**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Desarrollo de aplicaciones móviles |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501096- Desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo. | RESULTADOS DE APRENDIZAJE | 220501096-1 Modelar soluciones de softwarede acuerdo con paradigma de la programación orientada a objetos |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 11 |
| NOMBRE DEL COMPONENTE FORMATIVO | Diseño del modelo conceptual bajo el paradigma orientado a objetos |
| BREVE DESCRIPCIÓN | En este componente formativo podrá abordar saberes relacionados con el Lenguaje de Modelado Unificado, diagramas de clase y el uso de herramientas Case que le servirán para crear modelos fundamentales para visualizar, especificar y construir *software*. Así mismo, se desarrollarán temáticas relacionadas a la implementación en el lenguaje de programación Java, de clases, objetos, atributos, constructores, métodos, herencia y relaciones. |
| PALABRAS CLAVE | CASE, clases, diagramas, objetos, UML |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 6 - VENTAS Y SERVICIOS |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS**

**Introducción**

**1. Introducción a UML**

**2. Diagrama de clase**

2.1 Control de acceso

2.2 Asociaciones

2.3 Herencia

2.4 Agregación

2.5 Composición

**3. Herramientas CASE**

**4. Características del lenguaje de programación orientada a objetos**

**5. Implementación de clases y objetos**

5.1 Atributos y métodos de la clase

5.2 Constructores y destructores

5.3 Métodos accesores y modificadores

5.4 Sobrecarga de métodos (*overload*)

**6. Comunicación entre clases**

6.1 Implementación de asociaciones

6.2. Implementación de composición/agregación

6.3 Implementación de herencia

**Síntesis**

1. **INTRODUCCIÓN**

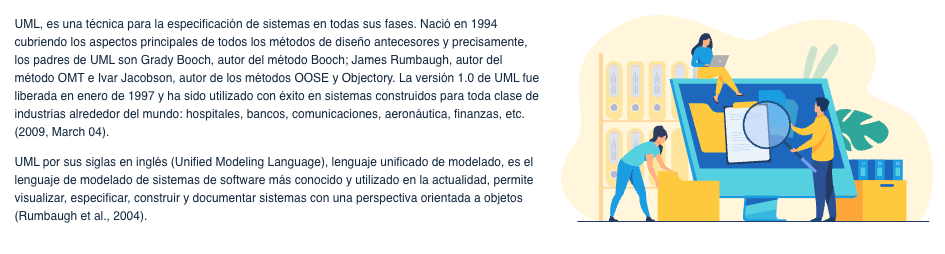
Para la construcción de *software* se hace evidente la importancia de la utilización de modelos ya que representan el aspecto y la dirección de una necesidad. Esa “necesidad” puede estar en un estado de desarrollo o en un estado de planeación. Los diseñadores del modelo deben realizar un levante de los requerimientos del producto y dichos requerimientos pueden incluir áreas tales como funcionalidad, *performance* y confiabilidad. El modelo se divide en un número de vistas, cada una de las cuales describe un aspecto específico del producto o sistema en construcción. Un modelo permite documentar la estructura y conducta de un sistema antes de que sea codificado. En el siguiente video se expone la importancia de la programación orientada a objetos (POO) para modelar soluciones de *software*.

|  |
| --- |
| **CF11\_Video\_Introducción** |

Este componente formativo, entonces, también se dará un recorrido a las características del lenguaje de programación orientada a objetos, las implementaciones de clases y objetos, y en su interior se reconocerán los atributos, métodos de las clases, los constructores y destructores, los métodos de accesores y modificación, junto con la sobrecarga de métodos. Esto permitirá conocer con la comunicación entre clases desde su implementación de asociaciones, composición / agregación y la implementación de herencia.

1. **DESARROLLO DE CONTENIDOS**

**1. Introducción a UML**

****

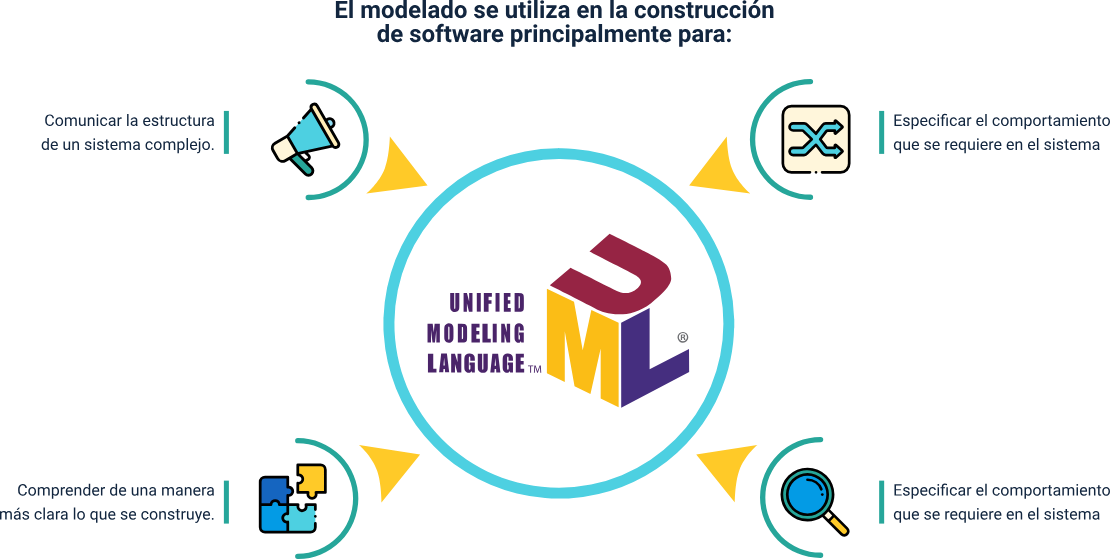
Reemplazar todo el texto así:

