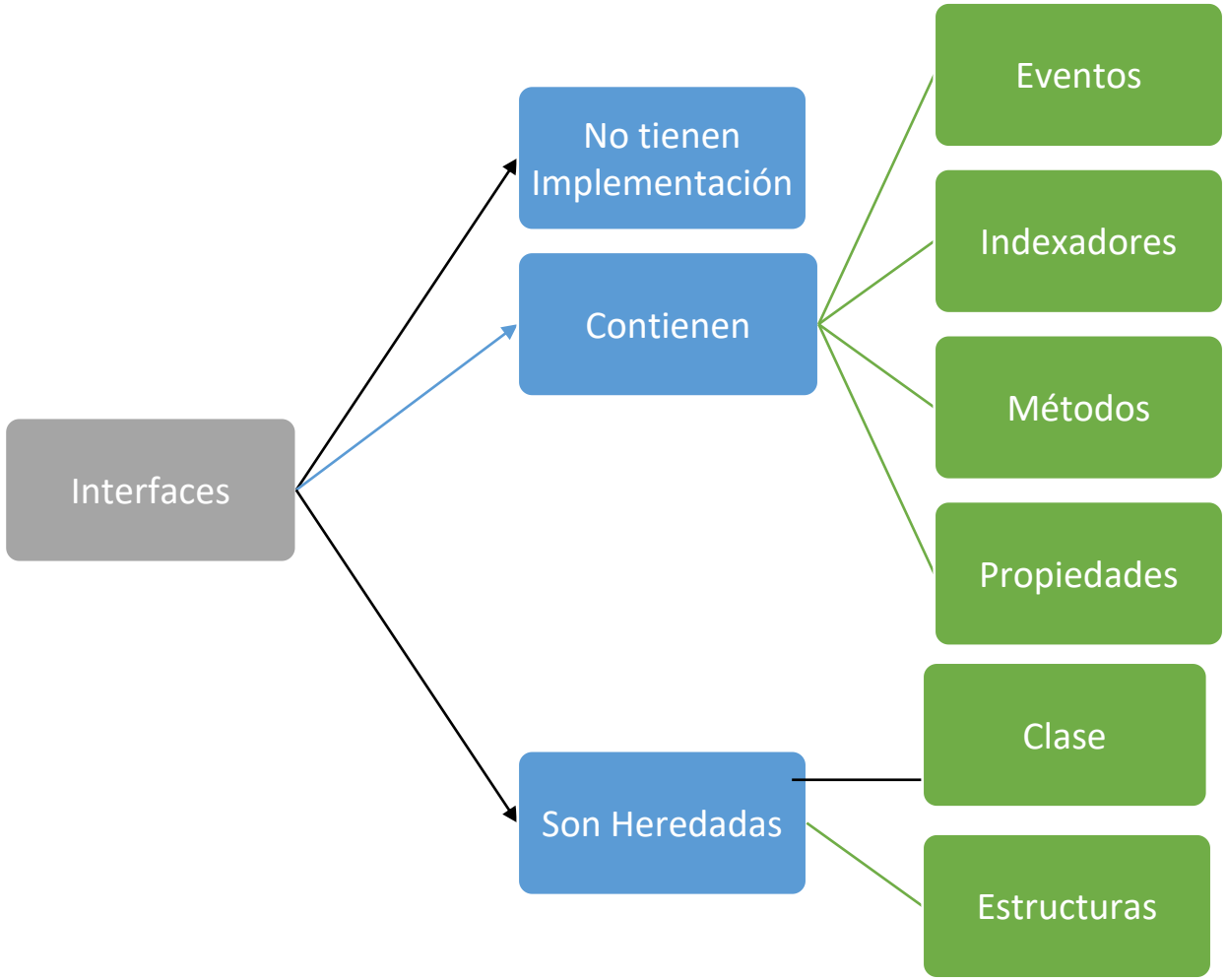
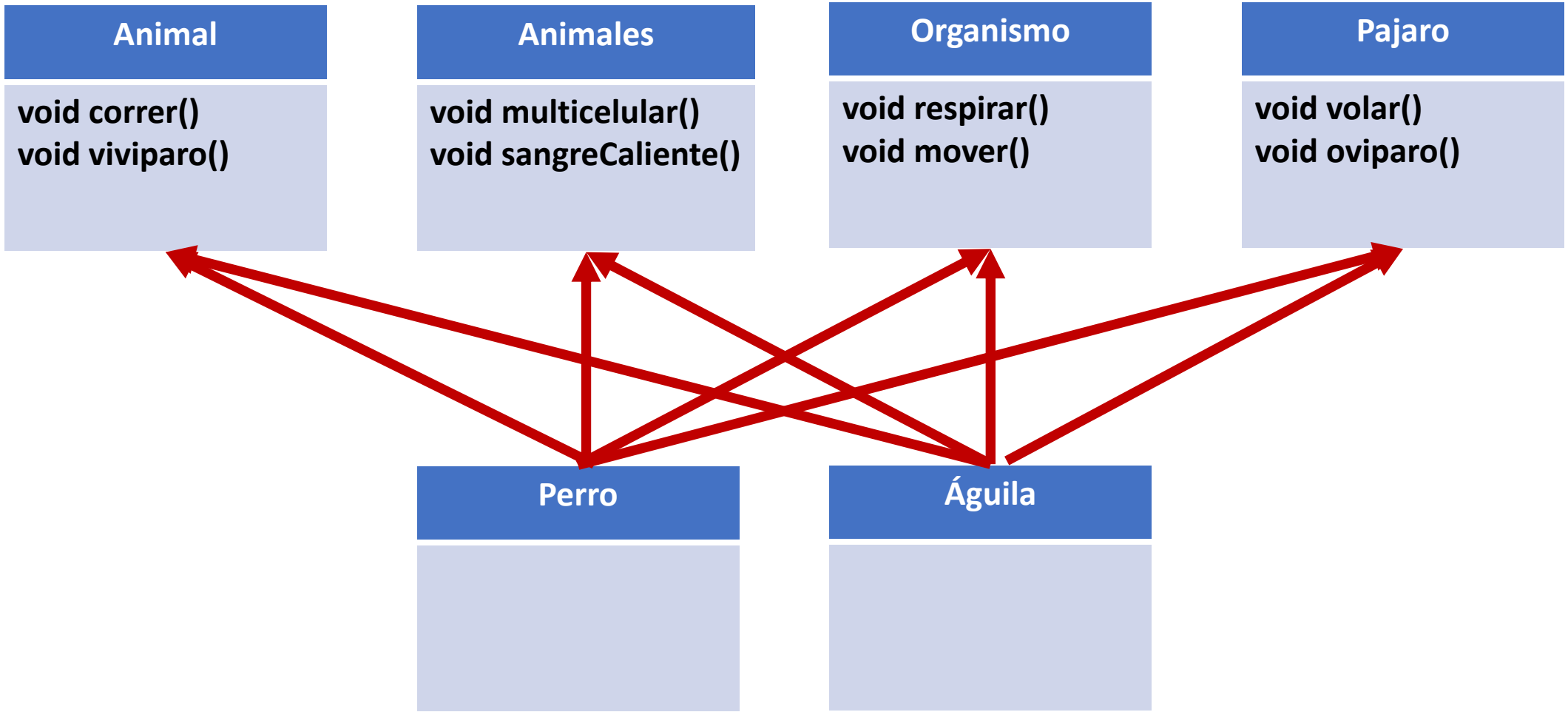
