UML y Conceptos de Programación (Parte II)

Herencia y Colaboración - Fundamentos de Programación

Competencias

- Comprender conceptos de Herencia y Colaboración para realizar diagramas de clases de alto nivel.
- Aplicar conceptos de Herencia y Colaboración para realizar diagramas de clases de alto nivel.

Introducción

Al estudiar "Herencia y Colaboración" aprenderemos nuevos conceptos de programación orientados a objetos, sus relaciones y el uso práctico en el mundo moderno. Estos conceptos son de uso cotidiano en nuestras vidas, además de pilares fundamentales de JAVA porque son la base para entender el paradigma.

A partir de su aplicación en la vida cotidiana lograremos comprender la herencia y la colaboración a cabalidad, pero para lograr este objetivo necesitaremos que la primera unidad de UML esté aprendida ya que es la base de los conocimientos globales de Java para codificar y reconocer los distintos conceptos.

Colaboración y Herencia

Colaboración

En la unidad 1 de UML, hablamos sobre propiedades y acción u operación de un objeto. La colaboración nos permite unir uno o más objetos distintos entre sí para crear uno nuevo, esto nace por los conceptos técnicos como, por ejemplo, "requiere de ", "necesita a" en las acciones de los mismos objetos.

Por ejemplo, describiremos una relación de colaboración de dos objetos distintos:

Se necesita crear un nuevo objeto, este objeto será llamado **AutoConRuedas**. Para crear el objeto **AutoConRuedas** se necesitan 4 objetos de tipo **Rueda**. El objeto **Rueda** ahora será una propiedad o atributo del objeto **AutoConRuedas**.

Esto es un claro ejemplo de colaboración de objetos distintos, aquí el término a utilizar es "el objeto **AutoConRuedas** *requiere de* cuatro objetos de tipo **Rueda** para ser un auto con ruedas".

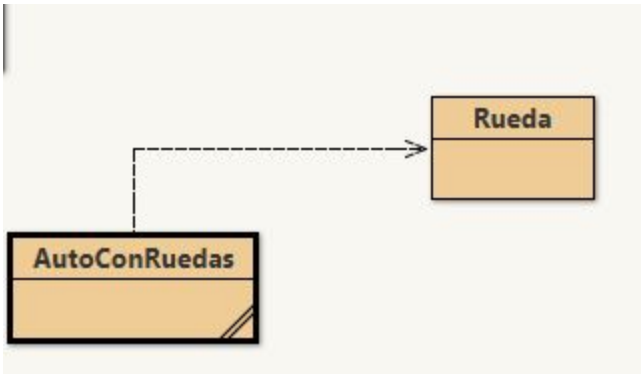

Imagen 1. Dos Clases Java colaborando entre ellas.
Fuente: Desafío Latam.

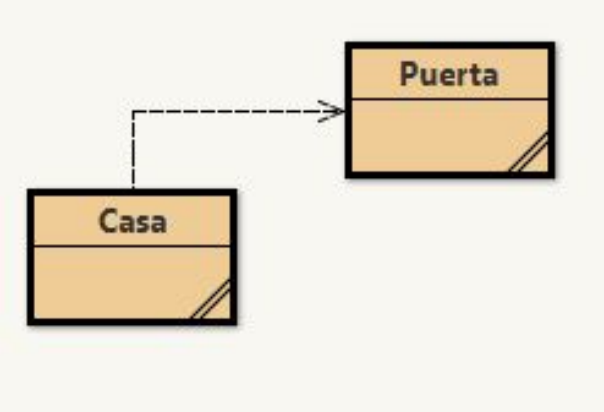

Imagen 2 : Una Puerta colabora con el Objeto Casa.

Herencia

La herencia en Java se define como objetos con jerarquía relacionadas entre ellas, es distinta a la colaboración porque en la colaboración no existe jerarquía.

Java utiliza este pilar para seguir con el concepto de no repetir código, propiedades u operaciones, esto es para utilizar lo menos posible las líneas de código y hacer sus programas más livianos y rápidos.

