Clases Contenedoras.

Una clase contenedora se usa para contener o agrupar referencias a objetos relacionados, con el objeto de manipularlos o crear nuevas funcionalidades.

Ejemplo.

Considere una clase Cocina que puede ofrecer los servicios de ser capaz de cocinar comidas y mantener comidas calientes. Usted, además, desarrolla una clase Horno y una clase Refrigerador. Los objetos de las tres clases están relacionados, pero la herencia no es apropiada puesto que Horno y Refrigerador no comparten miembros con Cocina. Para permitir que Cocina controle el funcionamiento de Horno y Refrigerador, se puede utilizar una clase contenedora (en este caso, Cocina).

Utilización de clases contenedoras.

```
public class Horno {
      // lo que la clase hace.
}

public class Refrigerador {
      // lo que la clase hace.
}

public class Cocina {
      private Horno miHorno;
      private Refrigerador miRerigerador;
      // etcétera.
}
```

> **Nota** – Para sacar ventaja de la herencia, especialmente si hay otras habitaciones que va a definir en su programa, podría definir una clase Habitación. Luego, podría declarar que Cocina extiende de la clase Habitación (“Una Cocina es una Habitación”).

Verificación.

Así como la frase “es un/a” valida la herencia, la frase “tiene un/a” valida la contención. Por ejemplo, una cocina tiene un horno, una cocina tiene un refrigerador.

Interacción entre Objetos.

Los objetos interaccionan enviando mensajes unos a otros. Este proceso se denomina pasaje de mensajes (message passinng).

El proceso exacto de envío de mensajes depende de la naturaleza de los objetos que se modelan. De todos modos,

- Un objeto envía un mensaje a otro (el objeto receptor).
- El objeto receptor puede enviar otro mensaje, cambiar sus variables o reaccionar de alguna otra forma adecuada para un objeto con sus variables.
- Los objetos pueden interaccionar solamente a través de sus interfaces públicas. Los mensajes se manejan por los métodos de la interfaz pública del objeto receptor.

Debido a las reglas de encapsulamiento y ocultamiento de la información, no hay otra forma de influir (es decir, comunicarse con) un objeto.

```
public class Cocina {
	private Horno miHorno;
	private Refrigerador miRerigerador;
	// etcétera.

	public void calentar (Comida comida) {
		this.miHorno.calentar(comida);
	}
}
```

El mensaje es recibido por el objeto Horno y se invoca el comportamiento .calentar().

Asociación y Composición.

Los objetos interaccionan a través de una de dos relaciones: composición o asociación.

Asociación.

Dos objetos independientes colaboran para alcanzar una meta, tal como una persona usando una computadora (relación "usa un/a"). Luego que el objetivo es logrado, los objetos continúan su existencia independientemente uno del otro. La asociación es una relación menos estrecha que la relación de composición.

> **Nota** – la composición es una relación de contención, en la que un objeto contiene al otro.

Ejemplo.

Una Cocina utiliza a ambos (Horno y Refrigerador), dado que los hornos y los refrigeradores pueden existir independientemente de las cocinas.

```
public class Cocina {
	private Horno miHorno;
	private Refrigerador miRerigerador;
	// etcétera.

	public void setMiHorno(Horno miHorno) {
		this.miHorno = miHorno;
	}

	public void setMiRerigerador(Refrigerador
miRerigerador) {
		this.miRerigerador = miRerigerador;
	}
}
```

Composición.

Un objeto contiene otro objeto, como por ejemplo un lápiz contiene una mina (relación "tiene un/a"). En muchas situaciones, no tiene sentido que ciertos objetos existan en forma

independiente. En este caso, la composición es una relación más adecuada ya que no tiene sentido que un lápiz exista sin su mina. Si el lápiz no tiene mina, no es un lápiz.

Ejemplo.

Una clase que realiza la composición, tal como Auto, inicializa los objetos que la componen dentro de su bloque de sentencias.

```
public class Auto {
      private Asiento miAsiento = new Asiento();
      private Rueda miRueda = new Rueda();
      private Carroceria miCarroceria = new Carroceria();
      // estcétera.
}
```

Nota - se usan clases contenedoras para definir ambas relaciones. El lápiz "tiene una mina". El lápiz "usa un" sacapuntas.

Recursos.

Bibliografía.

[MEYER-97] Object-oriented software construction *Bertrand Meyer* - Prentice Hall PTR - 1997

[MEYER-09] Touch of class: learning to program well with objects and contracts, *Bertrand Meyer* - Springer - 2009

[SUN EDUCATIONAL SERVICE] Migración a la Programación OO con Tecnología Java – SL-210.

Links.

Association, Composition and Aggregation in Java.