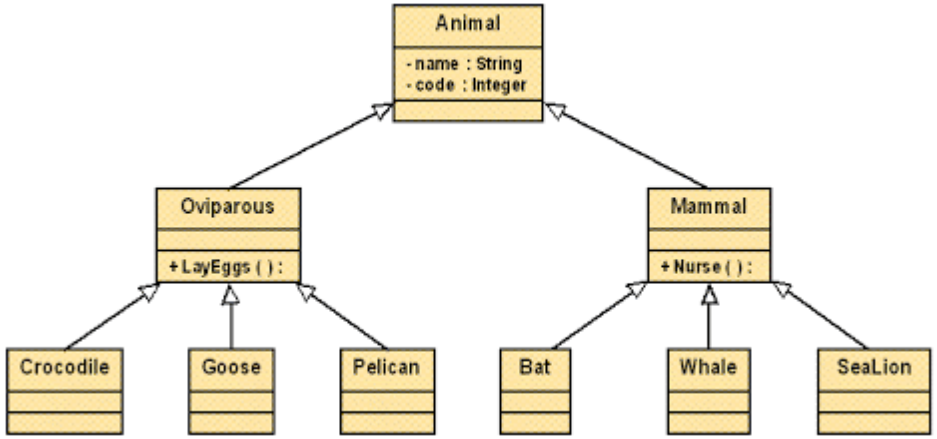
Subclases/Herencia

- En Java se pueden definir *subclases* o clases que *heredan* estado y comportamiento de otra clase (la *superclase*) a la que amplían, en la forma:

```
class SubClase extends SuperClase {
    //...
}
```

- En Java sólo se permite *herencia simple*, por lo que pueden establecerse jerarquías de clases.



- Todas las clases son herederas directas o indirectas de la clase **`Object`** del paquete **`java.lang`** que recoge los comportamientos básicos que debe presentar cualquier clase.
- Cada clase que creamos en Java hereda los métodos de instancia de **`Object`**.

Subclases/Herencia

- Por razones de seguridad o de diseño, se puede prohibir la definición de subclases para una clase etiquetándola con **`final`**.
  - Recordad que una subclase puede sustituir a su superclase donde ésta sea necesaria y tener comportamientos muy distintos
- El compilador rechazará cualquier intento de definir una subclase para una clase etiquetada con **`final`**.
- También se pueden etiquetar con **`final`**:
  - métodos, para evitar su redefinición en alguna posible subclase, y
  - variables, para mantener constantes sus valores o referencias.

Clases abstractas

- A veces, una clase puede modelar una abstracción y , aunque sea capaz de proporcionar una definición del comportamiento esperado, no le es posible concretar una implementación adecuada o suficiente de todos los métodos definidos.

- Estas clases se etiquetan como **`abstract`** y pueden tener métodos no implementados, también etiquetados como **`abstract`**.

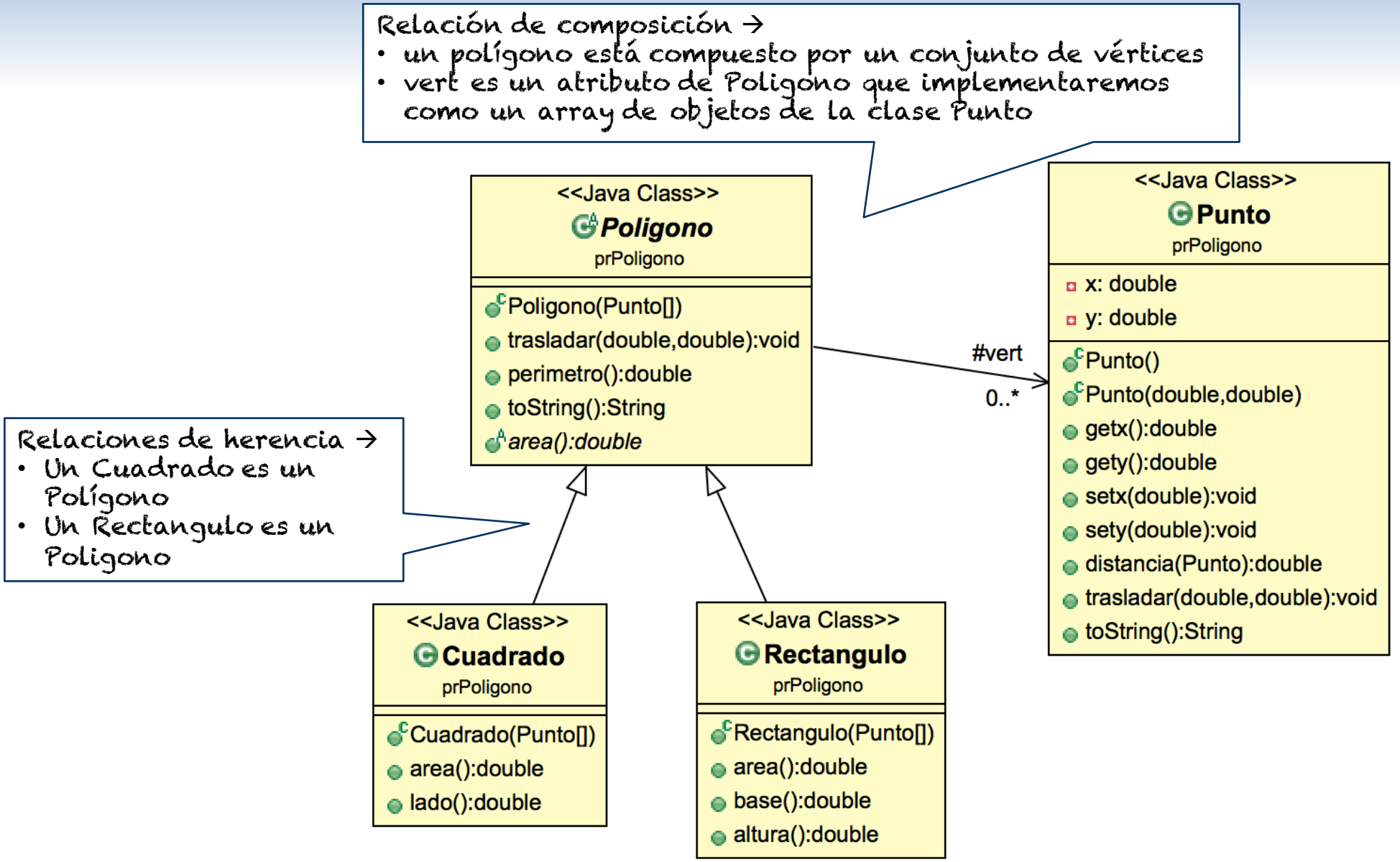
- Se utilizan para formar jerarquías.

- Se pueden utilizar como tipos, pero no se pueden crear instancias suyas.

- Deben tener subclases que no sean abstractas para generar objetos.

Clases abstractas

- Ejemplo: clase abstracta `Poligono`



Clases abstractas

```
public abstract class Poligono {
    protected Punto[] vert;

    public Poligono(Punto[] vs) {
        vert = vs;
    }
    public void trasladar(double a, double b) {
        for (Punto pto : vert) pto.trasladar(a,b);
    }
    public double perimetro() {
        Punto ant = vert[vert.length-1];
        double res = 0;
        for (Punto pto : vert) {
            res += pto.distancia(ant);
            ant = pto;
        }
        return res;
    }
    public String toString() {
        return Arrays.toString(vert);
    }
    abstract public double area();
}
```

La clase Poligono es abstracta porque...

... podemos dar una definición abstracta del comportamiento esperado de un polígono, esto es, podemos decir que todo polígono debe tener un método perímetro(...), un método trasladar(...) y un método área(...) pero...

...no podemos definir un algoritmo que sirva para calcular el área de cualquier polígono, o sea, no es posible dar una implementación concreta del área...

...de modo que indicamos que el método area(...) es abstracto y lo dejamos sin implementar.

Clases abstractas

```
public class Cuadrado extends Poligono {
      public Cuadrado(Punto[] vs) {
            super(vs);
      }
      public double area() {
            return lado() * lado();
      }
      public double lado() {
        return vert[0].distancia(vert[1]);
       }
}

public class Rectangulo extends Poligono {
      public Rectangulo(Punto[] vs) {
            super(vs);
      }
      public double base() {
            return vert[0].distancia(vert[1]);
      }
      public double altura() {
            return vert[1].distancia(vert[2]);
      }
      public double area() {
            return base() * altura();
      }
}
```

La clase Cuadrado hereda de la clase abstracta Poligono, luego está obligada a proporcionar una implementación del método area(). Si no lo hiciese, ella misma debería ser declarada como abstracta.

El lado del cuadrado se calcula como la distancia entre los dos primeros vértices (recuerde que los vértices son de tipo Punto y que distancia(...) es un método de Punto)

Igualmente en Rectangulo, heredera de Poligono, se proporciona una implementación del método area()

Clases abstractas

```
3  public class EjemploUsoPoligono {
4      public static void main(String[] args) {
5          Punto[] v = {new Punto(0,1), new Punto(1,1), new Punto(1,0), new Punto(0,0)};
6          Poligono p = new Poligono(v);
```

Multiple markers at this line
- TODO Auto-generated method stub
- Cannot instantiate the type Poligono

Al intentar crear un objeto de la clase Poligono, el compilador da un error ya que Poligono es una clase abstracta y por lo tanto no se puede instanciar.

```
3  public class EjemploUsoPoligono {
4      public static void main(String[] args) {
5          Punto[] v = {new Punto(0,1), new Punto(1,1), new Punto(1,0), new Punto(0,0)};
6          Poligono c = new Cuadrado(v);
7          double a = c.area();
8          System.out.println("área: " + a);
9      }
10 }
```

Sin embargo, sí se pueden crear instancias de sus subclases.

Console | Progress

<terminated> EjemploUsoPoligc

área: 1.0