Introducción

1. Diagramas de clases.

Dentro de los diagramas estructurales, y de todos en general, es el más importante porque representa los elementos estáticos del sistema, sus atributos y comportamientos, y como se relacionan entre ellos. Contiene las clases del dominio del problema, y a partir de éste se obtendrán las clases que formarán después el programa informático que dará solución al problema.

En un diagrama de clases podemos encontrar los siguientes elementos:

1. **Clases**: recordemos que son abstracciones del dominio del sistema que representan elementos del mismo mediante una serie de características, que llamaremos atributos, y su comportamiento, que serán métodos. Los atributos y métodos tendrán una visibilidad que determinará quién puede acceder al atributo o método. Por ejemplo una clase puede representar a un coche, sus atributos serán la cilindrada, la potencia y la velocidad, y tendrá dos métodos, uno para acelerar, que subirá la velocidad, y otro para frenar que la bajará.
2. **Relaciones**: en el diagrama representan relaciones reales entre los elementos del sistema a los que hacen referencia las clases. Pueden ser de asociación, agregación y herencia. Por ejemplo si tengo una clase persona, puedo establecer una relación conduce entre persona y coche.
3. **Notas**: Se representan como un cuadro donde podemos escribir comentarios que nos ayuden a entender algún concepto que queramos representar.
4. **Elementos de agrupación**: Se utilizan cuando hay que modelar un sistema grande, entonces las clases y sus relaciones se agrupan en paquetes, que a su vez se relacionan entre sí.

2. Creación de clases.

Una clase se representa en el diagrama como un rectángulo divido en tres filas, arriba aparece el nombre de la clase, a continuación, los atributos con su visibilidad y después los métodos con su visibilidad que está representada por el signo menos "–" para los atributos (privados) y por el signo más "+" para los métodos (públicos).

"Una clase es una descripción de un conjunto de objetos que manifiestan los mismos atributos, operaciones, relaciones y la misma semántica."

(Object Modelling and Design [Rumbaught et al., 1991])

"Una clase es un conjunto de objetos que comparten una estructura y un comportamiento comunes."

[Booch G., 1994]

3. Atributos

Forman la parte estática de la clase. Son un conjunto de variables para las que es preciso definir:

Su nombre.
Su tipo, puede ser un tipo simple, que coincidirá con el tipo de dato que se seleccione en el lenguaje de programación final a usar, o compuesto, pudiendo incluir otra clase.

Además se pueden indicar otros datos como un valor inicial o su visibilidad. La visibilidad de un atributo se puede definir como:

Público: Se pueden acceder desde cualquier clase y cualquier parte del programa.
Privado: Sólo se pueden acceder desde operaciones de la clase.
Protegido: Sólo se pueden acceder desde operaciones de la clase o de clases derivadas en cualquier nivel.
Paquete: Se puede acceder desde las operaciones de las clases que pertenecen al mismo paquete que la clase que estamos definiendo. Se usa cuando el lenguaje de implementación es Java.

4. Métodos

Representan la funcionalidad de la clase, es decir, qué puede hacer. Para definir un método hay que indicar como mínimo su nombre, parámetros, el tipo que devuelve y su visibilidad. También se debe incluir una descripción del método que aparecerá en la documentación que se genere del proyecto.

Existen un caso particular de método, el **constructor** de la clase, que tiene como característica que no devuelve ningún valor. El constructor tiene el mismo nombre de la clase y se usa para ejecutar las acciones necesarias cuando se instancia un objeto de la clase. Cuando haya que destruir el objeto se podrá utilizar una función para ejecutar las operaciones necesarias cuando el objeto deje de existir, que dependerán del lenguaje que se utilice.

5. Relaciones entre clases

Una relación es una conexión entre dos clases que incluimos en el diagrama cuando aparece algún tipo de relación entre ellas en el dominio del problema.
Se representan como una línea continua. Los mensajes "navegan" por las relaciones entre clases, es decir, los mensajes se envían entre objetos de clases relacionadas, normalmente en ambas direcciones, aunque a veces la definición del problema hace necesario que se navegue en una sola dirección, entonces la línea finaliza en punta de flecha.
Las relaciones se caracterizan por su cardinalidad, que representa cuantos objetos de una clase se pueden involucrar en la relación, y pueden ser:

De herencia.
De composición.
De agregación.

6. Relación de herencia

La **herencia** es una propiedad que permite a los objetos ser construidos a partir de otros objetos, es decir, la capacidad de un objeto para utilizar estructuras de datos y métodos presentes en sus antepasados.

El objetivo principal de la herencia es la *reutilización*, poder utilizar código desarrollado con anterioridad. La herencia supone una clase base y una jerarquía de clases que contiene las clases propio código especial y datos, incluso cambiar aquellos elementos de la clase base que

necesitan ser diferentes, es por esto que los atributos, métodos y relaciones de una clase se muestran en el nivel más alto de la jerarquía en el que son aplicables.

