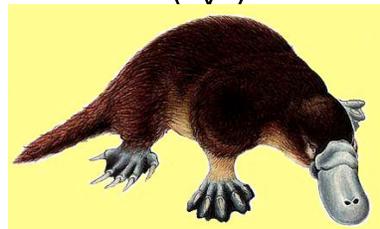
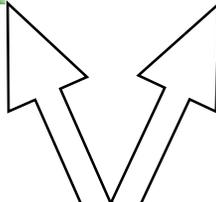


Herencia

Patricio.Inostroza@dcc.uchile.cl

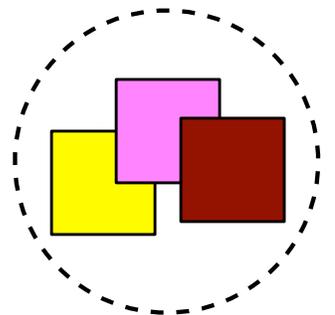
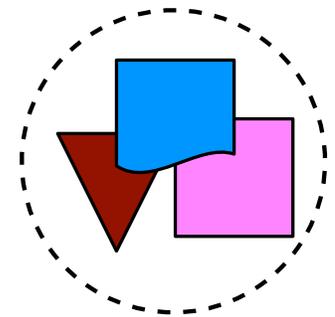
Herencia

- Herencia en un matrimonio joven...
 - Él: Me prometes que no engordarás como tu madre.
 - Ella: Por supuesto! Si tu me prometes que no quedarás calvo como tu padre.



Herencia

- Sea A un objeto con características generales
 - Ej: Un polígono tiene una posición, un color, etc.
- Sea B un objeto que es una especialización de A, i.e. tiene las mismas características de A y otras propias
 - Ej: Un rectángulo es un polígono que tiene 4 lados



Herencia

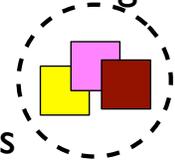
- Importante:
 - El objeto especializado tiene todas las propiedades de objeto genérico
 - El objeto genérico **NO** tiene todas las propiedades del objeto especializado

Polígono



Un rectángulo **tiene** un color

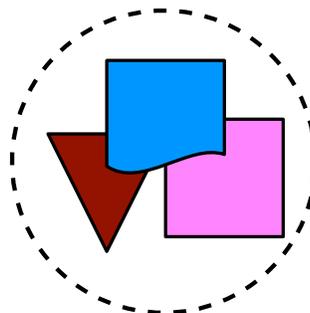
Rectángulo



Un polígono **NO tiene** (siempre) 4 lados

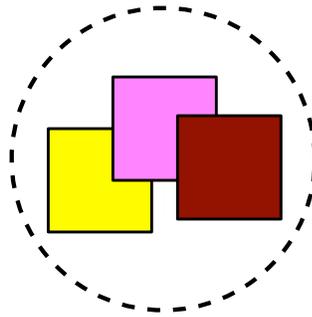
Herencia

- Superclase: El objeto con características generales
- Clase padre: otra forma de decir Superclase
 - Ej: Polígono



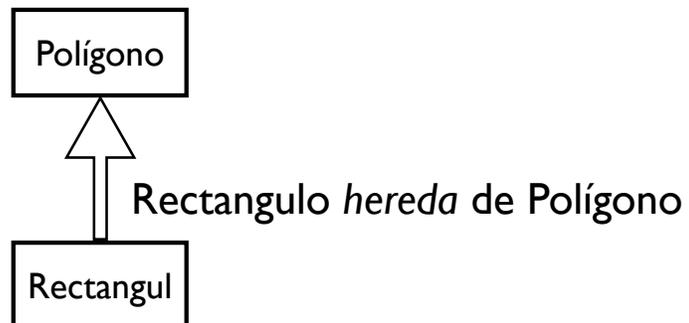
Herencia

- Subclase: Clase especializada
- Clase hija: otras forma de decir subclase
 - Ej: Rectángulo



Herencia

- Notación



```
clase Poligono {
    int posx, posy;    // posicion del poligono

    public Poligono(int x, int y) {
        posx = x; posy = y;
    }

    public int getPositionX() {
        return posx;
    }

    public int getPositionY() {
        return posy;
    }
}
```

