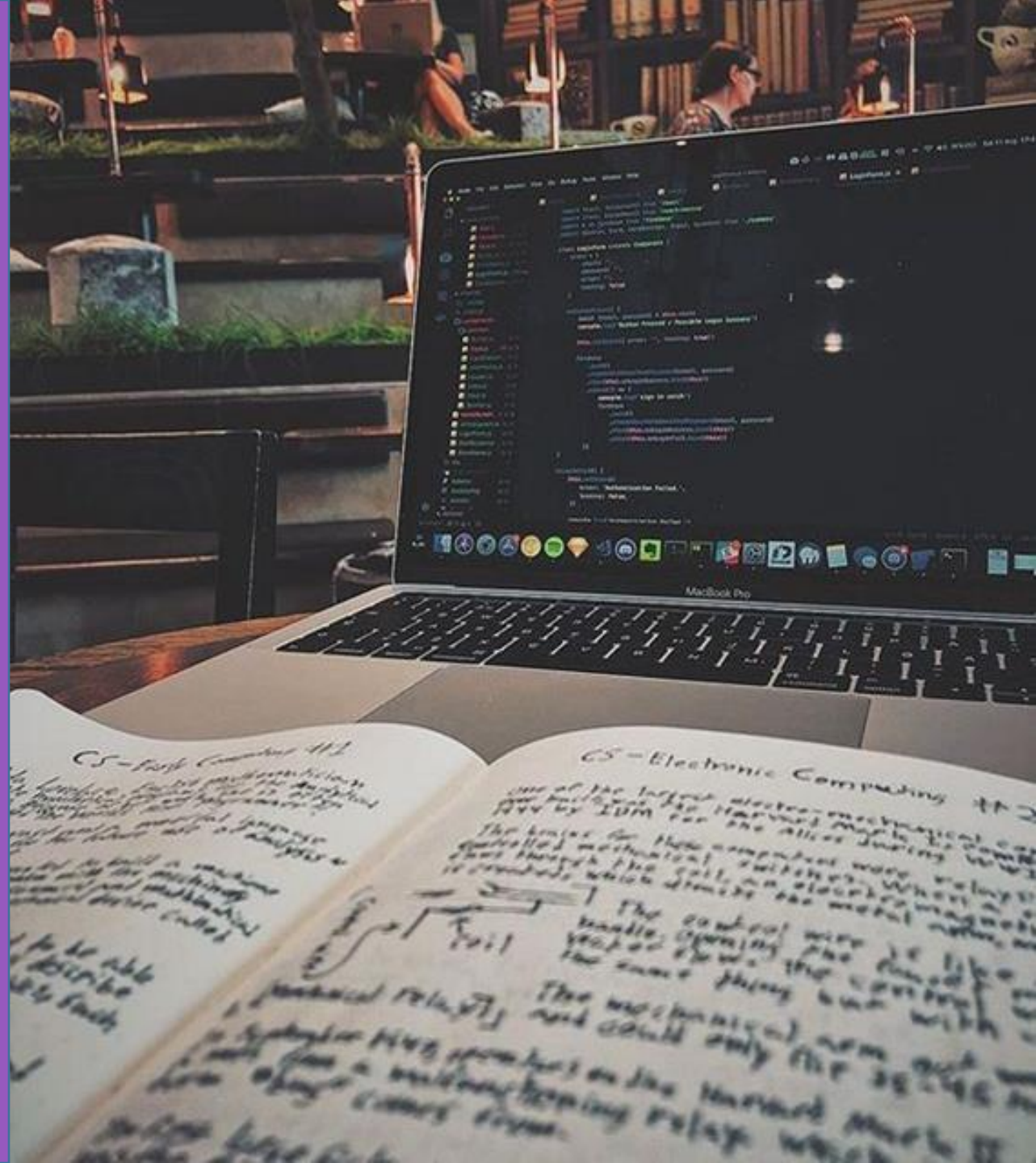
POO

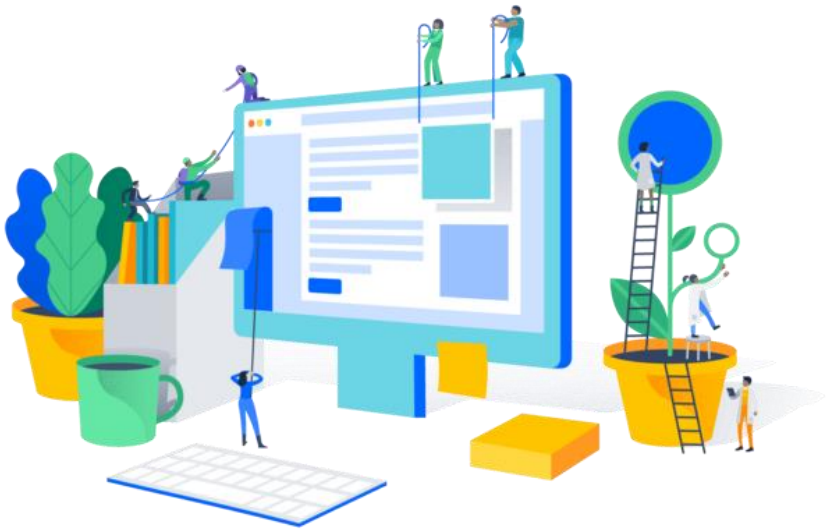
- Asociación entre clases
- Nivel de acceso
- Composición
- Agregación



Arquitectura de una aplicación POO

Construyendo un sistema

En los inicios de la Ingeniería de Software, el desarrollo de software se realizaba libremente, pero con el tiempo se han ido descubriendo