Tipos:

. **Herencia simple**: Una clase puede tener sólo un ascendente. Es decir una subclase puede heredar datos y métodos de una única clase base.
. **Herencia múltiple**: Una clase puede tener más de un ascendente inmediato, adquirir datos y métodos de más de una clase.

Representación:

En el diagrama de clases se representa como una asociación en la que el extremo de la clase base tiene un triángulo.

7. Agregación y composición

Muchas veces una determinada entidad existe como un conjunto de otras entidades. En este tipo de relaciones un objeto componente se integra en un objeto compuesto. La orientación a objetos recoge este tipo de relaciones como dos conceptos: la agregación y la composición.

La agregación es una asociación binaria que representa una relación todo-parte (pertenece a, tiene un, es parte de). Los elementos parte pueden existir sin el elemento contenedor y no son propiedad suya. Por ejemplo, un centro comercial tiene clientes o un equipo tiene unos miembros. El tiempo de vida de los objetos no tiene porqué coincidir.
En el siguiente caso, tenemos un ordenador que se compone de piezas sueltas que pueden ser sustituidas y que tienen entidad por sí mismas, por lo que se representa mediante relaciones de agregación. Utilizamos la agregación porque es posible que una caja, ratón o teclado o una memoria RAM existan con independencia de que pertenezcan a un ordenador o no.

La composición es una agregación fuerte en la que una instancia 'parte' está relacionada, como máximo, con una instancia 'todo' en un momento dado, de forma que cuando un objeto 'todo' es eliminado, también son eliminados sus objetos 'parte'. Por ejemplo: un rectángulo tiene cuatro vértices, un centro comercial está organizado mediante un conjunto de secciones de venta...
Para modelar la estructura de un ciclo formativo vamos a usar las clases Módulo, Competencia y Ciclo que representan lo que se puede estudiar en Formación Profesional y su estructura lógica. Un ciclo formativo se compone de una serie de competencias que se le acreditan cuando supera uno o varios módulos formativos.

Dado que si eliminamos el ciclo las competencias no tienen sentido, y lo mismo ocurre con los módulos hemos usado relaciones de composición. Si los módulos o competencias pudieran seguir existiendo sin su contenedor habríamos utilizado relaciones de agregación.

Estas relaciones se representan con un rombo en el extremo de la entidad contenedora. En el caso de la agregación es de color blanco y para la composición negro. Como en toda relación hay que indicar la cardinalidad.

8. Atributos de enlace

Es posible que tengamos alguna relación en la que sea necesario añadir algún tipo de información que la compete de alguna manera. Cuando esto ocurre podemos añadir atributos a la relación.

Práctica

9. Actividad

- **Crear un diagrama de clases nuevo en Visual Paradigm UML que incluya su nombre y su descripción.** Para crear un diagrama de clases en VP-UML seleccionamos **Archivo >> Nuevo Diagrama** y seleccionamos Diagrama de clases. También podemos acceder al Navegador de diagramas, que se encuentra en el panel de la izquierda y en diagramas de clases hacer clic con el botón secundario y Seleccionar Nuevo diagrama de clases. Cuando generamos un diagrama nuevo tenemos que indicar su nombre y una descripción. Esto es importante para la generación de la documentación posterior.
  Cuando creamos un diagrama nuevo aparece en blanco en el panel central de la aplicación. Si es necesario cambiar sus propiedades podemos hacerlo seleccionándolo en el Navegador de diagramas, a través de la opción "Abrir <nombre> Specification" del menú contextual. También podemos abrir el menú contextual, haciendo clic con el botón derecho del ratón sobre el panel central de la aplicación.
- **Crear una clase nueva en el diagrama de clases del punto anterior.** Cuando generamos un diagrama nuevo aparece un panel con los elementos que podemos añadir al diagrama. Si hacemos clic sobre el icono que genera una clase nueva y a continuación sobre el lienzo aparecerá un cuadro para definir el nombre de la nueva clase. Posteriormente, cuando la clase esté creada si hacemos clic con el botón secundario sobre la clase y seleccionamos "Abrir Especificación" podremos añadir atributos y métodos. Al crear una clase no es obligatorio definir nombre, atributos y métodos.
- **Crear una clase de nombre "Módulo" y que tenga tres atributos:**
  - **Nombre, de tipo string.**
  - **Duración de tipo Int.**
  - **Contenidos de tipo string.**

