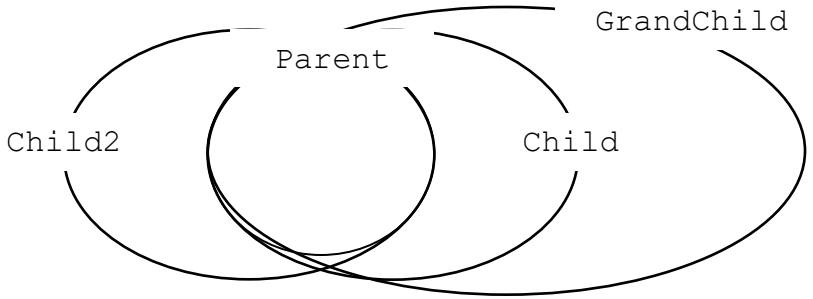


클래스간의 관계 – 상속관계(inheritance)

- 공통부분은 조상에서 관리하고 개별부분은 자손에서 관리한다.
- 조상의 변경은 자손에 영향을 미치지만, 자손의 변경은 조상에 아무런 영향을 미치지 않는다.

```
class Parent {}
class Child extends Parent {}
class Child2 extends Parent {}
class GrandChild extends Child {}
```

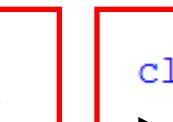




클래스간의 관계 – 포함관계(composite)

- 포함(composite)이란?
  - 한 클래스의 멤버변수로 다른 클래스를 선언하는 것
  - 작은 단위의 클래스를 먼저 만들고, 이 들을 조합해서 하나의 커다란 클래스를 만든다.

```
class Circle {
    int x; // 원점의 x좌표
    int y; // 원점의 y좌표
    int r; // 반지름(radius)
}
```



```
class Circle {
    Point c = new Point(); // 원점
    int r; // 반지름(radius)
}
```

```
class Point {
    int x;
    int y;
}
```

```
class Car {
    Engine e = new Engine(); // 엔진
    Door[] d = new Door[4];  // 문, 문의 개수를 넷으로 가정하고 배열로 처리했다.
    //...
}
```

클래스간의 관계결정하기 – 상속 vs. 포함

- 가능한 한 많은 관계를 맺어주어 재사용성을 높이고 관리하기 쉽게 한다.

-'is-a'와 'has-a'를 가지고 문장을 만들어 본다.

원(Circle)은 점(Point)**이다**. - Circle **is a** Point.
원(Circle)은 점(Point)을 **가지고 있다**. - Circle **has a** Point.

상속관계 - '~은 ~이다.(is-a)'
포함관계 - '~은 ~을 가지고 있다.(has-a)'

```
class Point {
    int x;
    int y;
}
```

```
class Circle extends Point{
    int r; // 반지름(radius)
}
```

```
class Circle {
    Point c = new Point(); // 원점
    int r; // 반지름(radius)
}
```

클래스간의 관계결정하기

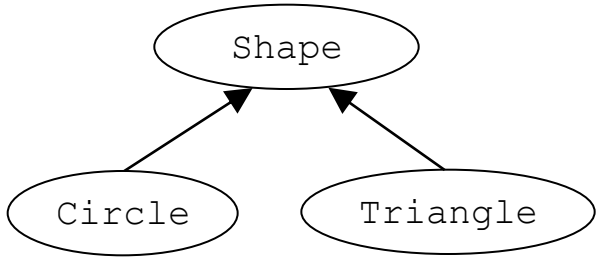
- 원(Circle)은 도형(Shape)이다.(A Circle is a Shape.) : 상속관계
- 원(Circle)은 점(Point)를 가지고 있다.(A Circle has a Point.) : 포함관계

```
class Shape {
    String color = "blue";
    void draw() {
        // 도형을 그린다.
    }
}
```

```
class Point {
    int x;
    int y;

    Point() {
        this(0,0);
    }

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

```
class Circle extends Shape {
    Point center;
    int r;

    Circle() {
        this(new Point(0,0),100);
    }

    Circle(Point center, int r) {
        this.center = center;
        this.r = r;
    }
}
```

```
class Triangle extends Shape {
    Point[] p;

