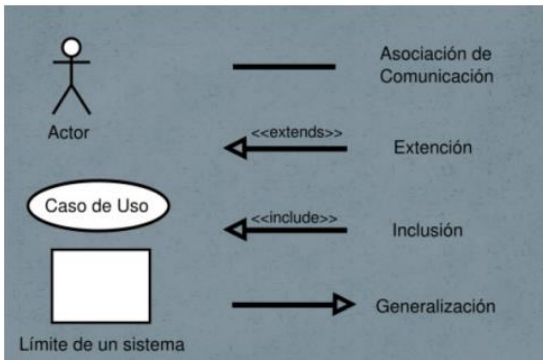
TEMA 4 Y 5: *Ingeniería del Software y Programación Orientada a Objetos.*

TEMA 4:

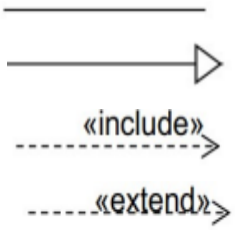
4.1) DIAGRAMA DE USOS:

- Componentes:



4.2) RELACIONES DE LOS CASOS DE USO:

- ***Asociación***: Línea de comunicación entre un actor y un caso de uso en el que participa.
- ***Generalización***: Relación entre un caso de uso general y un caso de uso más específico, que hereda y añade propiedades al caso de uso base.
- ***Inclusión***: Inserción de comportamiento adicional en un caso de uso base, describe explícitamente la inserción.
- ***Extensión***: Inserción de comportamiento adicional en un caso de uso base que no tiene conocimiento sobre él.
-



4.3) FASE EN EL DESARROLLO DE UNA APLICAIÓN:

1.Analisis de Requisitos: requisitos funcionales y no funcionales del sistema.

2.Diseño: Se divide en partes y se determina la función de cada una.

3.Implementación/Codificación: Se elige el lenguaje de programación, se construye el software de acuerdo con el diseño.

4.Prueba /Testing: Son programas para detectar errores.

5.Documentación: se documenta todas las etapas y se guarda información.

6.Explotación/Despliegue: configuramos, instalamos e probamos la aplicación.

7.Mantenimiento: Se mantiene contacto con el cliente para posibles cambios.

4.4) ANALISIS DE REQUISITOS Y CASOS DE USOS:

- ***Funcionales:*** Explican la funcionalidad que necesita tener el sistema.
- ***No Funcionales:*** Son aquellos que no afectan a una funcionalidad, sino a cómo esa funcionalidad debe ofrecerse a los usuarios.
- ***Requisitos de datos:*** Son las entidades que el sistema tendrá que almacenar. Por ejemplo, entidad CLIENTE con atributos CÓDIGO, DESCRIPCIÓN, etc.

4.5) CASO DE USO:

Tiene los siguientes apartados:

- **Descripción:** una descripción reducida del requisito.
- **Actores:** que tipo de usuarios podrán utilizar esta funcionalidad.
- **Precondición:** como está el sistema cuando empieza el caso de uso.
- **Flujo normal:** descripción paso a paso de lo que haga el usuario.
- **Excepciones:** cómo se comporta el sistema si durante el flujo normal se da alguna circunstancia anormal, como un error.
- **Postcondición:** cómo queda el sistema al terminar el caso de uso.

4.6) DISEÑO:

Esta etapa se compone por:

- Entidades y relaciones de las BBDD.
- Selección del lenguaje de programación.
- Selección del Sistema Gestor de Base de Datos.
- Otros elementos.

4.7) CODIFICACIÓN:

Las características son:

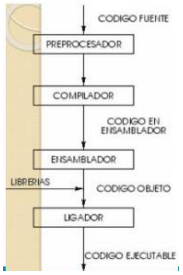
- **Modularidad:** está dividido en trozos pequeños.
- **Corrección:** que haga realmente lo que pida.
- **Fácil de leer:** facilita al usuario.
- **Eficiencia:** que haga un buen uso de los recursos.
- **Portabilidad:** que se pueda implementar en cualquier equipo.

Pasa por diferentes estados:

- Código fuente.

- Código objeto.

- Código ejecutable.



- **Código fuente:** es el escrito por los programadores en algún editor de texto (lenguaje de programación de alto nivel).
- **Código objeto:** es un código binario resultado de compilar el código fuente, se usa utilizando un compilador.
- **Código ejecutable:** es el código binario de enlazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias.

