PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL FORMATO GUÍA DE APRENDIZAJE

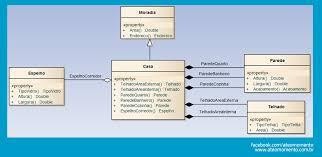
# IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

* + **Denominación del Programa de Formación:** ANÁLISIS Y DESARROLLO DE SOFTWARE

# Código del Programa de Formación: 228118

* + **Nombre del Proyecto**: Construcción de software integrador de tecnologías orientadas a servicios.
  + **Fase del Proyecto:** PLANEACION
  + **Actividad de Proyecto:** Determinar La Estructura Lógica Y Tecnológica Del Sistema
  + **Competencia: 220501095** modelado de los artefactos del software
  + **Resultados de Aprendizaje Alcanzar**: **220501095 01** Elaborar los artefactos de diseño del software siguiendo las prácticas de la metodología seleccionada.
  + **Duración de la Guía:** 30 horas

# PRESENTACIÓN

El diagrama de clase entra en la categoría de los diagramas estructurales, y es sin duda el más común de ellos. Son diagramas que permiten describir la arquitectura de un sistema con bastante detalle. Es muy recomendable tener un diagrama de clase antes de empezar a escribir código para una base de datos o una aplicación.

Fuente: https://[www.ateomomento.com.br/uml-diagrama-de-classes](http://www.ateomomento.com.br/uml-diagrama-de-classes)

Escribiendo un diagrama de clase, podemos describir cómo se relacionan los elementos de un sistema, qué atributos les caracterizan. Después de haber escrito diagramas de casos de uso, tendremos las cosas lo suficientemente despejadas como para poder ir diseñando la arquitectura del sistema.

A través del desarrollo de esta guía, el aprendiz reconocerá los elementos que componen un diagrama de

# FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

* + **Descripción de la(s) Actividad(es)**

Durante esta actividad los aprendices deben estar prestos a la contextualización del tema que brinda el instructor. Las actividades se realizan de forma presencial en la formación con la orientación del instructor y de manera autónoma con la revisión del material de apoyo, referencias bibliográficas y con la elaboración de actividades.

# Ambiente Requerido

El ambiente de aprendizaje debe estar conformado por:

22 equipos con los requerimientos mínimos:

* Sistema operativo: Windows
* Disco Duro: 400 GB
* Ram: 2GB como mínimo
* Procesador: Intel Core 2Duo de 2,66 Mhz
* Mesas y sillas

1 servidor de aplicaciones Conexión a Internet permanente

# Materiales

Computadores de escritorio y portátiles con acceso a internet, software de aplicación para realizar informes y visualizar material digital.

# Actividades de Reflexión Inicial

El UML (Lenguaje Unificado de Modelado), es una de las herramientas más emocionantes en el mundo actual del desarrollo de sistemas. Esto se debe a que los creadores de sistemas generan diseños que capturen sus ideas en forma convencional y fácil de aprender para comunicarlas a otras personas. El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. “Los tres Amigos” como se apodaron estos tres grandes utilizaban metodologías diferentes para el diseño de software hasta que en los años 90 decidieron unirse y trabajar en conjunto en una solo metodología, el UML está compuesto de diversos elementos gráficos que se combinan para conformar diagramas, con las mismas finalidades que es presentar diversas perspectivas de un sistema a las cuales se conoce como modelo.

Fuente <https://www.ecured.cu/Diagrama_de_Clase>

El diagrama de clases hace parte de los diagramas UML y muestra la estructura estática del sistema que se está modelando.

¿Cada aprendiz debe analizar cuáles son las características esenciales de los objetos, el ser humano hace parte de los objetos para un sistema? Justifique su respuesta.

Las características importantes de los objetos del sistema desde la perspectiva de la programación orientada a objetos son las siguientes:

**Encapsulación:** los objetos encapsulan los datos y el comportamiento asociado en una sola entidad. Esto significa que los detalles internos del objeto están ocultos y solo son visibles las interfaces necesarias para interactuar con él. La encapsulación promueve el modularidad y la capacidad de mantenimiento del código al reducir las dependencias entre las diferentes partes del sistema.