y desarrollando nuevos modelos y estándares[1], con base a las cuales se puedan resolver las problemáticas modernas. A estos, se les ha denominado **arquitectura de software**, porque, a semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software.



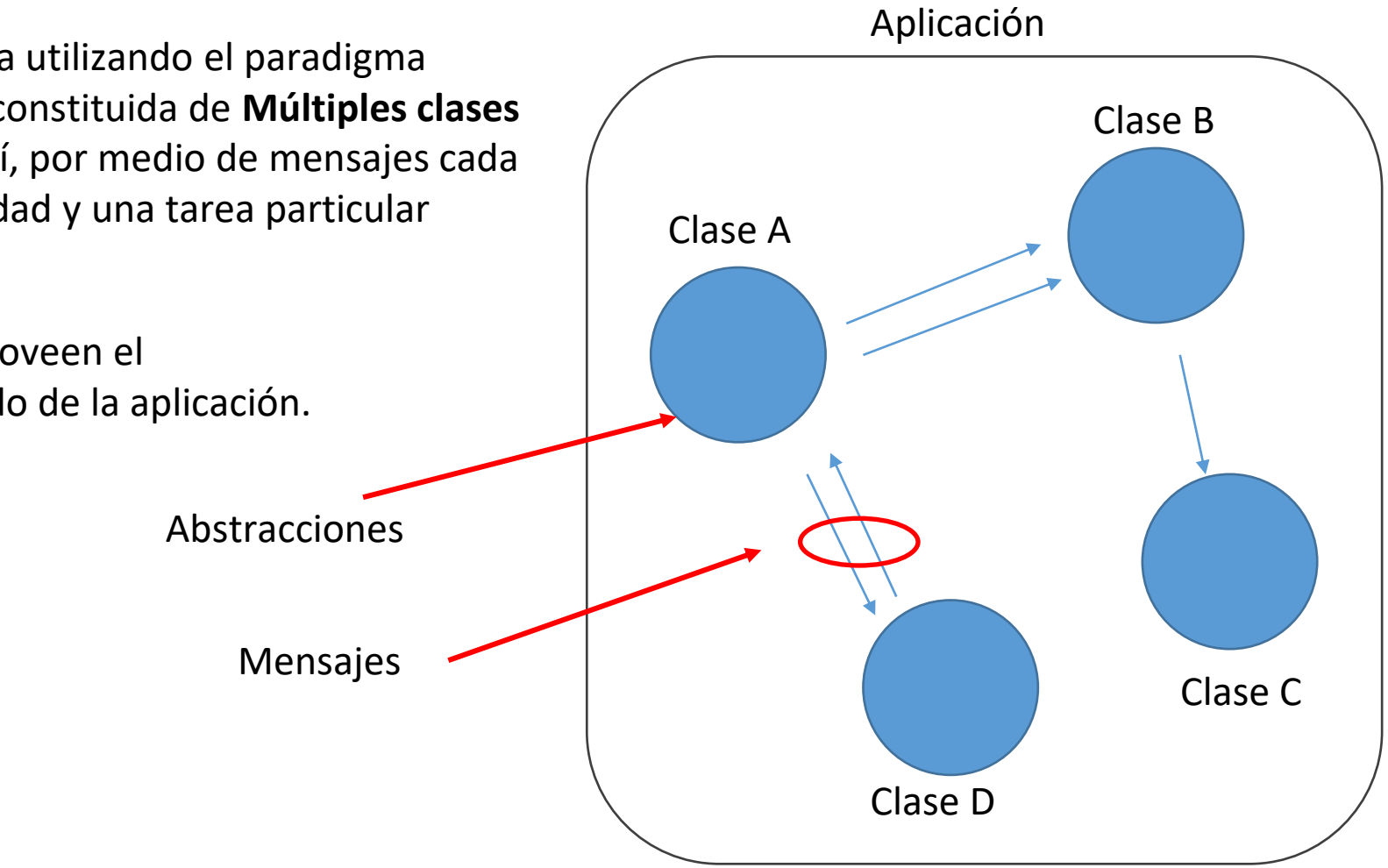
https://es.wikipedia.org/wiki/Arquitectura_de_software