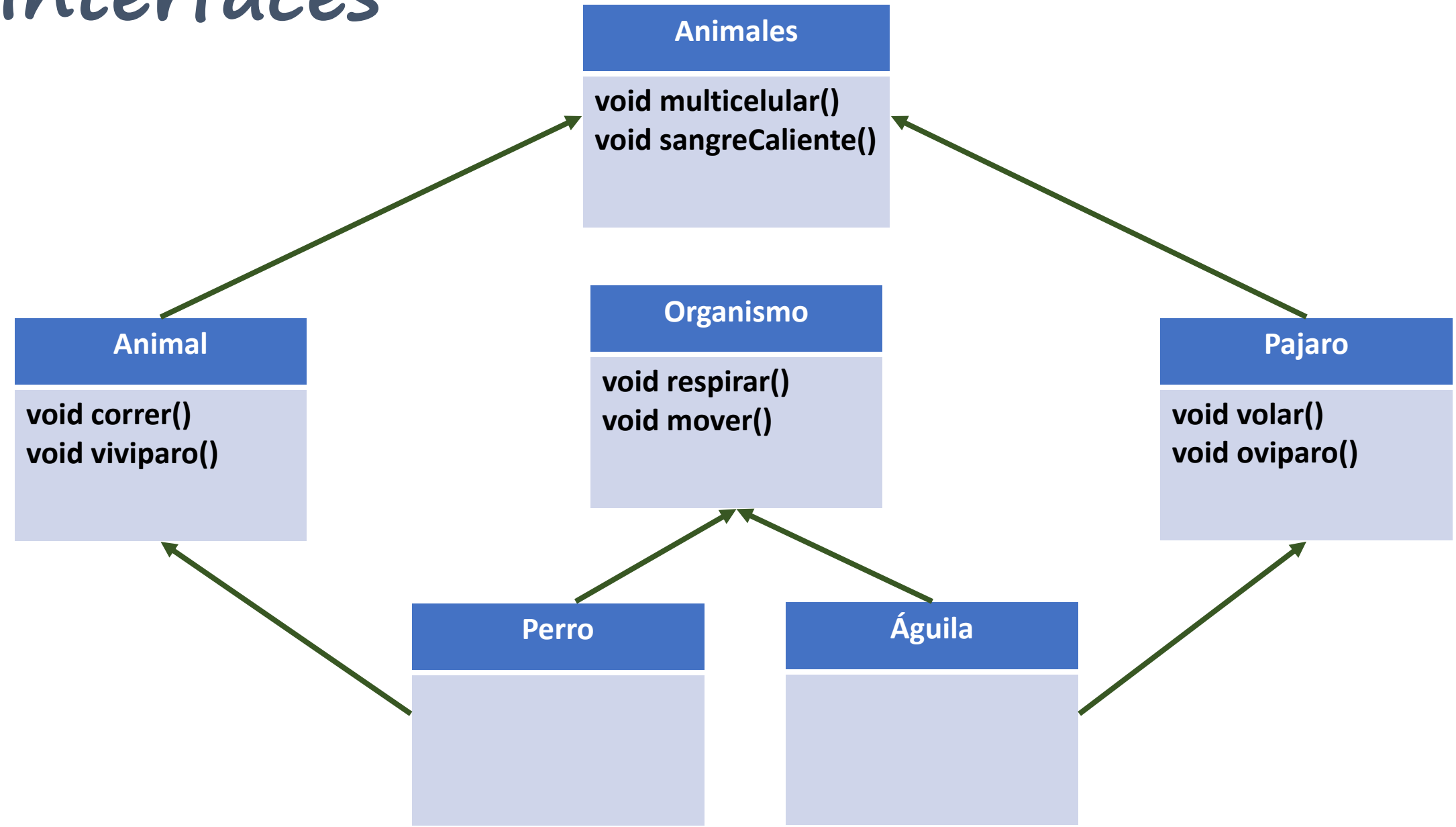
Interfaces