**Abstracción:** los objetos representan entidades del mundo real o conceptos abstractos seleccionando y representando sus propiedades esenciales. Al abstraer solo lo esencial, los objetos simplifican la complejidad y permiten una comprensión más clara de la estructura y el comportamiento del sistema.

**Herencia:** La herencia le permite crear nuevas clases (objetos) basados en clases existentes, heredando sus propiedades y comportamiento. Esto facilita la reutilización del código y la organización jerárquica de las clases, lo que a su vez mejora la eficiencia y la legibilidad del sistema.

**Polimorfismo:** El polimorfismo permite que diferentes objetos respondan a la misma solicitud de una manera específica dependiendo de su tipo o contexto. Por lo tanto, el método se puede implementar en varias clases de diferentes maneras, lo que le da flexibilidad y adaptabilidad al sistema.

Una persona puede ser considerada parte de los objetos del sistema desde el momento que ingresa al sistema como usuario, en este caso las personas pueden ser representadas como objetos que tienen sus propias características, comportamiento y relaciones en el sistema. En este enfoque, los objetos representan entidades que pueden comunicarse y colaborar con otras partes del sistema, incluidas las personas, para lograr objetivos y resolver problemas específicos. Involucrar a las personas en los objetos del sistema es una forma de diseñar sistemas más intuitivos y centrados en el usuario, donde la interacción humana se considera una parte esencial del diseño y la funcionalidad.

# Diagrama Descripción generada automáticamenteActividades de Contextualización e Identificación de Conocimientos Necesarios para el Aprendizaje

* + 1. En parejas de trabajo los aprendices deben investigar para realizar una presentación que responda a los siguientes interrogantes:
       1. ¿Qué es una clase?
       2. ¿Qué elementos hacen parte de la clase, describa cada uno?

Seguidamente el instructor realiza una contextualización a los aprendices acerca de los interrogantes propuestos.

RTA:

¿Qué es una clase?

En la programación orientada a objetos (POO), una clase es una plantilla o un modelo para crear objetos. Representa un conjunto de atributos (variables) y métodos (funciones) que definen el comportamiento y las características de los objetos que se crearán a partir de ella. En otras palabras, una clase es una abstracción de un concepto o entidad particular, y los objetos son instancias específicas de esa clase.

Las clases se utilizan para organizar y estructurar el código de manera modular, lo que permite reutilizar el código y facilita la gestión de grandes programas. Cada objeto creado a partir de una clase tendrá sus propios valores únicos para los atributos, pero compartirá los mismos métodos definidos en la clase.

¿Qué elementos hacen parte de la clase, describa cada uno?

Una clase consta de varios elementos, cada uno de los cuales está destinado a definir y representar la estructura y el comportamiento de los objetos creados a partir de esta clase.

Atributos (o propiedades):

**Los atributos** son variables que describen las características o estados de los objetos que pertenecen a la clase. Representan los datos que contiene un objeto y definen sus características únicas. Los atributos son variables definidas dentro de una clase y tienen varios tipos de datos, como números enteros, cadenas, valores booleanos, etc.

**Métodos** (o funciones): Los métodos son funciones que definen el comportamiento de los objetos de clase. Los métodos se definen dentro de una clase y pueden acceder y manipular los atributos de la clase. Cada objeto creado a partir de la clase puede llamar a estos métodos para realizar ciertas acciones.

**Constructor:** Un constructor es un método de clase especial que se ejecuta automáticamente cuando se crea un nuevo objeto a partir de la clase. Su función principal es inicializar los atributos de un objeto con ciertos valores.

**Destructor:** Un destructor es otro método especial en una clase que se ejecuta automáticamente cuando un objeto se elimina o sale del alcance del programa. Su función es liberar los recursos utilizados por el objeto y realizar la limpieza antes de que el objeto deje de existir.