Polígono

```
clase Rectangulo extends Poligono {
    int ancho, largo;    // ancho y largo

    public Rectangulo(int x, int y, int a, int l) {
        posx = x;
        posy = y;
        ancho = a; largo = l;
    }

    public int getArea() {
        return ancho * largo;
    }
}
```

Polígono



Rectangulo

```

class Rectangulo extends Poligono {
    int ancho, largo;    // ancho y largo

    public Rectangulo(int x, int y, int a, int l) {
        posX = x; }
        posY = y; }
        ancho = a; largo = l;
    }

    public int getArea() {
        return ancho * largo;
    }
}

```

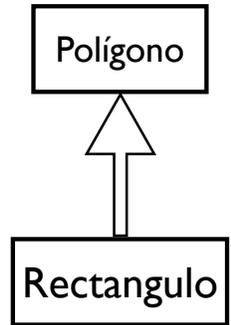
Está inicializado en la clase padre

```

class Poligono {
    int posX, posY;

    public Poligono(int x, int y) {
        posX = x; posY = y;
    }
    // ...
}

```



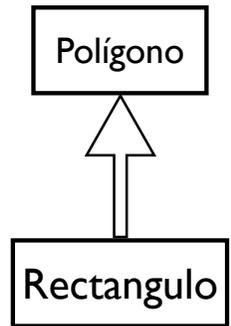
```

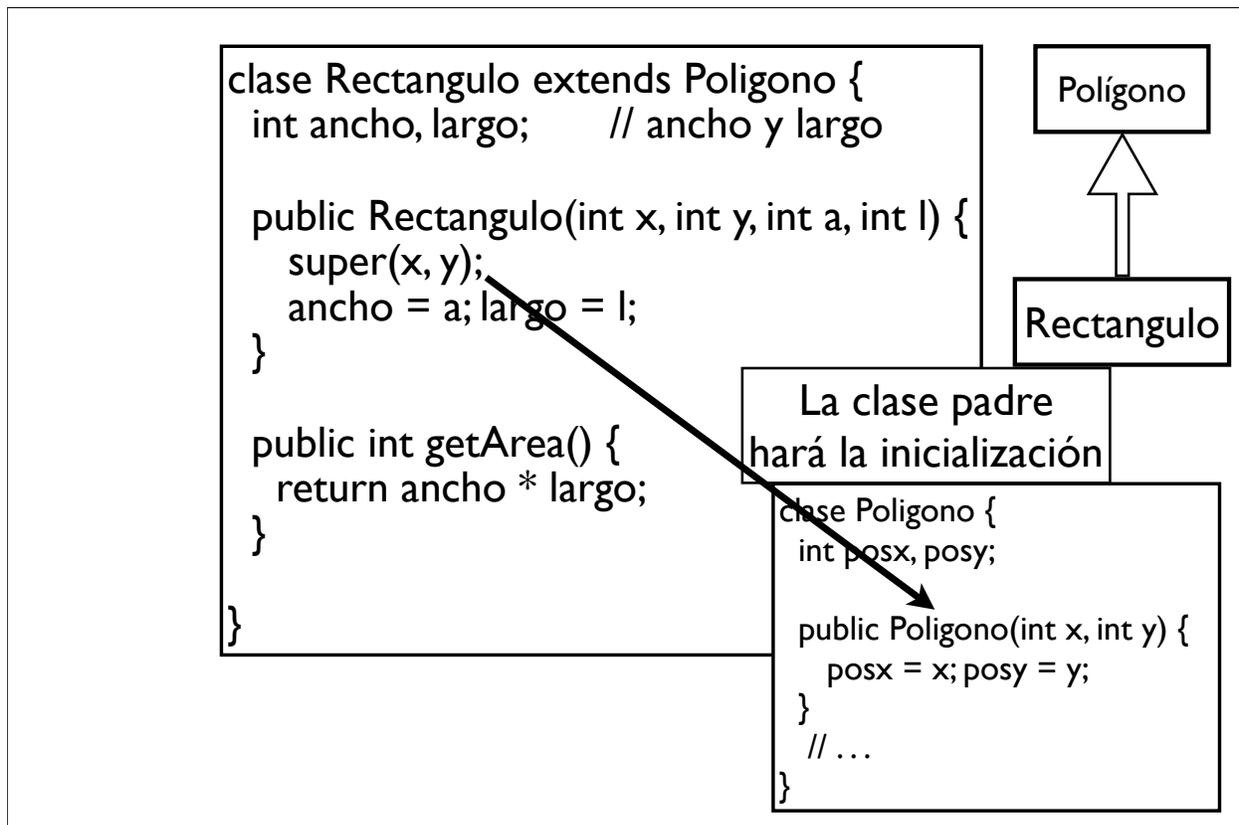
class Rectangulo extends Poligono {
    int ancho, largo;    // ancho y largo

    public Rectangulo(int x, int y, int a, int l) {
        super(x, y);
        ancho = a; largo = l;
    }

    public int getArea() {
        return ancho * largo;
    }
}