Interfaces
No tienen Implementación
Contienen
Son Heredadas
Eventos
Indexadores
Métodos
Propiedades
Clase
Estructuras

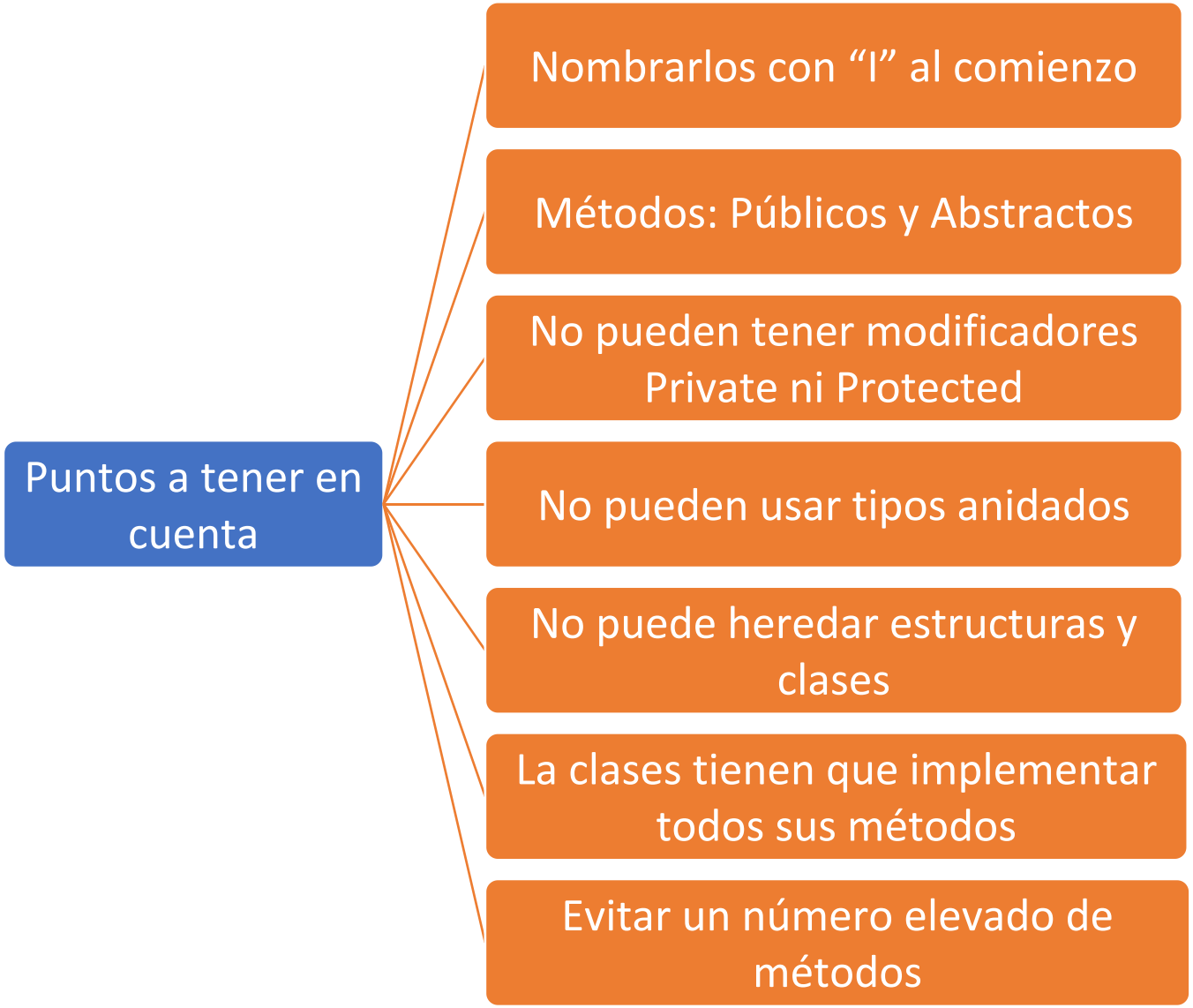
Interfaces


Animal
void correr()
void viviparo()
Animales
void multicelular()
void sangreCaliente()
Organismo
void respirar()
void mover()
Pajaro
void volar()
void oviparo()
Perro
Águila

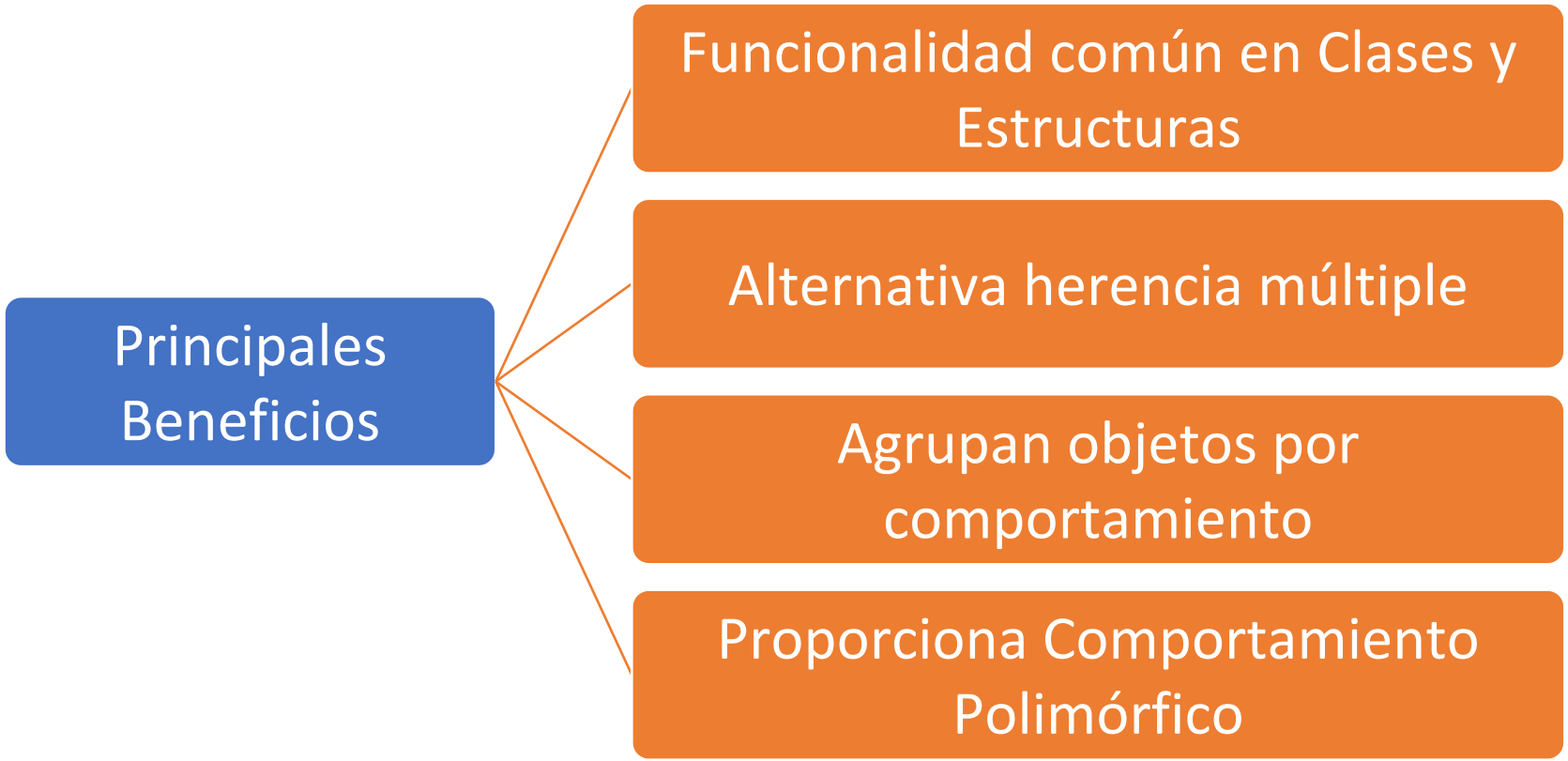
Interfaces


Animales
void multicelular()
void sangreCaliente()
Animal
void correr()
void viviparo()
Organismo
void respirar()
void mover()
Pajaro
void volar()
void oviparo()
Perro
Águila

