Modelado y Diseño de Clases– Asociación y Composición

Programación

Septiembre 2016

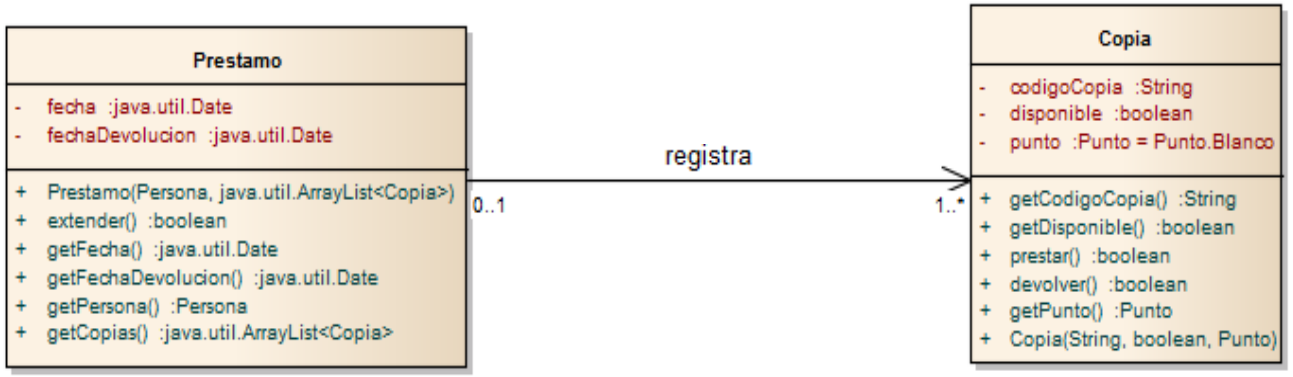
Recordatorio….

- El dominio de un lenguaje de programación no te garantiza la realización de un buen diseño del sistema

Asociaciones

- Las asociaciones permiten que objetos puedan unirse para trabajar en conjunto
- En las asociaciones los objetos asociados son independientes entre sí
- Normalmente para las asociaciones se utilizan frases como “usa.., realiza…”
  - **Trabajador** realiza **Registro**
  - **Préstamo** registra **Copia**
  - **Piloto** usa **Auto**

Codificación de una Asociación




```
public class Prestamo{
     //Atributos
     private String fecha;
     private String fechaDevolucion;
     //Asociacion Presta registra Copia
     private Copia[] registra; //Por la multiplicidad de 1..*
     private int indice; //Por la multiplicidad es necesario llevar un índice
     //Operaciones
     ……
     //Operación de Asociación
     public void registrarCopia(Copia copia){
          this.registra[indice++] = copia;
     }
}
```

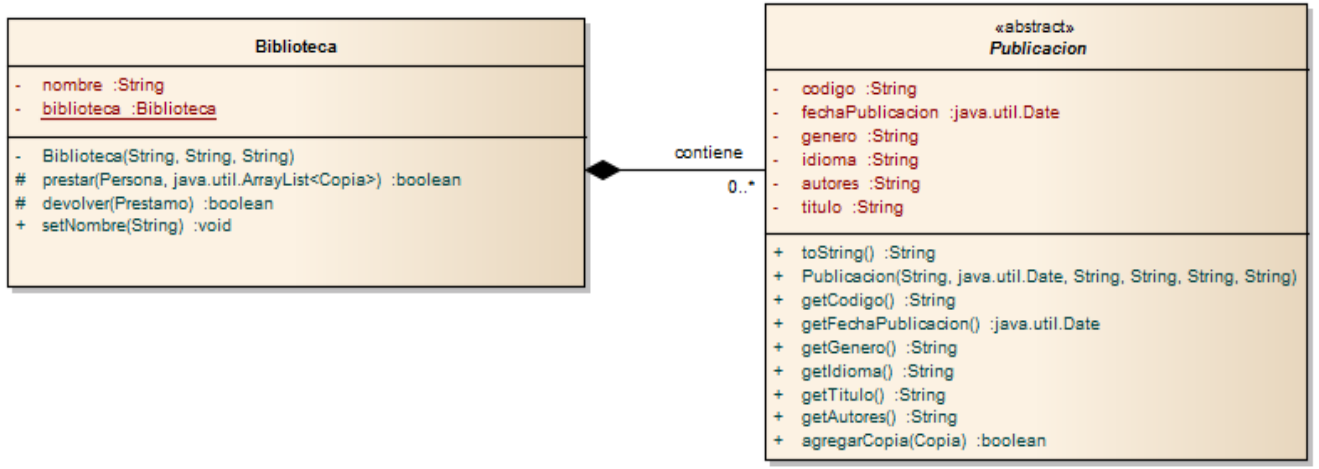
- La creación de un objeto *Copia* **NO DEPENDE** de *Préstamo* y viceversa