Arquitectura de una aplicación Poo

Construyendo un sistema

Una **aplicación** constituida utilizando el paradigma orientado a objetos está constituida de **Múltiples clases** que interaccionan entre sí, por medio de mensajes cada una con una responsabilidad y una tarea particular dentro de la aplicación.

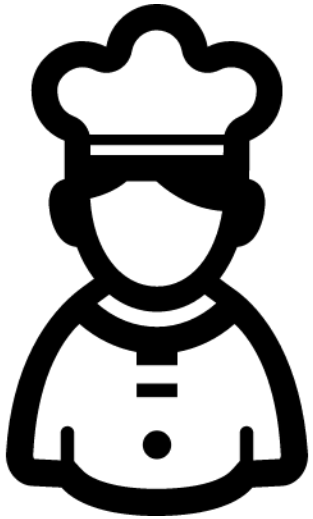
Todas las clases juntas proveen el comportamiento esperado de la aplicación.



Arquitectura de una aplicación Poo

Interacción entre partes

Cocinero

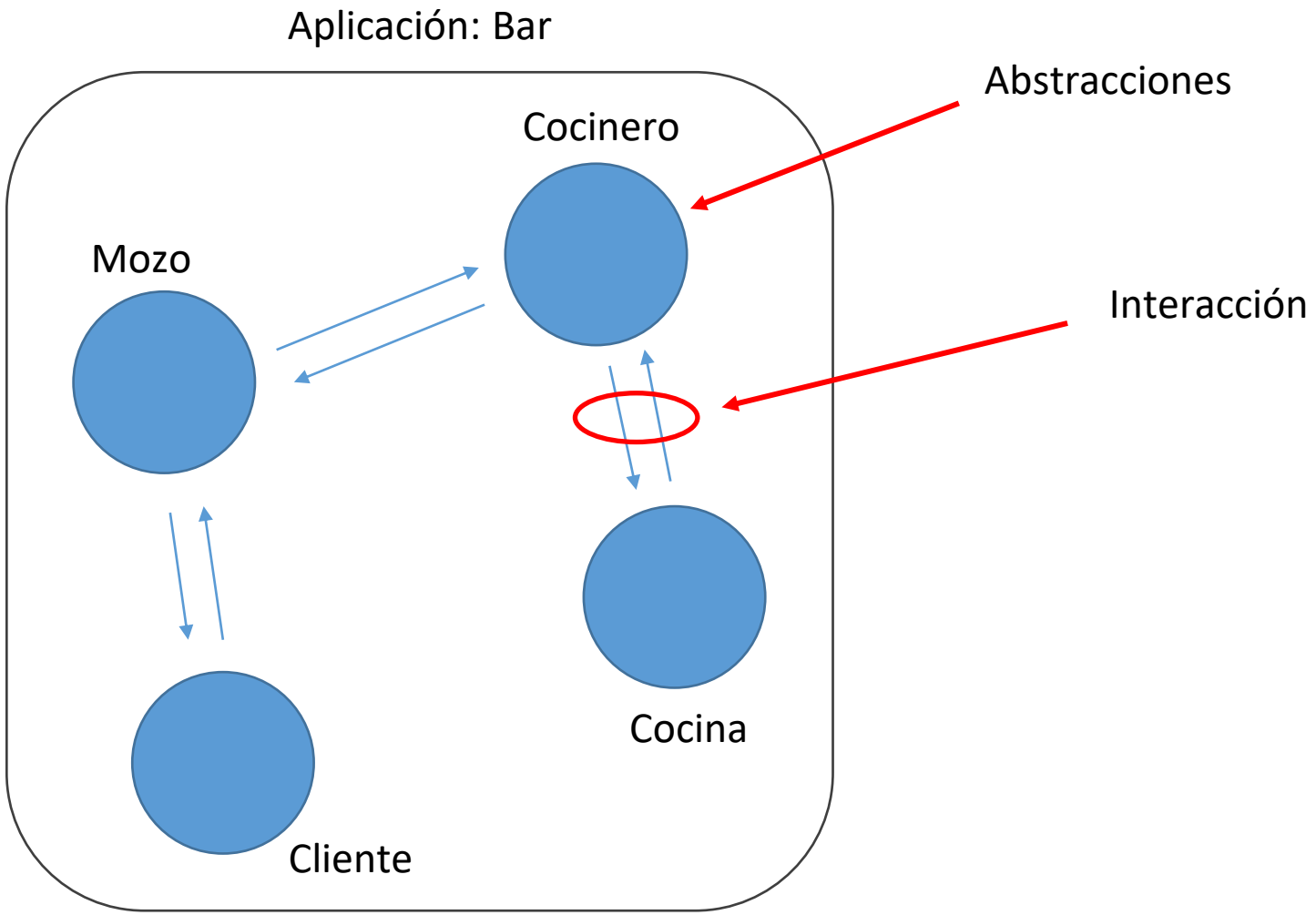


Mozo

Cliente

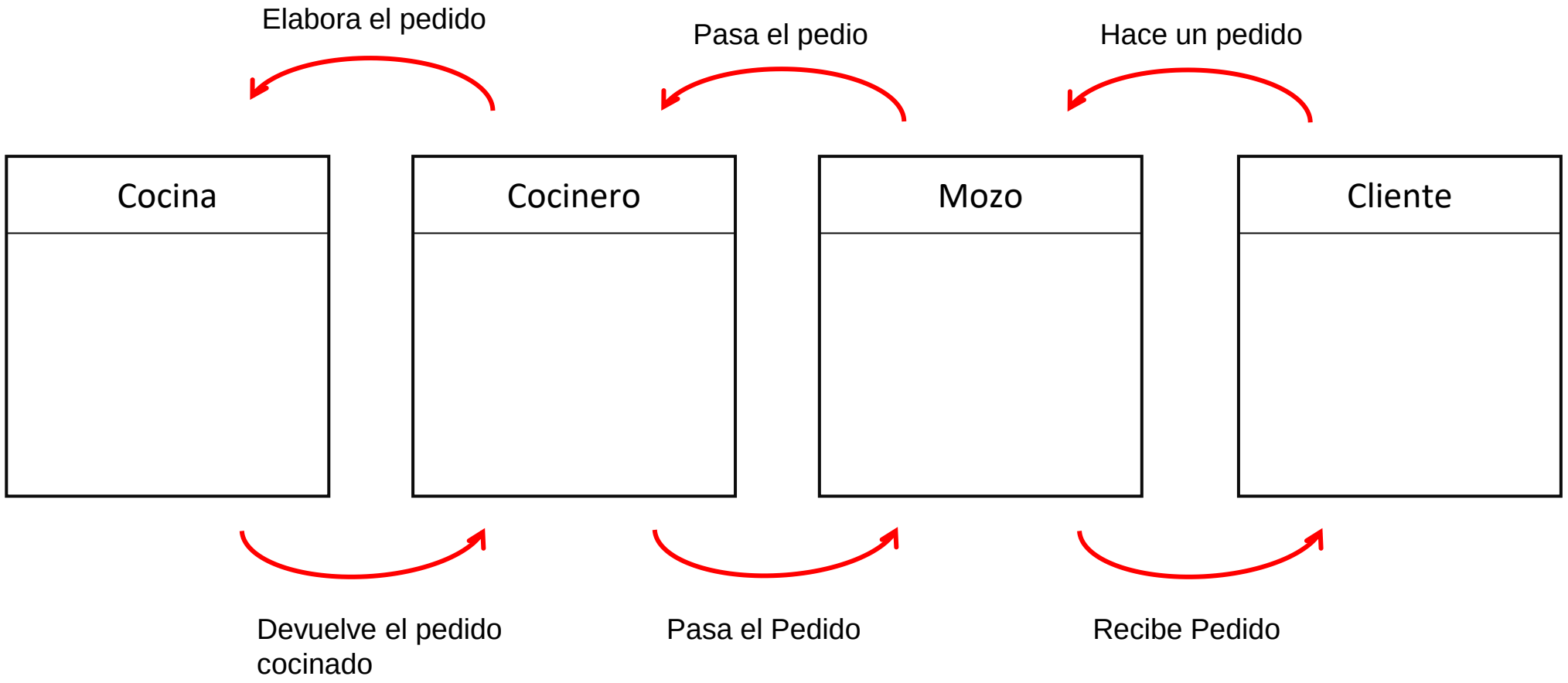