Visibilidad (pública, privada, protegida):En muchos lenguajes de programación, los modificadores de acceso como "público", "privado" y "protegido" se utilizan para determinar qué elementos son visibles desde el exterior y cuáles son internos de la clase.

**Herencia:** La herencia es un concepto clave en la POO que permite que una clase (llamada subclase o clase hija) herede atributos y métodos de otra clase (llamada superclase o clase padre). Esto permite la reutilización de código y la creación de jerarquías de clases. Una subclase puede extender o especializar la funcionalidad de la clase padre.

La Programación Orientada a Objetos supone un cambio en la concepción del mundo de desarrollo de software, introduciendo un mayor nivel de abstracción que permite mejorar las características del código ﬁnal. De manera muy básica, las aportaciones de este paradigma se pueden resumir en:

* Conceptos de clase y objeto, que proporcionan una abstracción del mundo centrada en los seres y no en los verbos.
* Los datos aparecen encapsulados dentro del concepto de clase. El acceso a los datos se produce de manera controlada e independiente de la representación final de los mismos. Como consecuencia, se facilita el mantenimiento y la evolución de los sistemas, al desaparecer las dependencias entre distintas partes del sistema.
* Mediante conceptos como la composición, herencia y polimorﬁsmo se consigue simplificar el desarrollo de sistemas. La composición y la herencia nos permiten construir clases a partir de otras clases, aumentando en gran medida la reutilización.
  + 1. Investigue, analice y conceptualice acerca de UML, consultando los siguientes temas:

1. ¿Qué se entiende por programación orientada a objetos?

Es un paradigma de programación que se basa en la organización del código en objetos, los cuales son entidades que representan conceptos del mundo real y cada objeto es una instancia de una clase.

1. ¿Cuál es la diferencia entre programación estructurada y programación orientada a objetos?

En la programación estructura el código se organiza en funciones o procedimientos que contiene una secuencia lógica de instrucciones. En la programación orientada a objetos el código se organiza en clases que representa entidades con atributos y métodos la programación estructurada se centra en la estructura del programa y como los datos fluyen entre las funciones utilizando estructuras de control como bubles y condicionales la de objetos se entra en los objetos y como interactúan entre si utilizando la encapsulación, herencia y el polimorfismo.

1. ¿Qué es una clase y cuáles son los elementos que la componen?

Es la plantilla o modelo que define la estructura y comportamiento de un objeto. Define los atributos y métodos que puede tener los objetos de esa clase.

La componen el nombre de la clase, atributos, métodos, herencia.

1. ¿Qué es un objeto?

Un objeto es una instancia concreta de una clase. Posee un estado interno (atributos)y puede realizar acciones (métodos).

1. ¿Qué es herencia, de un ejemplo?

Permite crear nuevas clases basadas en clases existentes, aprovechando la estructura y comportamiento de la clase base. O super clase la herencia permite la reutilización de código y la organización jerárquica de las clases. De ejemplo podríamos imaginar una tienda de productos electrónicos donde tendríamos una super clase que tenga todos los atributos y métodos que pueden tener en común esto productos como fecha de lanzamiento, precio y fabricante luego se generarían las sub clase que heredan este código y tendría ya sus datos propios como tamaño de la pantalla tipo de procesador memoria entre otros.

1. ¿Qué se entiende por encapsulación?

Es el concepto de ocultar los detalles internos de un objeto y proporcionar una interfaz para interactuar con el. Se utiliza para proteger los datos y los métodos internos de un objeto.

1. ¿Qué es polimorfismo?

Permite que objetos de diferentes clases respondan de manera diferente a la misma función o método permite tratar a objetos de diferentes clases de manera uniforme a través de una interfaz común.

1. ¿Explique los diferentes tipos de relaciones de un diagrama de clases UML: asociación, agregación, dependencia, composición…; con que línea se representa cada una?