```





```

class Programa {

    static public void main(String argv) {
        Poligono p = new Poligono(2, 2);
        Rectangulo r = new Rectangulo(4, 4, 3, 2);

        System.out.println("Area= " + r.getArea());
        System.out.println("Posición en X= " + p.getPositionX());
        System.out.println("Posición en X= " + r.getPositionX());
    }
}

class Poligono {
    int posx, posy;
    public Poligono(int x, int y) { ... }
    public int getPositionX() { ... }
    public int getPositionY() { ... }
}

class Rectangulo extends Poligono {
    int ancho, largo;
    public Rectangulo(int x, int y, int a, int l) { ... }
    public int getArea() { ... }
}

```

```

class Programa {

    static public void main(String argv) {
        Poligono p = new Poligono(2, 2);
        Rectangulo r = new Rectangulo(4, 4, 3, 2);

        System.out.println("Area= " + r.getArea());
        System.out.println("Posición en Y= " + r.getPositionY());
        System.out.println("Posición en X= " + r.getPositionX());
    }
}

```

```

class Poligono {
    int posx, posy;
    public Poligono(int x, int y) { ... }
    public int getPositionX() { ... }
    public int getPositionY() { ... }
}

```

```

class Rectangulo extends Poligono {
    int ancho, largo;
    public Rectangulo(int x, int y, int a, int l) { ... }
    public int getArea() { ... }
}

```

```

class Programa {

    static public void main(String argv) {
        Poligono p = new Poligono(2, 2);
        Rectangulo r = new Rectangulo(4, 4, 3, 2);

        System.out.println("Area= " + r.getArea());
        System.out.println("Posición en X= " + p.getPositionX());
        System.out.println("Posición en X= " + r.getPositionX());
    }
}

```

```

class Poligono {
    int posx, posy;
    public Poligono(int x, int y) { ... }
    public int getPositionX() { ... }
    public int getPositionY() { ... }
}

```

```

class Rectangulo extends Poligono {
    int ancho, largo;
    public Rectangulo(int x, int y, int a, int l) { ... }
    public int getArea() { ... }
}

```

```

class Programa {

    static public void main(String argv) {
        Poligono p = new Poligono(2, 2);
        Rectangulo r = new Rectangulo(4, 4, 3, 2);

        System.out.println("Area= " + r.getArea());
        System.out.println("Posición en X= " + r.getPositionX());
        System.out.println("Posición en X= " + r.getPositionX());
    }
}

```

```

class Poligono {
    int posx, posy;
    public Poligono(int x, int y) { ... }
    public int getPositionX() { ... }
    public int getPositionY() { ... }
}

```

```

class Rectangulo extends Poligono {
    int ancho, largo;
    public Rectangulo(int ?, int a, int l) { ... }
    public int getArea() { ... }
}

```

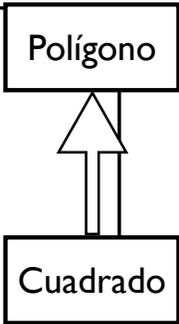
```

class Programa {

    static public void main(String argv) {
        Poligono p = new Poligono(2, 2);
        Rectangulo r = new Rectangulo(4, 4, 3, 2);

        System.out.println("Area= " + r.getArea());
        System.out.println("Posición en X= " + r.getPositionX());
        System.out.println("Posición en X= " + r.getPositionX());
    }
}