    Triangle(Point[] p) {
        this.p = p;
    }

    Triangle(Point p1, Point p2, Point p3) {
        p = new Point[]{p1,p2,p3};
    }
}
```




```
Circle c1 = new Circle();
Circle c2 = new Circle(new Point(150,150),50);

Point[] p = {new Point(100,100),
             new Point(140,50),
             new Point(200,100)
            };
Triangle t1 = new Triangle(p);
```

1.4 단일상속(single inheritance)

- Java는 단일상속만을 허용한다.(C++은 다중상속 허용)

```
class TVCR extends TV, VCR {       // 이와 같은 표현은 허용하지 않는다.
    //...
}
```

- 비중이 높은 클래스 하나만 상속관계로, 나머지는 포함관계로 한다.

```
class Tv {
    boolean power;   // 전원상태(on/off)
    int channel;     // 채널

    void power() {  power = !power; }
    void channelUp() {  ++channel; }
    void channelDown() {    --channel; }
}
```

```
class VCR {
    boolean power;   // 전원상태(on/off)
   int counter = 0;
    void power() {  power = !power; }
    void play() { /* 내용생략*/ }
    void stop() { /* 내용생략*/ }
    void rew() { /* 내용생략*/ }
    void ff() { /* 내용생략*/ }
}
```

상속

포함

```
class TVCR extends Tv {
    VCR vcr = new VCR();
    int counter = vcr.counter;

    void play() {
        vcr.play();
    }

    void stop() {
        vcr.stop();
    }

    void rew() {
        vcr.rew();
    }

    void ff() {
        vcr.ff();
    }
}
```

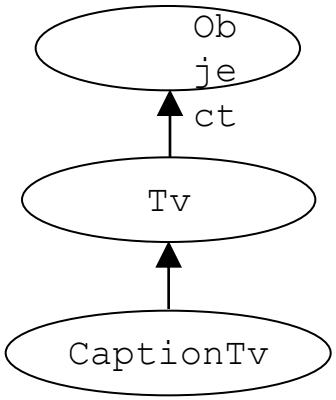
Object클래스 – 모든 클래스의 최고조상

- 조상이 없는 클래스는 자동적으로 Object클래스를 상속받게 된다.
- 상속계층도의 최상위에는 Object클래스가 위치한다.
- 모든 클래스는 Object클래스에 정의된 11개의 메서드를 상속받는다.

  toString(), equals(Object obj), hashCode(), ...

```
class Tv {
    // ...
}

class CaptionTv extends Tv {
    // ...
}
```

```
class Tv extends Object {
    // ...
}

class CaptionTv extends Tv {
    // ...
}
```



오버라이딩(overriding)

오버라이딩(overriding)이란?

"조상클래스로부터 상속받은 메서드의 내용을 상속받는 클래스에 맞게 변경하는 것을 오버라이딩이라고 한다."

* override - vt. '~위에 덮어쓰다(overwrite).', '~에 우선하다.'

```
class Point {
      int x;
      int y;

      String getLocation() {
            return "x :" + x + ", y :"+ y;
      }
}

class Point3D extends Point {
      int z;
      String getLocation() {        // 오버라이딩
            return "x :" + x + ", y :"+ y + ", z :" + z;
      }
}
```

오버라이딩의 조건

1. 선언부가 같아야 한다.(이름, 매개변수, 리턴타입)

2. 접근제어자를 좁은 범위로 변경할 수 없다.

   - 조상의 메서드가 protected라면, 범위가 같거나 넓은 protected나 public으로만 변경할 수 있다.

3. 조상클래스의 메서드보다 많은 수의 예외를 선언할 수 없다.

```
class Parent {
    void parentMethod() throws IOException, SQLException {
        // ...
    }
}

class Child extends Parent {
    void parentMethod() throws IOException {
        //..
    }
}

class Child2 extends Parent {
    void parentMethod() throws Exception {
        //..
    }
}
```

오버로딩 vs. 오버라이딩

오버로딩(overloading) - 기존에 없는 새로운 메서드를 정의하는 것(new)
오버라이딩(overriding) - 상속받은 메서드의 내용을 변경하는 것(change, modify)

```
class Parent {
    void parentMethod() {}
}

class Child extends Parent {
    void parentMethod() {}          // 오버라이딩
    void parentMethod(int i) {}     // 오버로딩

    void childMethod() {}
    void childMethod(int i) {}      // 오버로딩
    void childMethod() {}           // 에러!!! 중복정의임
}
```

super – 참조변수(1/2)

- this – 인스턴스 자신을 가리키는 참조변수. 인스턴스의 주소가 저장되어있음
  모든 인스턴스 메서드에 지역변수로 숨겨진 채로 존재
- super – this와 같음. 조상의 멤버와 자신의 멤버를 구별하는 데 사용.