Con la herencia esto se viene a reafirmar, ya que nace el concepto de superclase, donde todas las propiedades y operaciones comunes entre sí, siempre que sean públicas, se heredarán a las clases hijas o de jerarquía menor.

Ejemplo:

Tenemos el objeto **Abogado** con sus atributos nombre, rut y título, también tenemos el objeto llamado **Juez** con sus atributos nombre, rut y dirección.

Estos objetos tienen atributos similares y relación entre sí, y para no crear dos objetos donde se repiten las mismas operaciones y atributos, se crea una superclase donde se identifica la relación directa entre **Juez** y **Abogado,** esta relación es que ambos son personas.

Con esta información podemos crear el objeto Persona con los atributos rut y nombre, y bajo la definición de lo que es herencia, se dirá que la clase Persona le heredará todos sus atributos y operaciones públicas a las clases **Juez** y **Abogado**.

La clase **Juez** solo tendrá el atributo dirección y **Abogado** solo el atributo título, sin embargo, Java interpretará que la clase **Persona** y **Abogado** es solo una, así como **Persona** y **Juez,** ya que al heredar y tener jerarquía Java lo interpreta como una sola clase, para la **Java Virtual Machine (JVM)** solo existen dos clases **Juez** y **Abogados** con atributos y operaciones de la clase **Persona**.

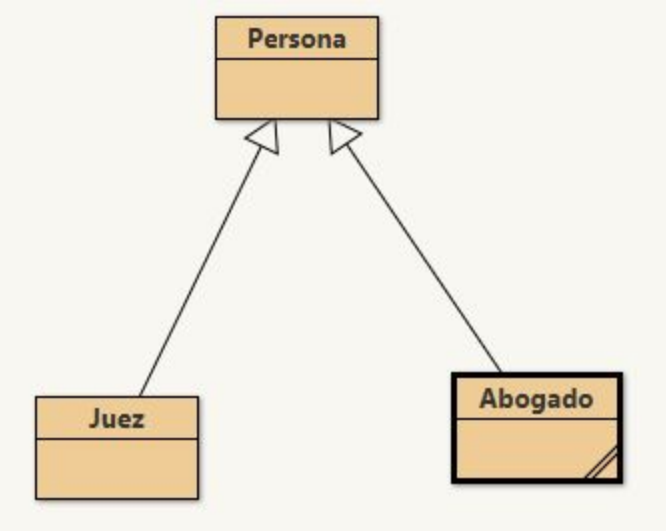

Imagen 3: Representación de Herencia en Java de las clases Persona-Juez-Abogado.
Fuente: Desafío Latam.

Ejercicio guiado: Identificar las clases y sus relaciones

En un zoológico hay un zorro, un león y una ciudad.

El zorro tiene:

- Edad.
- Tipo de zorro.
- Origen.

El león tiene:

- Edad.
- Origen.
- Sexo.

Identificar y describir las clases y sus relaciones.

Solución Guiada:

Se debe identificar lo siguiente:

1. Si hay relación entre un león y zorro.
2. Dónde están ubicados el león y el zorro.
3. Características de cada objeto.
4. Identificar la relación entre zoológico, león y zorro.

Luego de identificar las clases y sus relaciones, quedaría de la siguiente manera:

Animal:

- edad: int
- origen: String

Zorro es un Animal:

- tipoDeZorro: String

León es un Animal:

- sexo: String

Zoológico:

- zorro: Zorro
- león: León
- ciudad: String

Ejercicio propuesto (1)

Analizar y crear distintos tipos de relaciones y objetos según el siguiente enunciado:

En un estacionamiento pueden estacionarse cinco camionetas y cinco autos sedán. En sus distintas sucursales.
Los autos sedán tienen:

- Cantidad de puertas.
- Marca.
- Modelo.

Una camioneta posee:

- Tamaño.
- Modelo.
- Marca.
- Precio.