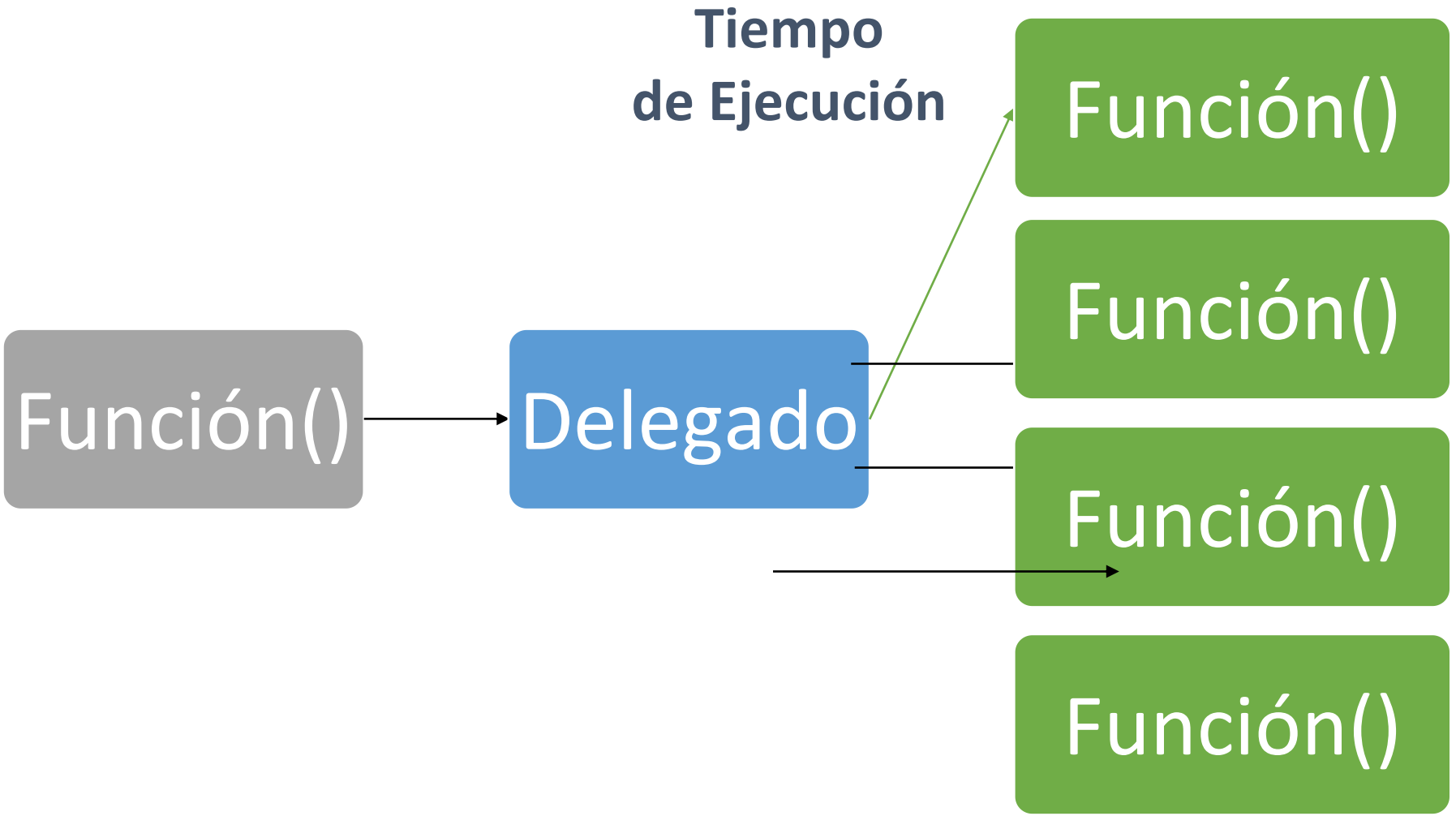
Interfaces


Puntos a tener en cuenta
Nombrarlos con “I” al comienzo
Métodos: Públicos y Abstractos
No pueden tener modificadores Private ni Protected
No pueden usar tipos anidados
No puede heredar estructuras y clases
La clases tienen que implementar todos sus métodos
Evitar un número elevado de métodos

Interfaces


Principales Beneficios
Funcionalidad común en Clases y Estructuras
Alternativa herencia múltiple
Agrupan objetos por comportamiento
Proporciona Comportamiento Polimórfico

Delegados


Tiempo
de Ejecución
Función()
Delegado
Función()
Función()
Función()
Función()

Delegados

Ejemplo de Delegado

```
public delegate int MiDelegado (string s);
```

Declaración de Delegado

```
delegate <valor retorno> <nombre delegado> <lista parámetros>
```

Ejemplo Inicialización Delegado

```
public delegate void imprimirCadena(string s);
...
imprimirCadena ic1 = new imprimirCadena(MetodoEscribirPantalla);
imprimirCadena ic2 = new imprimirCadena(MetodoEscribirArchivo);
```