* **Asociación:** Representa una relación entre dos clases donde una clase usa o interactúa con la otra. Se representa con una línea sólida que conecta las dos clases.
* Agregación: Representa una relación de "todo-parte" entre dos clases, donde una clase es parte de la otra clase. Se representa con una línea sólida que tiene un rombo en el extremo que apunta a la clase que contiene a la otra.
* **Composición**: Es similar a la agregación, pero en este caso la parte no puede existir sin el todo. Se representa con una línea sólida que tiene un rombo lleno en el extremo que apunta a la clase que contiene a la otra.
* **Dependencia:** Representa una relación donde una clase depende de otra clase para realizar alguna tarea o función. Se representa con una línea punteada que apunta desde la clase dependiente hacia la clase de la cual depende.
* **Herencia:** Representa una relación entre dos clases donde una clase (subclase) hereda los atributos y métodos de otra clase (superclase). Se representa con una línea sólida con una flecha que apunta desde la subclase hacia la superclase.

1. ¿Qué es cardinalidad de las relaciones y como se clasifican?

La cardinalidad describe la cantidad de ocurrencias o instancias que pueden estar relacionadas en un extremo de la relación con respecto a las instancias en el otro extremo.

Se clasifica en cuatro categorías principales

* + - Cardinalidad 1 a 1
    - Cardinalidad 1 a muchos
    - Cardinalidad muchos a 1
    - Cardinalidad muchos a muchos

# Actividades de Apropiación

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso. Fuente <https://www.ecured.cu/Diagrama_de_Clase>

Las clases se representan por rectángulos que muestran el nombre de la clase y opcionalmente el nombre de las operaciones y atributos. Los compartimientos se usan para dividir el nombre de la clase, atributos y operaciones. Adicionalmente las restricciones, valores iniciales y parámetros se pueden asignar a clases.

La notación que precede el nombre del atributo u operación indica la visibilidad del elemento, si se usa el símbolo + el atributo y la operación tienen un nivel público de visibilidad, si se usa un símbolo – el atributo u operación es privado. Además, el símbolo # permite definir una operación o atributo como protegido y el símbolo ~ indica la visibilidad del paquete.

Fuente <https://www.ecured.cu/Diagrama_de_Clase>

En parejas de trabajo explora la siguiente terminología de acuerdo con las instrucciones del instructor:

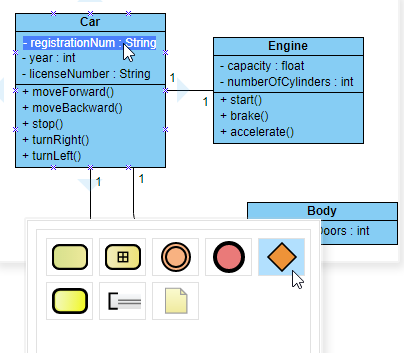
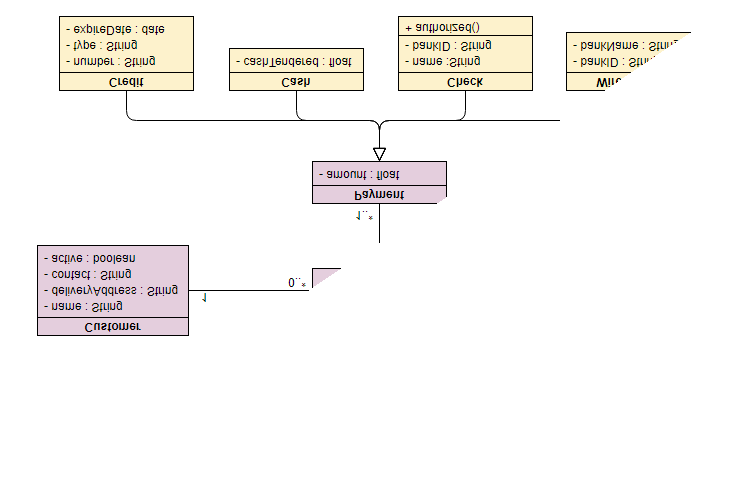
¿Qué significa instancia, atributo, método, visibilidad, herencia, asociación, agregación, composición, multiplicidad en diagramas de clase?