Arquitectura de una aplicación POO

Simulando un bar con clases



Arquitectura de una aplicación POO

Interacción entre partes



¿Cómo podemos vincular clases?

Asociación entre Clases

Definición

Es un tipo de relación donde una clase contiene a otra, es decir, una clase es miembro de otra clase.

Tipos de asociación

- Composición
- Agregación

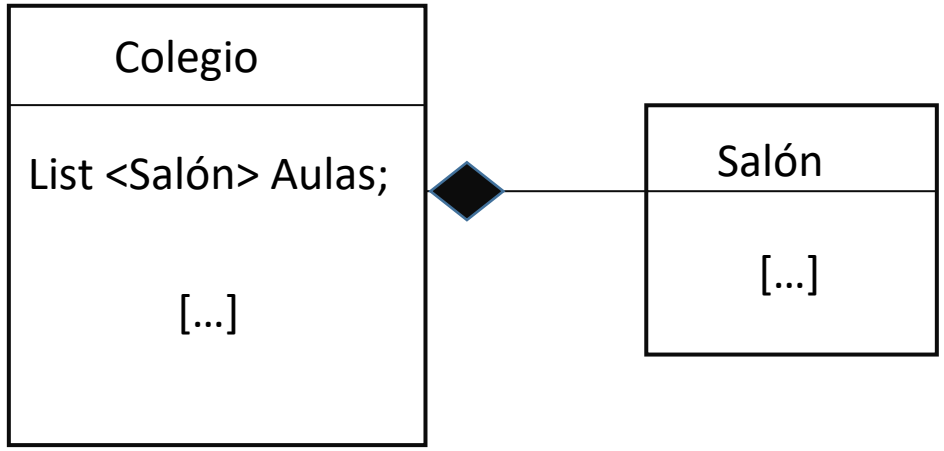
```
public class Posicion
{
     public int X { get; set; }
     public int Y { get; set; }
     public Posición()
     {
          […]
     }
}

public class Rectangulo
{