Métodos Anónimos

Ejemplo de Método Anónimos

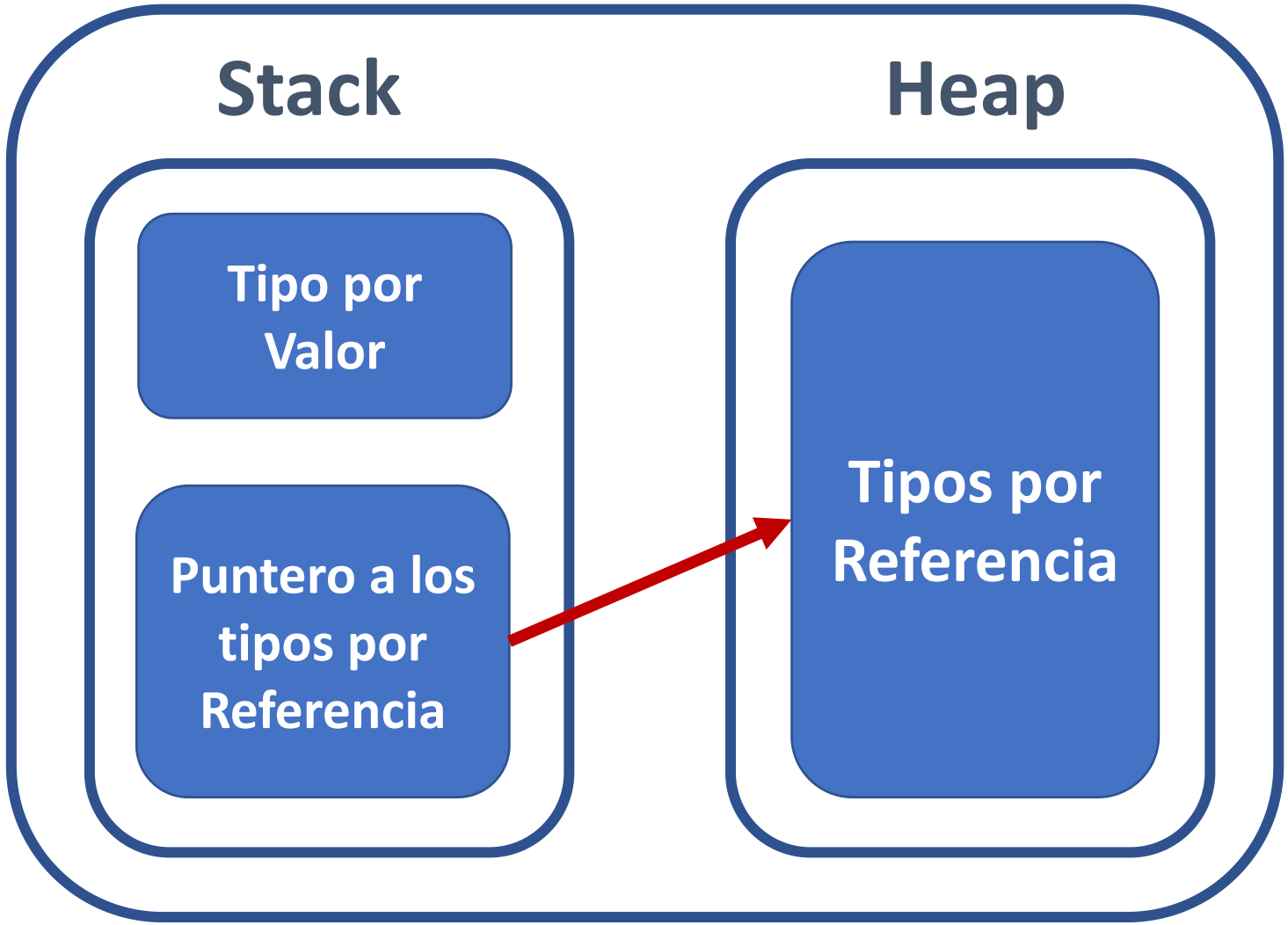
```
delegate void Numeros(int n);
...
Numeros num = delegate(int x) {
   Console.WriteLine("Método Anónimo: {0}", x);
};
```

Declaración de Delegado

```
delegate <valor retorno> <nombre delegado> (<lista parámetros>);
…
<nombre delegado> = delegate(<lista parámetros>)
        {
                Método Anónimo
        };
```

Estructuras

MEMORIA



Libros

+ Título
+ Autor
+ Categoría
+ id

Estructuras

Definición de Estructura

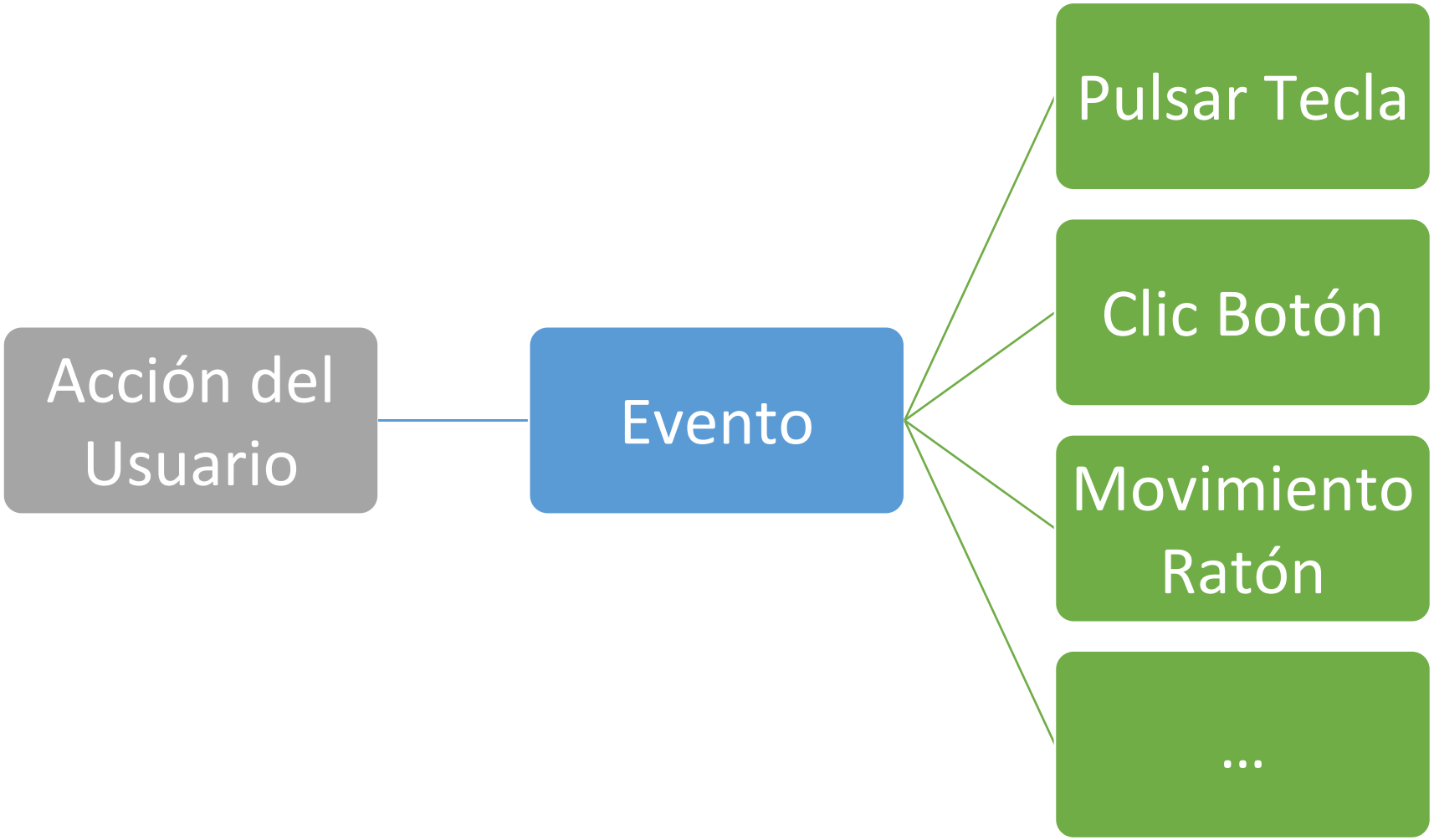
Palabra clave: struct

```
struct Libros {
   public string titulo;
   public string autor;
   public string categoria;
   public int libro_id;
};
```