```



```

class Poligono {
    int posx, posy;
    public Poligono(int x, int y) { ... }
    public int getPositionX() { ... }
    public int getPositionY() { ... }
}

```

```

class Cuadrado extends Poligono {
    int ancho, largo;
    public Cuadrado(int x, int y, int a, int l) { ... }
    public int getArea() { ... }
}

```

Herencia

- Una clase especializada (hija) tiene todas las características de su clase padre
 - Ej: El Rectangulo heredó getPositionX()
- Una clase genérica (padre) NO tiene todas las características de una clase hija
 - Ej: El Poligono NO tiene getArea()

Herencia

```
class Poligono {  
    int posx, posy;  
    public Poligono(int x, int y) { ... }  
    public int getPositionX() { ... }  
    public int getPositionY() { ... }  
}
```

```
class Rectangulo extends Poligono {  
    int ancho, largo;  
    public Rectangulo(int x, int y, int a, int l) { ... }  
    public int getArea() { ... }  
}
```

Herencia

```
class Poligono {  
    int posx, posy;  
    public Poligono(int x, int y) { ... }  
    public int getPositionX() { ... }  
    public int getPositionY() { ... }
```

```
class Rectangulo extends Poligono {  
    int ancho, largo;  
    public Rectangulo(int x, int y, int a, int l) { ... }  
    public int getArea() { ... }  
  
    int posx, posy;  
    public Poligono(int x, int y) { ... }  
    public int getPositionX() { ... }  
    public int getPositionY() { ... }  
}
```

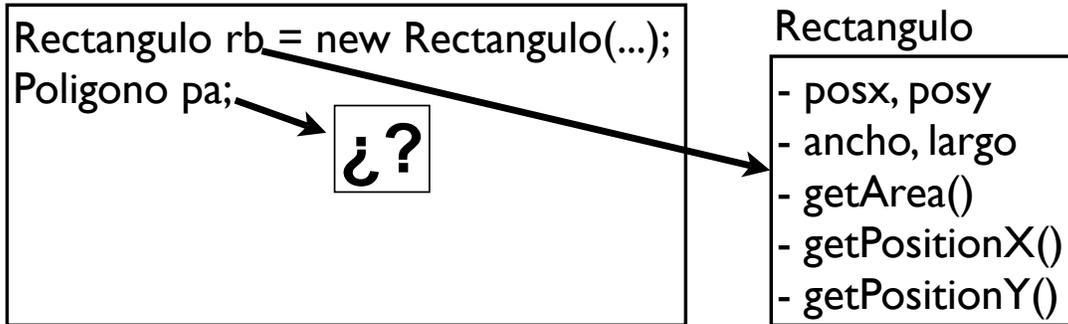
Herencia

- Se puede asignar un hijo a una clase padre, lo inverso NO es correcto

```
Poligono pa;  
Poligono pb = new Poligono(2, 2);  
Rectangulo ra;  
Rectangulo rb = new Rectangulo(1, 1, 4, 5);  
  
pa = rb; // padre <= hijo OK  
ra = pb // hijo <= padre ERROR
```