  Crear la clase como hemos visto en el punto anterior y modificar su nombre a "Módulo". Para añadir un atributo a una clase basta con seleccionar Añadir atributo del menú contextual y escribir su nombre. Si queremos añadir más información podemos hacerlo desde la especificación de la clase en la pestaña Atributos, en la imagen vemos la especificación de una clase llamada Módulo, y de su atributo Contenidos para el que se ha establecido su tipo (string) y su descripción. Por defecto la visibilidad de los atributos es privado y no se cambia a menos que sea necesario. Tenemos la posibilidad de añadir, desde el menú contextual de la clase, con el atributo seleccionado dos métodos llamados **getter** y **setter** que se utilizan para leer y establecer el valor del atributo cuando el atributo no es calculado, con la creación de estos métodos se contribuye al encapsulamiento y la ocultación de los atributos.
- **Añadir a la clase creada anteriormente los métodos:**

- **Matricula(alumno:Alumno):void**
- **asignarDuración(duración:int):void**

El método más directo para crear un método es en el menú contextual seleccionar "Añadir operación" y escribir la signatura del método:
+nombre(<lista_parámetros>) : tipo_devuelto

También se puede añadir desde la especificación de la clase en la pestaña Operations.

En la imagen vemos la especificación de la clase y del método "asignarDuración" que asigna el número de horas del módulo.

El signo + en la signatura del método indica que es público.

- **Crea una clase nueva llamada Alumno y establece una relación de asociación con el nombre "matrícula" entre esta y la clase Módulo.**

  Creamos la clase como hemos visto en puntos anteriores. Para crear una relación utilizamos el elemento asociación de la paleta o bien el icono **Association >> Class** del menú contextual de la clase. Otra forma consiste en hacer clic sobre la clase **Alumno**, seleccionar **Association >> Class** y estirar la línea hasta la clase **Módulo**, aparecerá un recuadro para nombrar la relación.

  Es posible establecer relaciones unarias de una clase consigo misma. En el ejemplo se ha rellenado en la especificación de la relación los roles y la multiplicidad.

  - Para obtener las relaciones de un diagrama nos basamos en la descripción de los requisitos del dominio, pero, ¿se pueden crear relaciones en el diagrama que no aparezcan especificadas en la lista de requisitos del problema? Hay veces en que la información se conoce aunque no aparezca en la descripción de los requisitos. Es información inferida.
  - **Establece la cardinalidad de la relación que has creado en el punto anterior para indicar que un alumno debe estar matriculado en al menos un módulo, o varios y que para cada módulo se puede tener ningún alumno, uno o varios.**

    Si queremos establecer la **cardinalidad** abrimos la especificación de la relación y establecemos el apartado Multiplicidad a alguno de los valores que indica, si necesitamos utilizar algún valor concreto también podemos escribirlo nosotros mismos. En el caso que nos ocupa seleccionaremos la cardinalidad 0..* para los alumnos y 1..* para los módulos.

- En nuestro diagrama tenemos Alumnos y Profesores. Aún no hemos hablado de su definición y estructura, pero en nuestro sistema tanto un alumno como un profesor tienen unas características comunes como el nombre, la fecha de nacimiento o el correo electrónico por el hecho de ser personas:
  Transforma este diagrama para hacer uso de la herencia añadiendo una clase "Persona".

  Podemos utilizar la relación de herencia para crear una clase nueva que se llame Persona y que recoja las características comunes de profesor y alumno. Persona será la **clase base** y Profesor y Alumno las **clases derivadas**.

    - Como los atributos Nombre, FechaNacimiento y correoElectronico se heredan de la clase base no hace falta que aparezcan en las clases derivadas, por lo que las hemos eliminado. Después podemos añadir atributos o métodos propios a las clases derivadas. La relación se añade de igual manera que una relación de asociación, pero seleccionando la opción Generalization.
    - Ejemplo: He creado una clase persona cuyos atributos son Nombre, fechaContratación y numeroCuenta. De esta clase derivan por herencia la clase Empleado y JefeDepartamento. ¿Cómo debe declararse un método en la clase Persona que se llame CalculaAntiguedad que se usa sólo para calcular el sueldo de los empleados y jefes de departamento? Público, privado, protegido o paquete.
- Cuando un alumno se matricula de un módulo es preciso especificar el curso al que pertenece la matrícula, las notas obtenidas en el examen y la tarea y la calificación final obtenida. Estas características no pertenecen totalmente al alumno ni al módulo sino a la relación específica que se crea entre ellos, que además será diferente si cambia el alumno o el módulo. Añade estos atributos al enlace entre Alumno y Módulo. Para modelar esto en Visual Paradigm creamos una clase nueva (Matrícula) junto a Alumno y Módulo, y la unimos a la relación utilizando el icono de la paleta "Association class".

Entrega

El alumno deberá preparar una presentación de la tarea para exponer en clase con la elaboración de la tarea.