Ambos vehículos pueden encender y apagar su motor.

Identificar y describir las clases y sus relaciones.

Colaboración y Herencia en diagramas de clases

Competencias

- Comprender el comportamiento de las relaciones entre los conceptos de herencia y colaboración.
- Crear Diagramas de clases de alto nivel para tener una mejor lectura del código que vamos a desarrollar.

Introducción

Comprenderemos la importancia de crear diagramas de clases con conceptos avanzados de programación.

Realizar y entender el buen uso de un diagrama de clases nos ayudará a obtener de manera más rápida y ágil el código, sin necesidad de mirar y analizar a fondo todos nuestros códigos.

UML - Herencia y Colaboración

Conexiones y nomenclaturas de Herencia y Colaboración en UML

Existen distintos tipos de conectores dentro de un diagrama de clases, nosotros estudiaremos tres para continuar con lo aprendido en las unidades anteriores. Tenemos nomenclaturas de conexiones para identificar si la relación es:

- Directa, de tipo normal o arreglo.
- Bidireccional de tipo normal o arreglo.
- Tipo herencia.

A continuación, describiremos los tipos de relación:

- **Asociación**: Conexión entre clases, esta relación nos permite realizar colaboración directa entre ambos objetos tanto de forma directa como bidireccional, también nos permite relacionar objetos como arreglos.

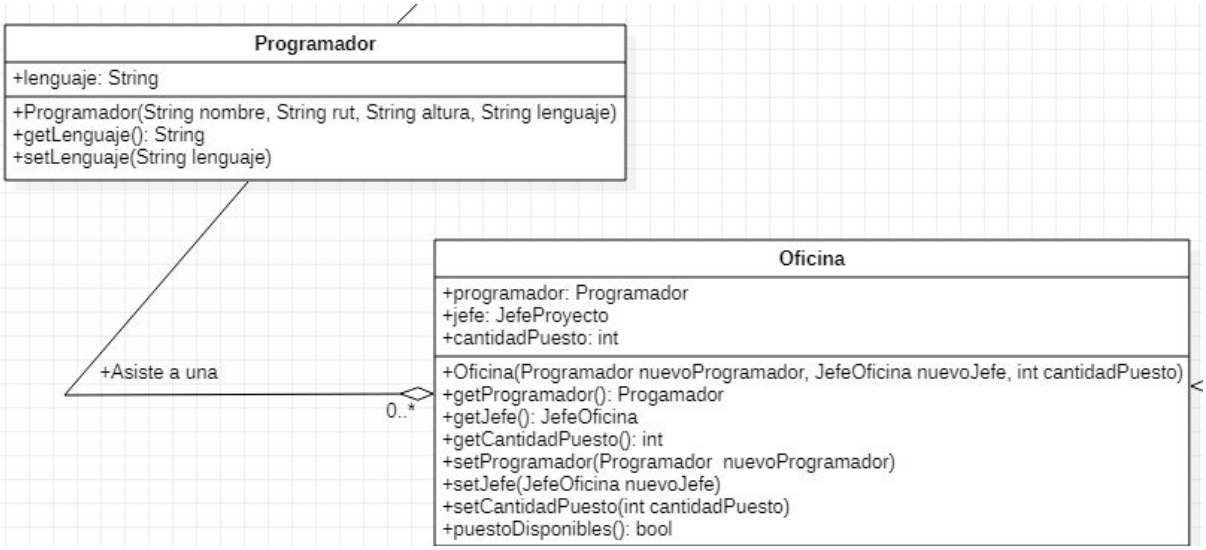

Imagen 4. Relación de una asociación de tipo directo de arreglos de objetos.
Fuente: Desafío Latam.

- **Generalización / Especialización**: Este tipo de conexión las usamos para representar herencia en un diagrama de clases.