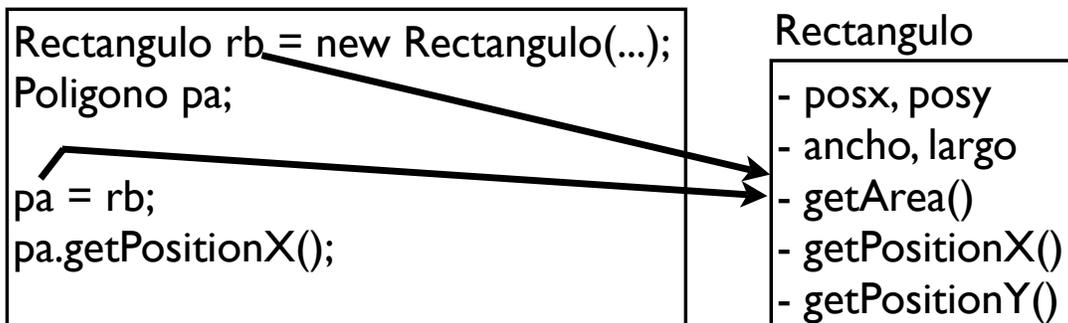
Herencia

- Se puede asignar un hijo a una clase padre, lo inverso NO es correcto



Herencia

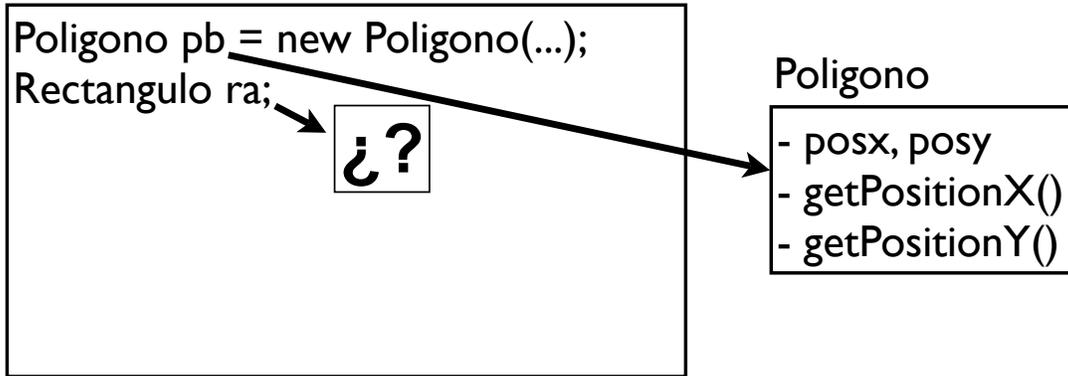
- Se puede asignar un hijo a una clase padre, lo inverso NO es correcto



Ojo: pa.getArea() no es valido, ya que no está definido para la Clase Poligono

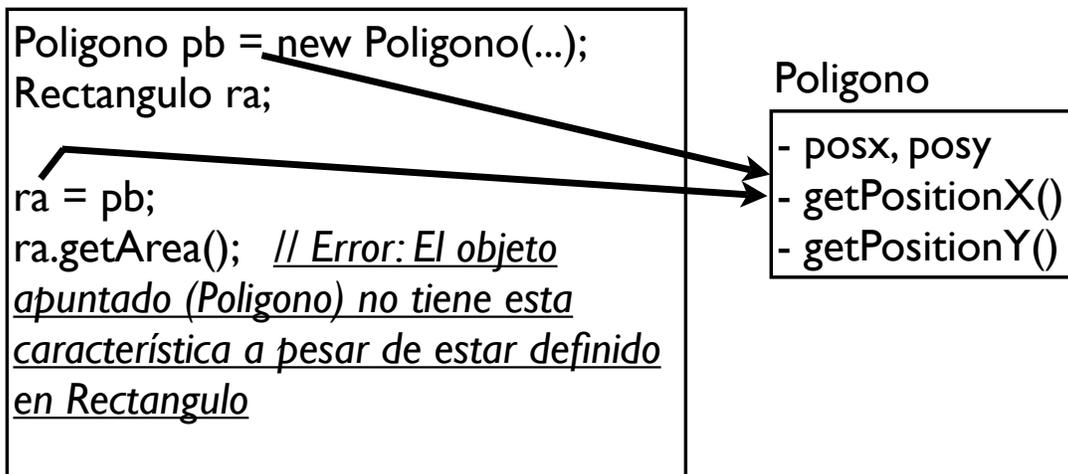
Herencia

- Se puede asignar un hijo a una clase padre, lo inverso NO es correcto



Herencia

- Se puede asignar un hijo a una clase padre, lo inverso NO es correcto



Clase abstracta

- Hay funcionalidades que una clase delega y obliga implementar a sus hijo
- Este tipo de clases se denominan abstractas
 - Ej: Un polígono deja a sus hijos el cálculo del perímetro
 - `public abstract int getPerimetro()`
- Una clase abstracta no puede ser instanciada (no se podrá hacer “new Poligono”)

```
abstract class Poligono {  
    int posx, posy;    // posicion del poligono  
  
    public Poligono(int x, int y) { posx = x; posy = y; }  
  
    public int getPositionX() { return posx; }  
  
    public int getPositionY() { return posy; }  
  
    public abstract int getPerimetro() ;  
}
```