* Investigue tres ejemplos y detalle su aplicación

# Actividades de Transferencia del Conocimiento

Visual Paradigm Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional, etc). Fue diseñado para una ámplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Esta herramienta tiene disponibilidad en múltiples plataformas (Windows, Linux), Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación, Modelo y código que permanece sincronizado en todo el ciclo de desarrollo, Disponibilidad de múltiples





Fuente <https://www.ecured.cu/Visual_Paradigm>

La anterior es una de las múltiples herramientas de software que encontramos para construir diagramas de clase, como actividad en parejas de acuerdo a las indicaciones del instructor desarrollar los diagramas propuestos para los casos de estudio y en los equipos de trabajo del **proyecto formativo**, de acuerdo a los casos uso planteados, modelo relacional notación crow´s foot construir los diagramas de clase pertinentes.

# ACTIVIDADES DE EVALUACIÓN

|  |  |  |
| --- | --- | --- |
| **Evidencias de Aprendizaje** | **Criterios de Evaluación** | **Técnicas e Instrumentos de Evaluación** |
| **Evidencias de Conocimiento:**  Prueba en línea sobre conceptos fundamentales de diagramas UML clases. | Identifica las principales características para la construcción del diagrama de clases. | Lista chequeo Cuestionario en línea  GFPI-F-135 V01 |

|  |  |  |
| --- | --- | --- |
| **Evidencias de Desempeño**  Desarrollo de Talleres y actividades en la formación.  Sustentación de los diagramas elaborados.  Valoración actitudinal frente al proceso.  **Evidencias de Producto:**  Desarrollo del diagrama de clases del proyecto formativo.  Taller diagrama de clases | Participa activamente con las actividades propuestas en los talleres.  Entrega las actividades con pertinencia, vigencia, autenticidad y calidad.  Elabora los diagramas de clases dando uso a los elementos y las relaciones que le corresponde, utilizando de manera correcta los recursos de la herramienta. | Observación directa Lista de chequeo  Cuestionario Lista de chequeo |

1. **GLOSARIO DE TÉRMINOS**

**Diagrama de Clase:** El diagrama de clase representa clases, sus partes y la forma en la que las clases de los objetos están relacionados con otro.

**Clases:** son descripciones de un juego de objetos con características, comportamiento, relaciones y semánticas comunes. Se usan para modelar un juego de conceptos o entidades.

**Atributos:** Son descripciones de características, se usan para modelar información asociada con una entidad, es un dato importante que contiene valores de esa clase (campos-variables-propiedades).

**Métodos Operaciones o Funciones:** Son descripciones del comportamiento, se usan para modelar los servicios u operaciones asociados con una entidad, esto es, lo que una entidad puede hacer.

# REFERENTES BILBIOGRÁFICOS

Senn, James A.. Análisis y diseño de sistemas de información, McGraw-Hill Interamericana, 1992. ProQuest Ebook Central, [https://ebookcentral-proquest-](https://ebookcentral-proquest-com.bdigital.sena.edu.co/lib/senavirtualsp/detail.action?docID=3194990) [com.bdigital.sena.edu.co/lib/senavirtualsp/detail.action?docID=3194990](https://ebookcentral-proquest-com.bdigital.sena.edu.co/lib/senavirtualsp/detail.action?docID=3194990).

Kendall, Kenneth y Julie. Análisis y Diseño de Sistemas. Pearson Educación. 6ª edición. México 2005.

Kendall, K. E., & Kendall, J. E. (2005). Análisis y Diseño de Sistemas (Sexta ed.). México: Pearson Educación.

# CONTROL DEL DOCUMENTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** |
| **Autor (es)** | Paola Tatiana Tovar Yaqueline Chavarro | Instructor | Centro de Electricidad, Electrónica y Telecomunicaciones | Julio 2019 |

1. **CONTROL DE CAMBIOS** (diligenciar únicamente si realiza ajustes a la guía)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** | **Razón del Cambio** |
| **Autor (es)** | Javier Yara | instructor | CEET | **Febrero 2023** | Cambio de  formato y  actualización de contenido |