     public Posición MiPosicion { get; set;}
     public Rectangulo()
     {
          […]
     }
}
```

Asociación entre Clases

Composición

La Composición, es una relación más fuerte que la Agregación.
En este tipo de asociación el objeto contenido **no puede** existir **independientemente** del objeto que lo contiene.
El tiempo de vida del objeto miembro está condicionado por el tiempo de vida del objeto que lo incluye.



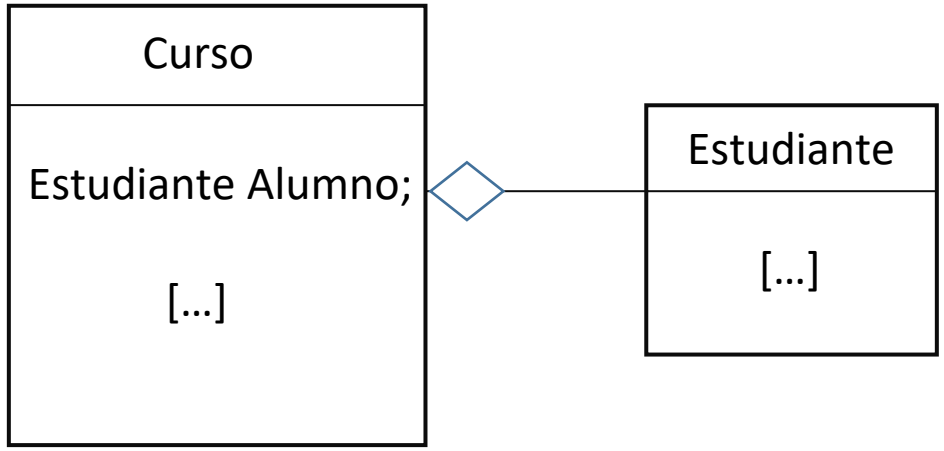
Normalmente, nos referimos a este tipo de relación con la expresión “parte de”, de la siguiente forma:

Ej: Salón es parte de un Colegio.