Polígono

`getPerimetro` sólo está definida, pero no tiene implementación, esto deberán hacerlo la clase hija (por ejemplo, Rectangulo)

```
abstract class Poligono {
    int posX, posY;    // posicion del poligono
    public Poligono(int x, int y) { posX = x; posY = y; }
    public int getPositionX() { return posX; }
    public int getPositionY() { return posY; }

    public abstract int getPerimetro() ;
}
```

```
class Rectangulo extends Poligono {
    int ancho, largo;    // ancho y largo
    public Rectangulo(int x, int y, int a, int l) {
        super(x, y); ancho = a; largo = l; }
    public int getArea() {return ancho * largo; }

    public int getPerimetro() { return 2 * (ancho + largo); }
}
```

Interface

- Interface: Clase que define métodos pero no tienen ninguna implementación

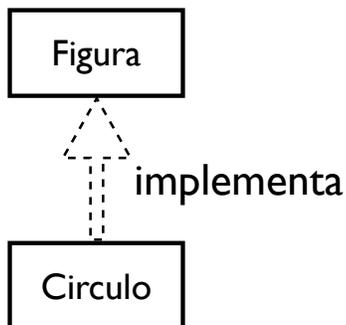
```
interface Figura {

    public int getArea();
    public int getPerimetro();
}
```

- Un interfaz NO puede ser instanciada

Interface

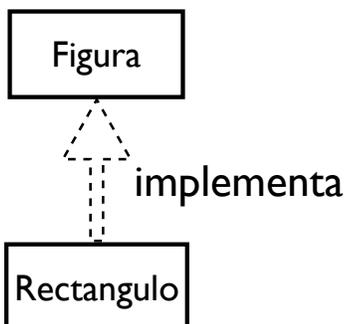
```
interface Figura {  
    public int getArea();  
    public int getPerimetro();  
}
```



```
class Circulo implements Figura {  
    int radio, cx, cy;  
  
    public Circulo(int x, int y, int r) {  
        cx = x; cy = y; radio = r;  
    }  
  
    public int getArea() {  
        return Math.PI * radio * radio;  
    }  
  
    public int getPerimetro() {  
        return 2 * Math.PI * radio;  
    }  
}
```

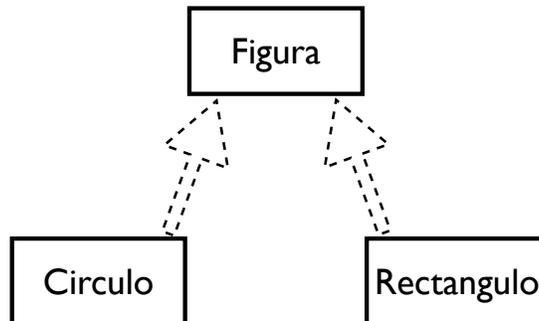
Interface

```
interface Figura {  
    public int getArea();  
    public int getPerimetro();  
}
```



```
class Rectangulo implements Figura {  
    int ancho, largo;  
  
    public Rectangulo(int a, int l) {  
        ancho = a; largo = l;  
    }  
  
    public int getArea() {  
        return ancho * largo;  
    }  
  
    public int getPerimetro() {  
        return 2 * (ancho + largo);  
    }  
}
```

Interface



Las interfaces permiten desentenderse de la implementación y del objeto

Interface

- Figura es una interfaz; no puede ser instanciada: `new Figura(...)` es un error!!

```
clase Programa {
    // ...

    static public int sumarAreas(Figura f1, Figura f2) {
        return f1.getArea() + f2.getArea();
    }
}
```