```
class Parent {
    int x=10;
}

class Child extends Parent {
    int x=20;
    void method() {
        System.out.println("x=" + x);
        System.out.println("this.x=" + this.x);
        System.out.println("super.x="+ super.x);
    }
}
```

```
class Parent {
    int x=10;
}

class Child extends Parent {
    void method() {
        System.out.println("x=" + x);
        System.out.println("this.x=" + this.x);
        System.out.println("super.x="+ super.x);
    }
}
```

```
public static void main(String args[]) {
    Child c = new Child();
    c.method();
}
```

super – 참조변수(2/2)

- this – 인스턴스 자신을 가리키는 참조변수. 인스턴스의 주소가 저장되어있음
  모든 인스턴스 메서드에 지역변수로 숨겨진 채로 존재
- super – this와 같음. 조상의 멤버와 자신의 멤버를 구별하는 데 사용.

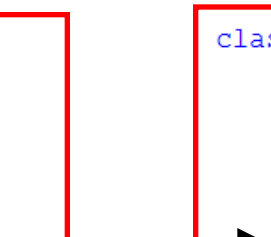
```
class Point {
      int x;
      int y;

      String getLocation() {
            return "x :" + x + ", y :"+ y;
      }
}

class Point3D extends Point {
      int z;
      String getLocation() {       // 오버라이딩
       // return "x :" + x + ", y :"+ y + ", z :" + z;
          return super.getLocation() + ", z :" + z; // 조상의 메서드 호출
      }
}
```

super() – 조상의 생성자(1/3)

- 자손클래스의 인스턴스를 생성하면, 자손의 멤버와 조상의 멤버가 합쳐진 하나의 인스턴스가 생성된다.
- 조상의 멤버들도 초기화되어야 하기 때문에 자손의 생성자의 첫 문장에서 조상의 생성자를 호출해야 한다.

Object클래스를 제외한 모든 클래스의 생성자 첫 줄에는 생성자(같은 클래스의 다른 생성자 또는 조상의 생성자)를 호출해야한다.
그렇지 않으면 컴파일러가 자동적으로 'super();'를 생성자의 첫 줄에 삽입한다.

```
class Point {
    int x;
    int y;

    Point() {
        this(0,0);
    }

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```



```
class Point extends Object {
    int x;
    int y;

    Point() {
        this(0,0);
    }

    Point(int x, int y) {
        super(); // Object();
        this.x = x;
        this.y = y;
    }
}
```

super() – 조상의 생성자(2/3)

```
class Point {
    int x;
    int y;

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    String getLocation() {
        return "x :" + x + ", y :"+ y;
    }
}

class Point3D extends Point {
    int z;

    Point3D(int x, int y, int z) {

        this.x = x;
        this.y = y;
        this.z = z;
    }

    String getLocation() {  // 오버라이딩
        return "x :" + x + ", y :"+ y + ", z :" + z;
    }
}
```

```
Point(int x, int y) {
    super(); // Object();
    this.x = x;
    this.y = y;
}
```

```
class PointTest {
    public static void main(String args[]) {
        Point3D p3 = new Point3D(1,2,3);
    }
}
```

```
---------- javac ----------
PointTest.java:24: cannot find symbol
symbol  : constructor Point()
location: class Point
        Point3D(int x, int y, int z) {
                                     ^
1 error
```

```
Point3D(int x, int y, int z) {
    super(); // Point()를 호출
    this.x = x;
    this.y = y;
    this.z = z;
}
```

```
Point3D(int x, int y, int z) {
    // 조상의 생성자 Point(int x, int y)를 호출
    super(x,y);
    this.z = z;
}
```