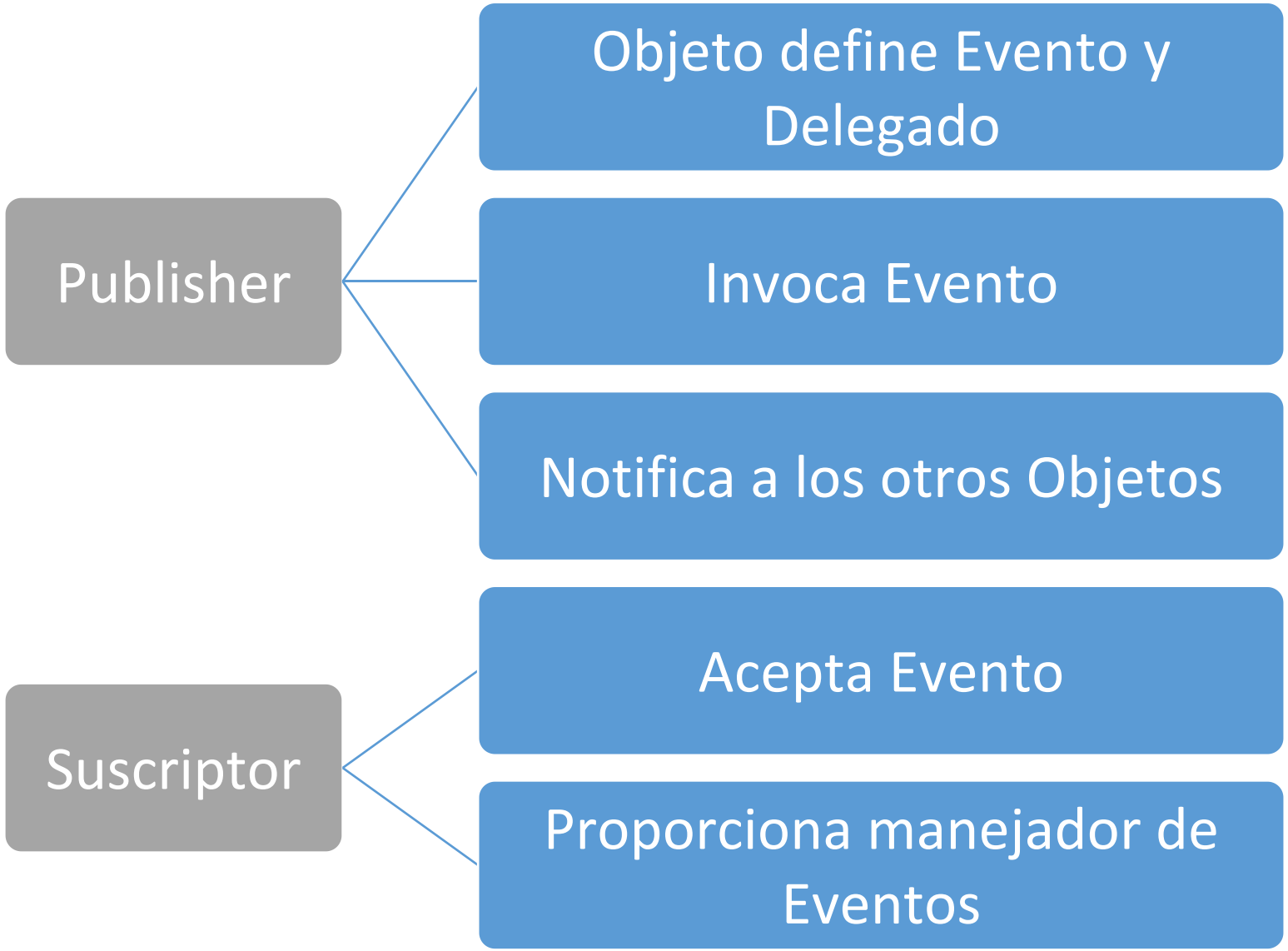
Estructuras - Características

- Pueden tener métodos, campos, indexadores, propiedades, métodos de operador y eventos
- Pueden tener constructores definidos, pero no destructores
- No pueden Heredar otras Estructuras o Clases
- Pueden Implementar Interfaces
- No Abstract, virtual o protected
- No es Obligatorio el uso del Operador New