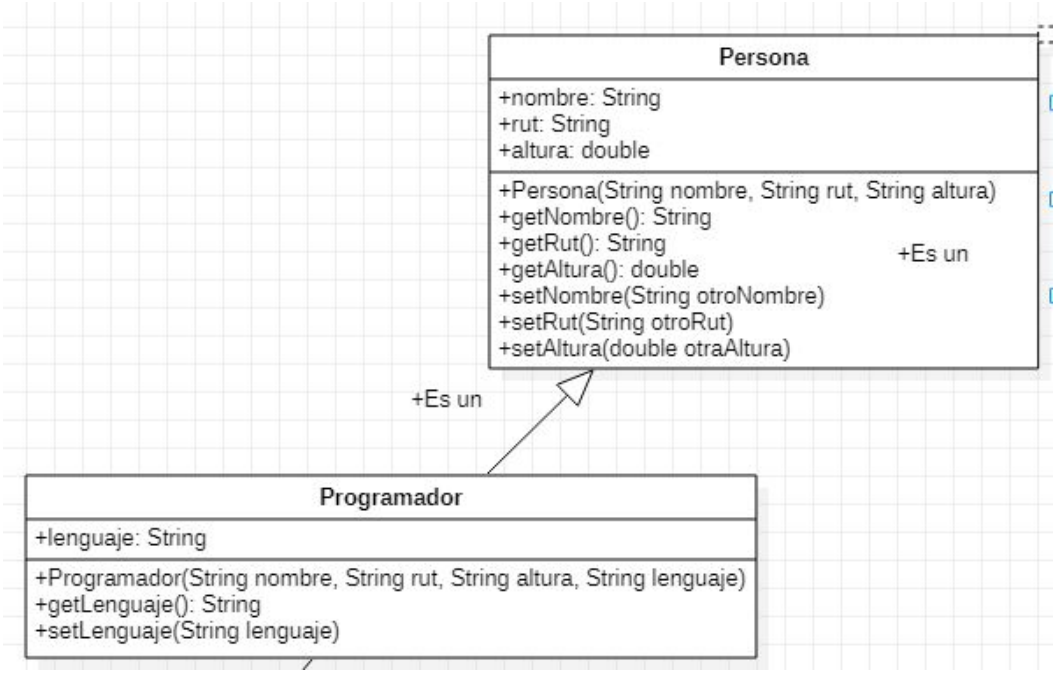

Imagen 5 : Herencia Persona-Programador.
Fuente: Desafío Latam.

A continuación, describiremos la nomenclatura que se utiliza para las conexiones de clases:

Multiplicidad	Significado	Ejemplo
1	Uno y solo uno	Un Computador tiene solo un Mouse.
0..1	Cero a uno	Una Casa puede tener ninguna o una Piscina.
N..M	N hasta M	Una Juguetería tiene varios Juguetes.
*	Cero a varios	Una Biblioteca puede tener cero o varios Estudiantes.
0 .. *	Cero a Varios	Un Estacionamiento puede tener cero o varios Autos.
1..*	1 a varios	Un Escritorio tiene uno o varios Lápices.

Tabla 1. Nomenclatura conexiones de clases.
Fuente: Desafío Latam.

Para complementar la información de esta sección te invitamos a acceder al material complementario y revisar los links sugeridos.

Ejercicio guiado: Oficina de informática

Realizar un diagrama de clases para el personal de una oficina de informática, sabemos que en la oficina existen un jefe de proyecto y N programadores.

El **jefe de proyecto** tiene como atributos los siguientes campos:

- nombre
- rut
- altura
- área

Un **programador** tiene como atributos los siguientes campos:

- nombre
- rut
- altura
- lenguaje

Una **oficina** tiene la siguiente estructura:

- programador: Programador
- jefeProyecto: JefeProyecto
- cantidadPuesto: int

Realizar un diagrama de clases utilizando herencia y colaboración.

Ejercicio resuelto

1. Debemos realizar un análisis de las clases involucradas, en el ejercicio nombran tres: *Programador*, *Jefe de proyecto* y *Oficina*.