- Pero, este programa es válido!
¿ Cómo puede ser llamado el método “sumarAreas” con dos objetos de tipo Figura ?

```

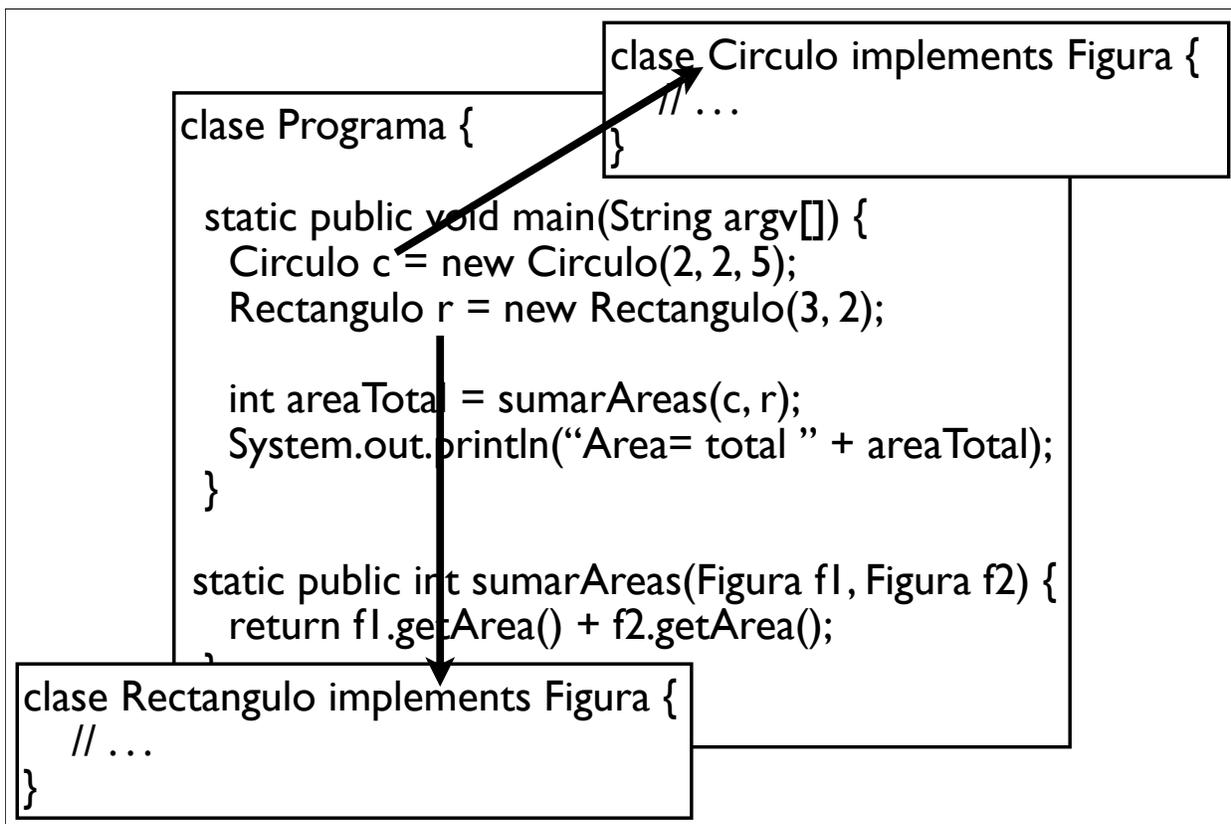
class Programa {

    static public void main(String argv[]) {
        Circulo c = new Circulo(2, 2, 5);
        Rectangulo r = new Rectangulo(3, 2);

        int areaTotal = sumarAreas(c, r);
        System.out.println("Area= total " + areaTotal);
    }

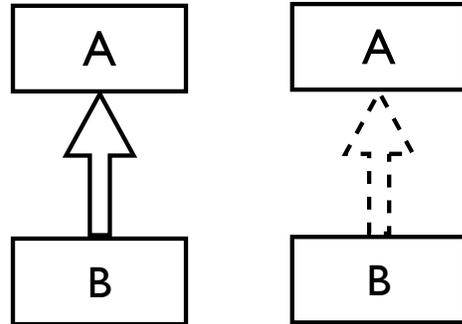
    static public int sumarAreas(Figura f1, Figura f2) {
        return f1.getArea() + f2.getArea();
    }
}

```



Herencia

- Sean A y B clases
- A es padre de B
- A a;
- B b;
- a es de tipo A
- b es de tipo B
- b es de tipo A !!



Esto también es válido cuando se trata de clases abstractas o de interfaces

Cuidado!!

- Similitud no necesariamente implica herencia
- No hay que forzar la herencia
- Usar herencia para “aprovechar” el código escrito en un par de métodos, en general es una mala práctica!!