Asociaciones

- En la asociación anterior la existencia del objeto ***Préstamo*** no depende de la existencia de los objetos ***Copia***

- Es posible asignar o quitar el objeto copia del arreglo **`registra`**, sin que la existencia de Préstamo se vea afectada

Codificación de una Composición




```
public class Biblioteca{
      //Atributos
      ….
      private Publicacion[] contiene = new Publicacion[10]; //Por la multiplicidad de 0..*
      private int indice; //Por la multiplicidad es necesario llevar un índice
      //Operaciones
      ……
      //La Composición expresa dependencia, entonces incluirla en el constructor de biblioteca
      public Biblioteca(){
            //Construir publicaciones: Revistas, Libros…
            registra[0] = new Libro(“par1”, “part1”, “part1”);
            registra[1] = new Libro(“par2”, “part2”, “part2”);
            registra[2] = new Revista(“par3”, “part3”, “part3”);
      }
      Public Publicacion[] getPublicaciones(){
            return contiene;
      }
}
```

- La creación de objetos *Publicación* **DEPENDE** de *Biblioteca*

Tiempo de Vida de los Objetos

- Es el tiempo que transcurre desde que un objeto es creado hasta que se destruye
- En la **Asociación**, los tiempos de vida de los objetos asociados se cruzan mientras trabajan juntos, no obstante, esto no significa que se hayan creado al mismo tiempo
- En la **Composición**, los componentes como la clase contenedora, son creados y destruidos al mismo tiempo, es decir, tienen el mismo tiempo de vida
- La **Composición** es una relación muy dependiente, si cualquier objeto es destruido, los demás también

Asociación o Composición

- **Un Reloj** tiene **Manecillas //Composición**
- **Un Reloj** utiliza **Manecillas //Asociación**

- ¿Cuál elegir?
  - Se debe tomar en cuenta la flexibilidad para implementar una y otra
  - Dependerá de la perspectiva
  - Por ejemplo, si se necesita tener un control sobre cada pieza que conforma el reloj, reemplazando piezas defectuosas entonces convendría **ASOCIACIÓN**…perspectiva del fabricante
  - Sin embargo, en una perspectiva del usuario final (consumidor) si las manecillas son defectuosas, el usuario desecharía todo el reloj y optaría por comprar uno nuevo, en este caso convendría **COMPOSICIÓN**
  - **La elección dependerá de lo que se necesite**

Conclusiones

- Considerando los conceptos anteriores, resulta de vital importancia saber identificar en qué momento utilizar herencia u optar por la composición (para la reutilización de código)

- Herencia (IS-A) y Composiciones (HAS-A) son los dos mecanismos más comunes para la reutilización

- Herencia: Relación ser-un: Entre clases, y significa contener una clase
  - Un Coche es un Vehículo
- Composición: Relación tener-un: Entre objetos, y significa contener un objeto
  - Un Coche tiene un tipo de Motor