2. Después de identificar las clases, necesitamos analizar cuáles clases tienen relación y cuáles no, en este caso programador y jefe de proyecto tienen relación ya que ambos son personas.

3. Se crea una superclase de tipo *Persona* con los atributos nombre, rut, altura y se crea dos subclases, *Programador* con atributo lenguaje y *Jefe de proyecto* con atributo área.

4. Se utiliza las conexión de herencia entre *Programador, JefeProyecto* y *Persona*.

5. Se conecta la clase *Programador* y *JefeProyecto* con la clase *Oficina*.

6. La conexión de *Programador* con *Oficina* es de cero a muchos, ya que puede o no puede haber programador, pero si debe existir un Jefe de proyecto.

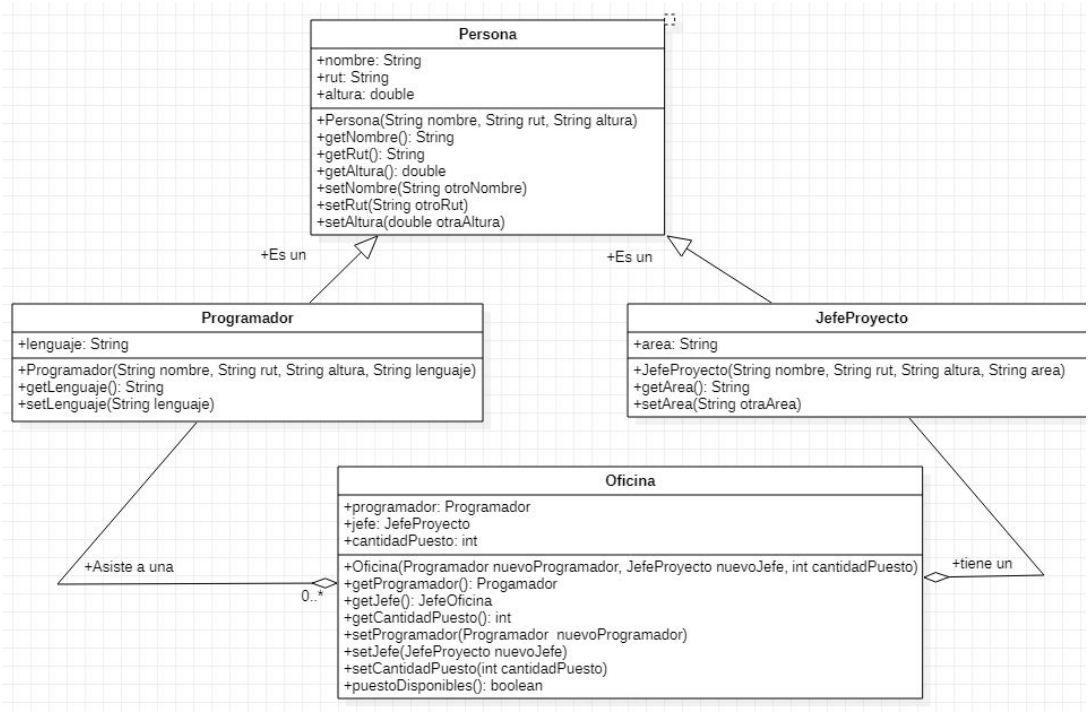

Imagen 6: Ejercicio resuelto en diagramas de Clases.
Fuente: Desafío Latam.