Asociación entre Clases

Agregación

La Agregación, es una relación entre clases más débil que la Composición.
En este tipo de asociación el objeto contenido **puede existir con independencia** del que lo contiene.

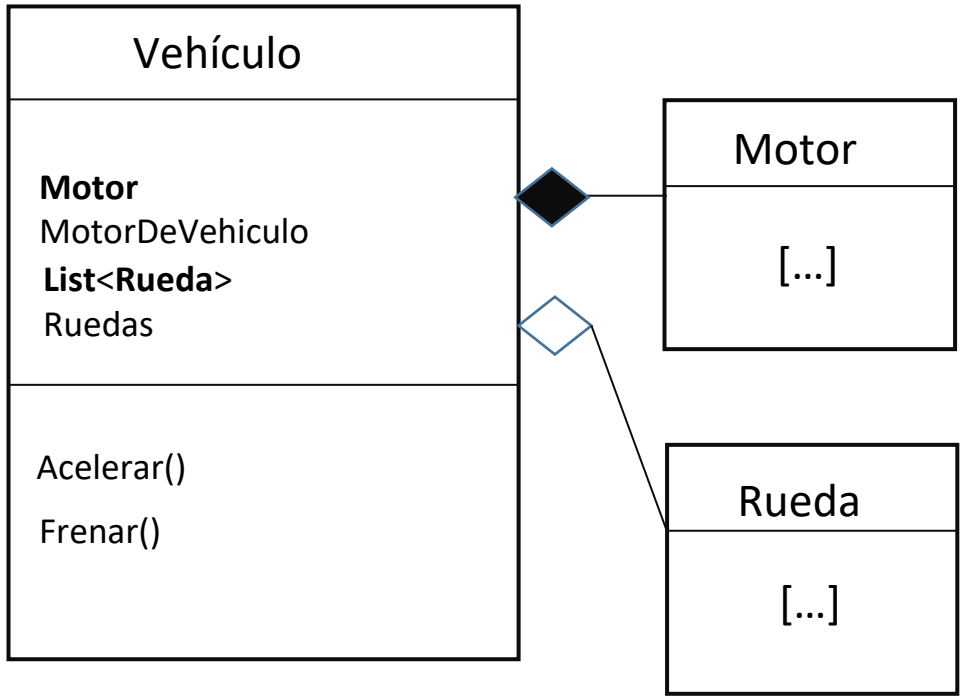


Normalmente, nos referimos a este tipo de relación con la expresión “tiene un”, de la siguiente forma:

Ej: El Curso tiene un Estudiante.

Asociación entre Clases

Ejemplo de asociaciones



Vehículo Tiene una o varias Ruedas.

Motor forma parte de un Vehículo.

```
public class Motor
{
     public Motor()
     {
          […]
     }
}

public class Rueda
{
     public Rueda()
     {
          […]
     }
}


public class Vehiculo
{
     public Motor MotorDeVehiculo { get; set;}
     public List<Rueda> Ruedas { get; set;}

     public Vehiculo()
     {
          […]
     }
}
```

Asociación entre Clases

Dejando en claro

- Siempre vamos a tratar de construir software evitado el acoplamiento
- Una asociación se conoce como **composición** cuando **un objeto posee otro**. Mientras que una asociación se conoce como **agregación** cuando un objeto **usa otro objeto**.
- Para el encapsulamiento el uso de las reservadas **prívate** y **public** va a ser fundamental.

Miembros de una clase

Nivel de Acceso

Miembros de instancia → Accesible desde un **objeto**

Miembros estáticos → Accesible desde la **clase**

```
public class Posicion
{
      public int X { get; set; }
      public int Y { get; set; }
      public Posición()
      {
            […]
      }
}

public class Rectangulo
{

      public Posición MiPosicion { get; set;}
      public int Alto  { get; set; }
      public int Ancho { get; set; }
      public Rectangulo()
      {
            […]
      }

      Static public int CaluloArea(Rectangulo rect)
      {
            return rect.Alto * rect.Ancho;
      }

}
```

Desafío

Considere la siguiente situación: En una distribuidora surgió la necesidad de llevar un control de las tareas realizadas por sus empleados. Usted forma parte del equipo de programación que se encargará de hacer el módulo en cuestión que a partir de ahora se llamará módulo TaskLists: