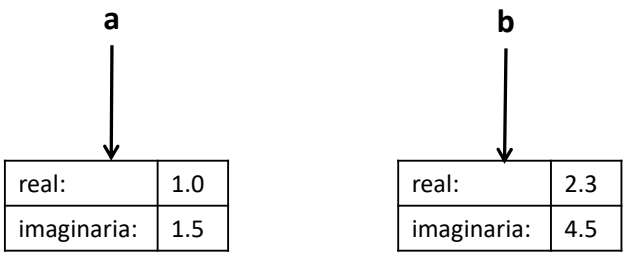
Modificadores *static* y *final*

Programación Orientada a Objetos

Atributos de instancia

- Son los atributos que hemos estudiado hasta ahora.
- Se definen dentro de la clase.
- Todos los objetos de esa clase tienen los mismos atributos, pero sus valores son **diferentes** en cada instancia.

```
Complex a = new Complex(1.0,1.5);
Complex b = new Complex(2.3,4.5);
```



Atributos de clase (o *static*)

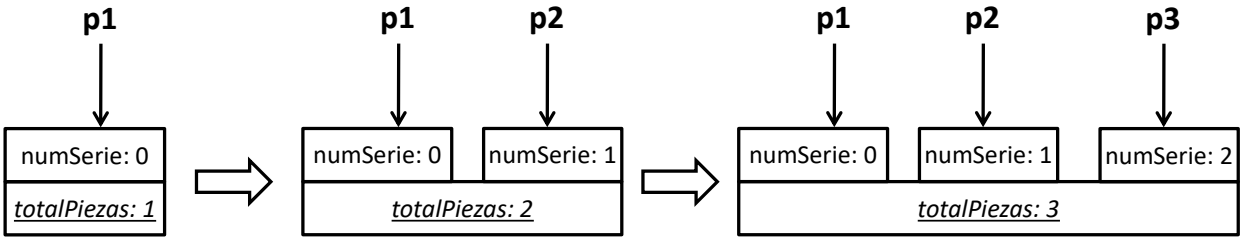
- El valor de un atributo estático no está asociado a una instancia (objeto) sino a una clase
- Se declara poniendo el modificador `static` entre el modificador de visibilidad (`public` / `protected` / `private`) y el tipo del atributo.
  - Ejemplos:
    ```
    protected static String variable;
    private static int numero;
    ```
- Manera de acceder: **NombreClase**.nombreAtributoEstático
  - (Nótese que para los atributos de instancia es: nombreObjeto.nombreAtributo)
- No es necesario instanciar ningún objeto para acceder a un atributo estático.

Ejemplo (1)

```
public class PiezaManufacturada {
  private int numSerie;

  protected static int totalPiezas= 0;

  public PiezaManufacturada() {
    numSerie = PiezaManufacturada.totalPiezas;
    PiezaManufacturada.totalPiezas++;
  }

  public int getNumeroDeSerie() {
    return numSerie;
  }
}
```

Ejemplo (2)

```
public class CadenaDeProduccion {
  public void producir() {
    PiezaManufacturada p1 = new PiezaManufacturada();
    PiezaManufacturada p2 = new PiezaManufacturada();
    System.out.println("Producción inicial: "
                + PiezaManufacturada.totalPiezas + " piezas");
    PiezaManufacturada p3 = new PiezaManufacturada();
    System.out.println("Producción total: "
                + PiezaManufacturada.totalPiezas + " piezas");
  }
}
```



Métodos estáticos

- Hay métodos (funciones) que no acceden a los atributos de la clase en la que están.
  - No tiene sentido tener que instanciar ningún objeto para llamar a dicho método.

- Solución: definir métodos como estáticos.
  - Se definen con el modificador static. Ej:
    `public static int metodoEstatico();`
  - Se invocan mediante el nombre de la clase:
    `int r = NombreClase.metodoEstatico();`

- Ejemplo: `double y = Math.sin(x);`
  - Java proporciona una clase llamada "Math" que proporciona decenas de funciones matemáticas sin necesidad de instanciar la clase.

Modificador final

- El valor de un atributo o variable marcados con el modificador *final* antes de su tipo solo puede ser asignado una vez (su valor no podrá cambiar nunca más).
  - El valor se le debe asignar en la propia definición del atributo, o en el constructor de la clase.

```
public class UnaClase {
  public final int valorInmutable;
  public UnaClase(int vInicial) {
    valorInmutable = vInicial;
  }
}
```

Modificador final

- Parámetro `final`: no puede modificarse dentro del cuerpo de la función.
  - `public void funcion(final int param);`
- Variable `final`: su valor sólo puede asignarse una vez.
- Un método `final` impide que una clase derivada modifique su comportamiento (esto se aclarará cuando se vea herencia).
- Ojo con esto!
- `final Complex c = new Complex(1.2, 2.5);`
  Esto quiere decir que la referencia "c" no podrá apuntar a ningún otro objeto; pero sí podemos modificar los atributos del objeto.

Combinando static y final

- Las **constantes** en Java se definen mediante la combinación de static y final.
  -
```
public class Constantes {
    public static final float PI = 3.14159265;
    public static final float E = 2.71828183;
    public static final float G = 6.67428e-11;
}
```
- Algunos métodos también pueden ser static y final:

```
public static final void main(String[] args) {
   //punto de acceso al programa
}
```