4.8) MANTENIMIENTO:

Los tipos de cambios que hacen necesario el mantenimiento del software son los siguientes:

- **Perfectivos:** mejorar funcionalidad del software.
- **Evolutivos:** cliente tiene nuevas necesidades.
- **Adaptativos:** modificaciones, actualizaciones... para adaptarse a las nuevas tendencias del mercado.
- **Correctivos:** la aplicación tendrá errores en el fututo.

TEMA 5:

5.1) CONCEPTO DE ORIENTACIÓN A OBJETOS:

- **Abstracción:** permite capturar las características y comportamiento similares de un conjunto de objetos con el objetivo de darles una descripción formal.
- **Encapsulación:** reúne los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción.
- **Modularidad:** permite subdividir una aplicación en partes más pequeñas, cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.
- **Principio de ocultación:** aísla las propiedades de un objeto contra su modificación.
- **Polimorfismo:** consiste en reunir bajo el mismo nombre comportamientos diferentes.
- **Recolección de basura:** se encarga de destruir automáticamente los objetos, y desvincular su memoria asociada.

5.2) VISIBILIDAD:

Se divide en dos partes:

- **Interfaz:** captura la visión externa de una clase.
- **Implementación:** comprende cómo se representa la abstracción, así como los mecanismos s que conducen al comportamiento deseado.

El POO tiene tres niveles:

- **Público:** accesibles desde cualquier lugar del código.
- **Protegido:** son accesibles solo por la clase y sus subclases.
- **Privado:** son accesibles solo dentro de la clase.

5.3) INSTANCIACIÓN:

Un objeto se define por:
-Su estado: es la corrección de los atributos definidos en la clase a un valor concreto.

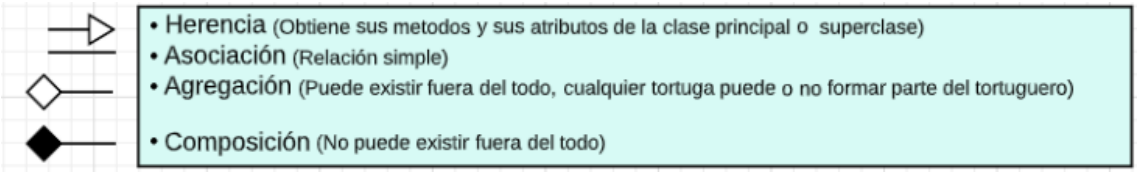
-Su comportamiento: definido por los métodos públicos de su clase.

-Su tiempo de vida: intervalo de tiempo a lo largo del programa en el que el objeto existe.

5.4) NOTACIÓN UML:

Un diagrama de clase consta de:

- Nombre de la clase, se escribe en el interior del rectángulo.
- Los atributos de la clase, que se escriben debajo del nombre de la clase.
- Los métodos de la clase, que se escriben debajo de los atributos.
- Las relaciones entre las clases, que se representan mediante flechas con puntas en las clases relacionadas.



El UML, es un sistema como una colección de modelos que describen sus diferentes perspectivas.

Un diagrama UML, se compone por cuatro tipos de elementos:

- **Estructuras:** son los nodos del grafo y definen el tipo de diagrama.
- **Relaciones:** son los arcos del grafo que se establecen entre los elementos estructurales.
- **Notas:** se representan como un cuadro donde podemos escribir comentarios que nos ayuden a entender algún concepto.
- **Agrupaciones:** se utilizan cuando modelamos sistemas de grandes para facilitar su desarrollo por bloques.
- **Diagramas estructurales:** representan la visión estática del sistema.
- **Diagramas de comportamiento:** muestran la conducta en tiempo de ejecución del sistema.

5.5) HERRAMIENTAS DE ELABORACIÓN DE DIAGRAMA DE CLASES:

- **Rational Systems Developer de IBM:** Herramienta propietaria que permite el desarrollo de proyectos software basados en la metodología UML.

- **Visual Paradigm for UML (VP-UML):** Incluye una versión para uso no comercial que se distribuye libremente sin más que registrarse para obtener un archivo de licencia.

- **ArgoUML:** se distribuye bajo licencia Eclipse. Soporta los diagramas de UML 1.4, y genera código para java y C++.