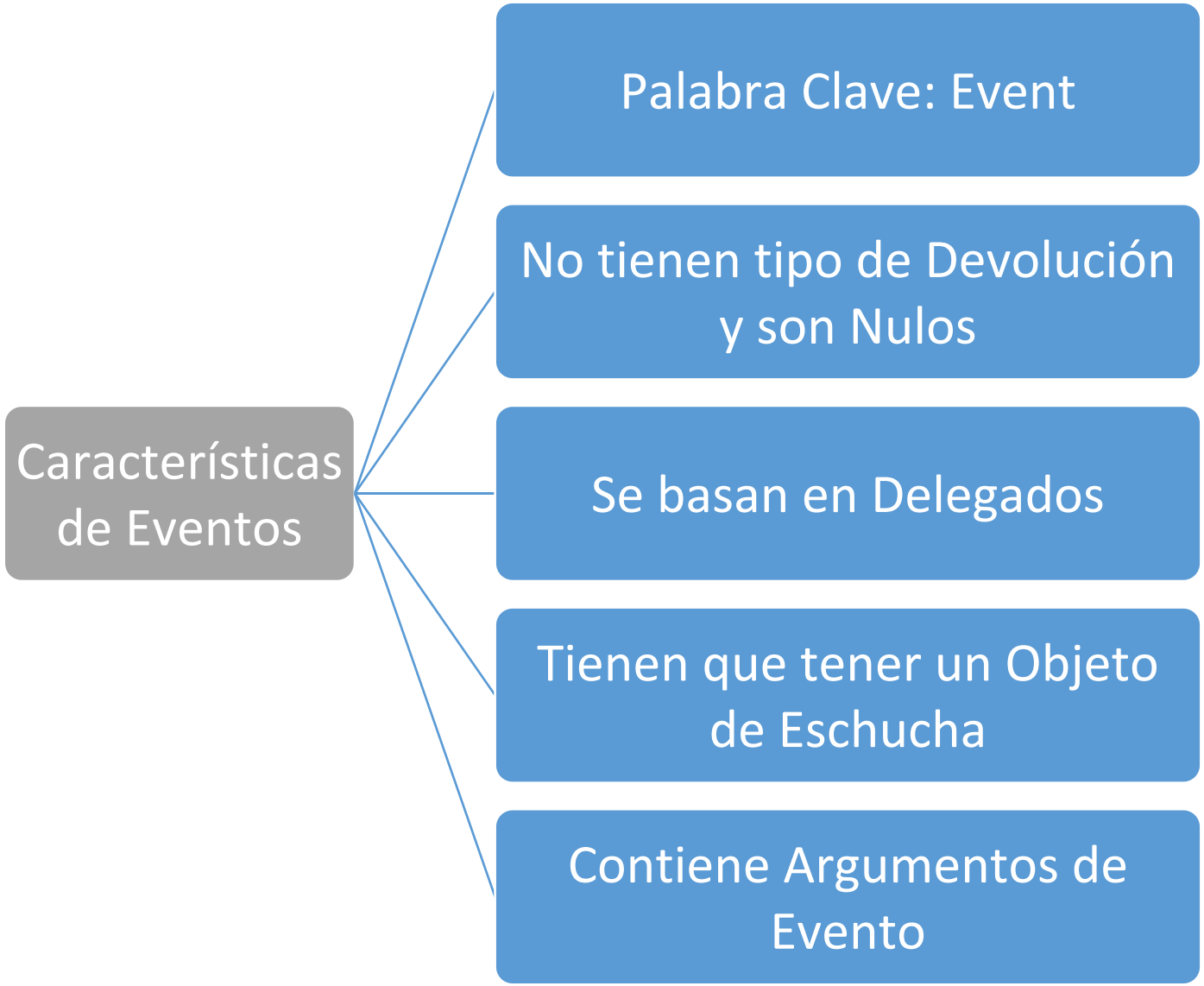
Eventos


Acción del Usuario
Evento
Pulsar Tecla
Clic Botón
Movimiento Ratón
...