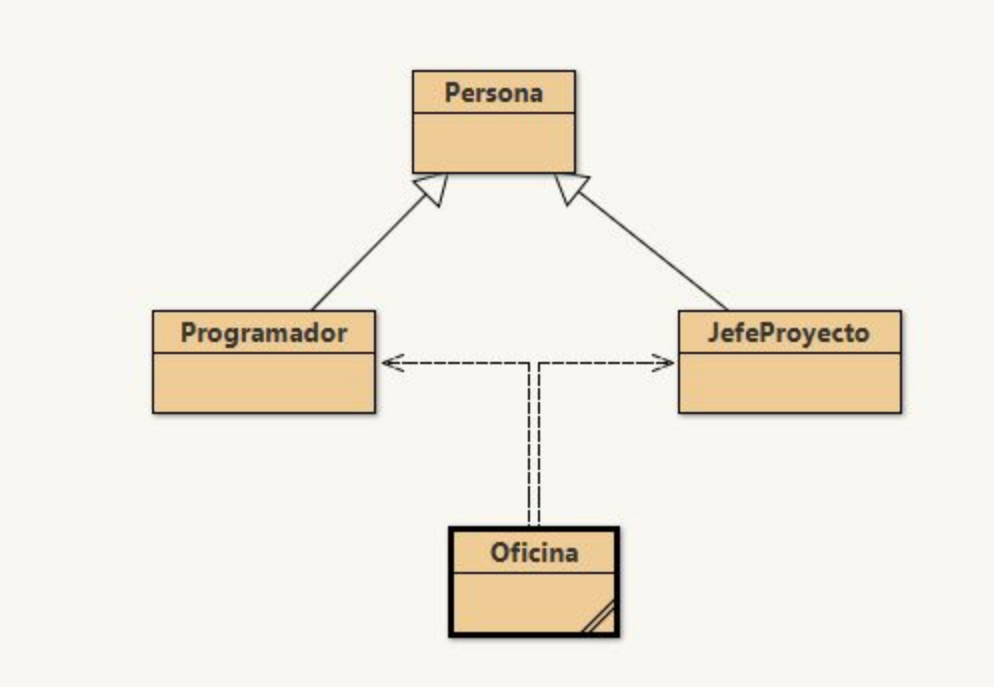

Imagen 7: Visualización del Ejercicio resuelto con clases Java.
Fuente: Desafío Latam,

Ejercicio propuesto (2)

La Aerolíneas JAVA-Fly necesita llevar un registro de sus vuelos nacionales e internacionales, para esto se les solicita modelar el sistema con estas clases.

Requerimiento:

Crear diagrama de clases utilizando Herencia y Colaboración donde corresponda.

Piloto :

- estaDisponible: campo de tipo boolean.

Vuelos Internacionales:

- lugarOrigen: Campo de tipo String.
- lugarDestino: Campo de tipo String.
- cantidadPasajeros: Campo de tipo entero.
- cantidadEscalas: campo de tipo entero.
- piloto: Campo de tipo Piloto

Vuelos Nacionales :

- lugarOrigen: Campo de tipo String.
- lugarDestino: Campo de tipo String.
- cantidadPasajeros: Campo de tipo entero.
- pesoEquipaje: Capo de tipo double.
- piloto: Campo de tipo Piloto

Soluciones ejercicios propuestos

Ejercicio propuesto (1)

Para realizar la solución debemos:

- Analizar si existe relación entre los objetos descritos.
- Analizar los atributos comunes.
- Crear clases padres si fuese necesario.

Vehiculo:

- Marca: String
- Modelo: String

Sedan:

- Cantidad de puertas: int

Camioneta :

- Tamanio: int
- Precio int

Estacionamiento:

- sucursal: String
- sedan: Sedan [4]
- camioneta: Camioneta [4]

Ejercicio propuesto (2)

Para realizar la solución debemos:

- Analizar si existe relación entre los objetos descritos.
- Analizar los atributos comunes.
- Crear clases padres si fuese necesario.

Piloto :

- estaDisponible: campo de tipo boolean.

Vuelos :

- lugarOrigen: Campo de tipo String.
- lugarDestino: Campo de tipo String.
- CantidadPasajeros: Campo de tipo entero.

Vuelos Internacionales:

- cantidadEscalas: campo de tipo entero.
- piloto: Campo de tipo Piloto

Vuelos Nacionales :

- pesoEquipaje: Capo de tipo double.
- piloto: Campo de tipo Piloto