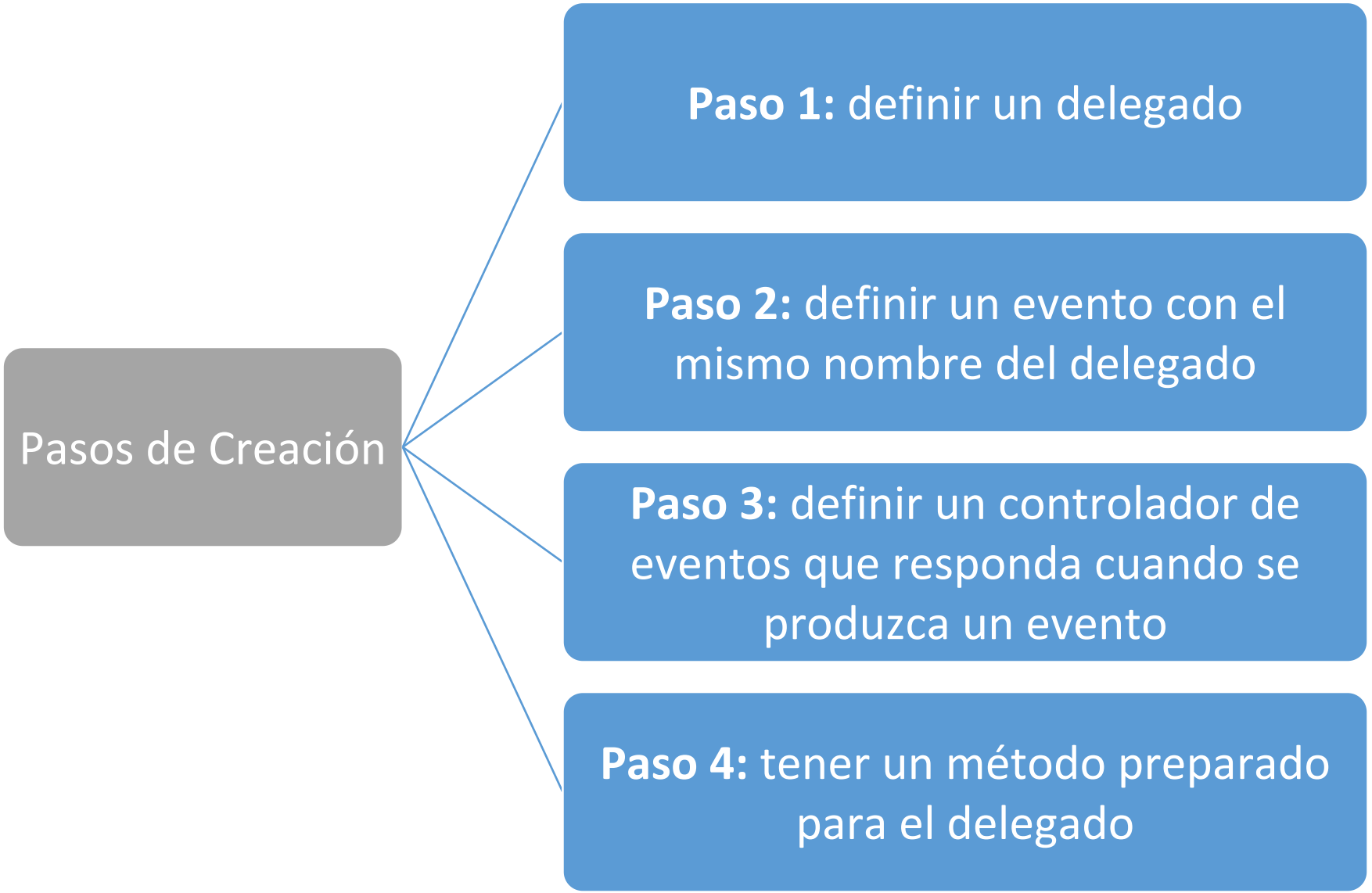
Eventos – Publisher - Suscriptor



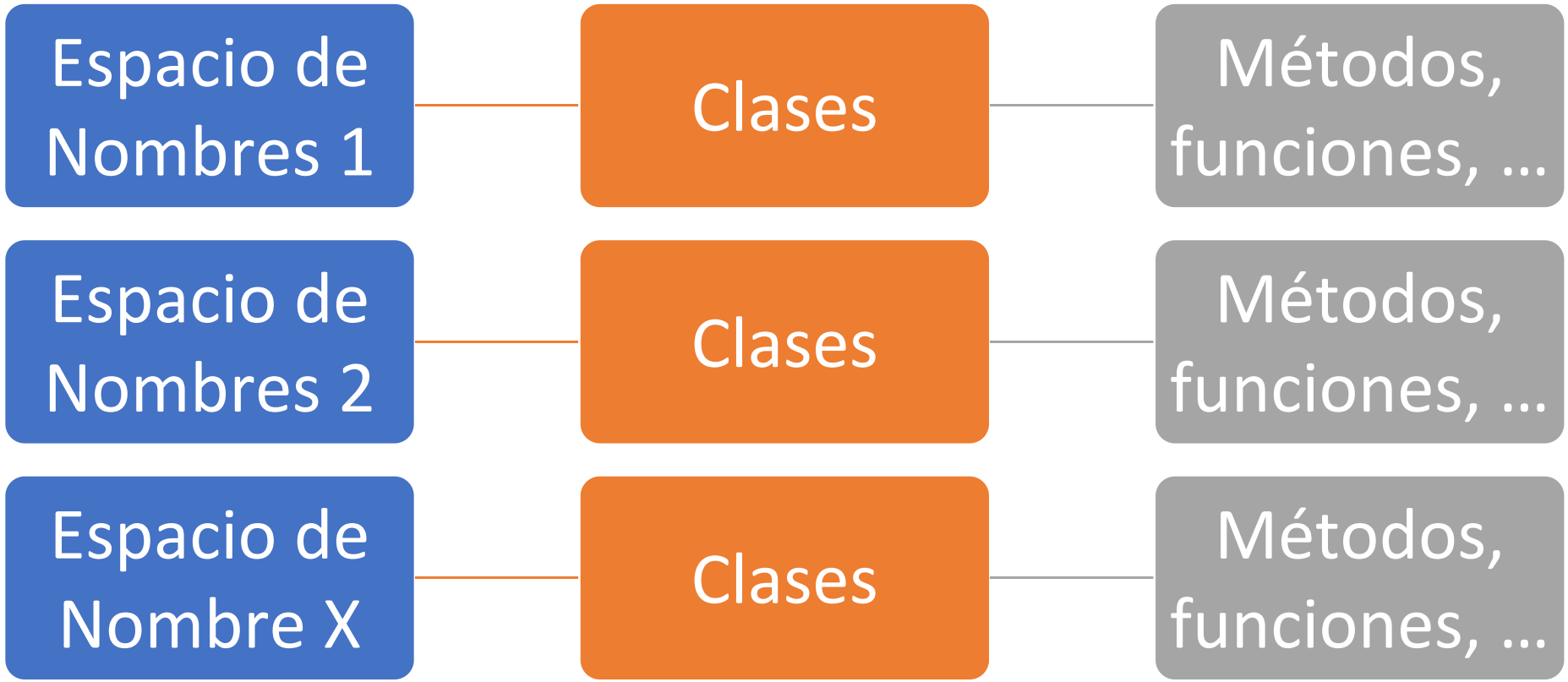
Eventos - Características



Eventos


Pasos de Creación
Paso 1: definir un delegado
Paso 2: definir un evento con el mismo nombre del delegado
Paso 3: definir un controlador de eventos que responda cuando se produzca un evento
Paso 4: tener un método preparado para el delegado

Espacios de Nombres



Espacios de Nombres

Declaración del Espacio de Nombres

```
namespace nombre_espacio_de_nombres {

// código fuente

}
```

```
namespace espacioNombres1 {
    class clase {
        public void funcion() {
            Console.WriteLine("Prueba Espacio de Nombres");
        }
    }
}
```

Invocar Espacio de Nombres

```
nombre_espacio_de_nombres.nombre_elemento_espacio_de_nombres

espacioNombres1.clase
```

Espacios de Nombres

Directiva *Using*

```
using System;
…
…

Console.WriteLine(“Hola Mundo”);
```

Sin Directiva *Using*

```
System.Console.WriteLine(“Hola Mundo”);
```

Clases Abstractas

Declaración Clase Abstracta

```
public abstract class Hablar
        {
                // código fuente
        {
```

Declarando Método Clase Abstracta

```
public abstract void charlar();
```

Clases Abstractas

Derivando la Clase Abstracta

```
public class Conversacion : Hablar
        {
                // código fuente
        {
```

Implementando Método Abstracto

```
public override void charlar()
{
        Console.WriteLine("Hola estamos teniendo una
conversación sobre las clases abstractas");
}
```