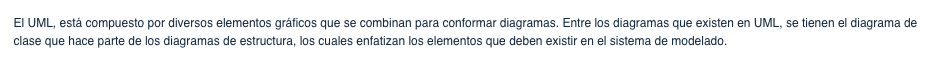
UML es una técnica para la especificación de sistemas en todas sus fases. Nació en 1994 cubriendo los aspectos principales de todos los métodos de diseño antecesores. Los padres de UML son Grady Booch, autor del método Booch, James Rumbaugh, autor del método OMT, e Ivar Jacobson, autor de los métodos OOSE y Objectory. La versión 1.0 de UML fue liberada en enero de 1997 y ha sido utilizada con éxito en sistemas construidos para toda clase de industrias alrededor del mundo: hospitales, bancos, comunicaciones, aeronáutica, finanzas, etc. (Holt, R., 2009).

UML por sus siglas en inglés (Unified Modeling Language), lenguaje unificado de modelado, es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad; permite visualizar, especificar, construir y documentar sistemas con una perspectiva orientada a objetos (Rumbaugh et al., 2004).

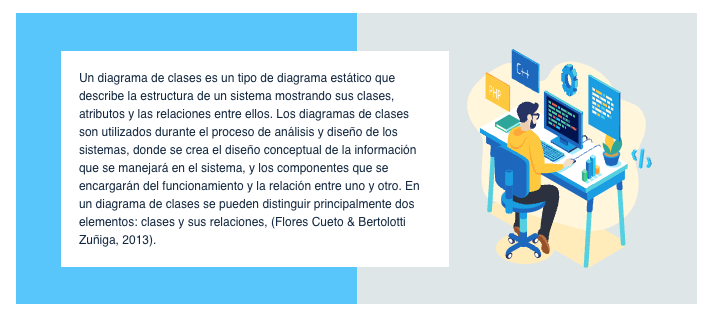
**Figura 1**

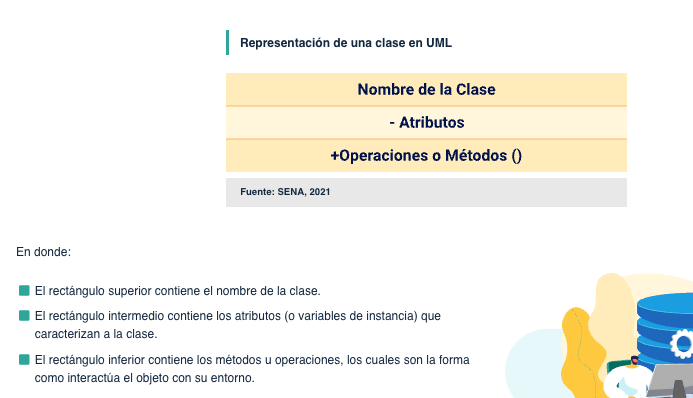
*Usos del modelado*

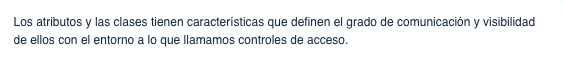




**2. Diagrama de clase**

****

****



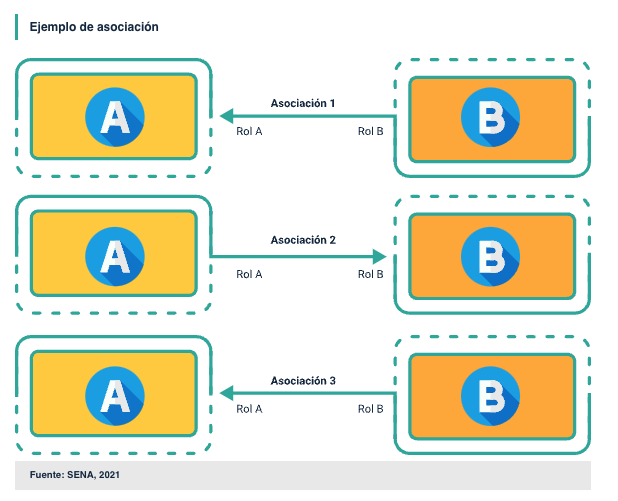
**2.1 Control de acceso**

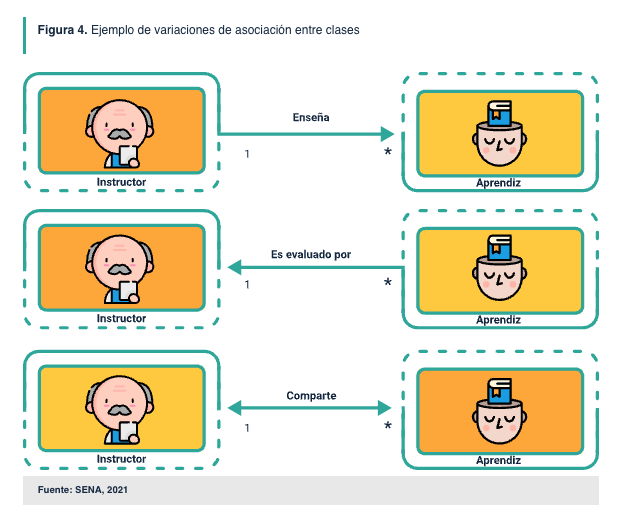


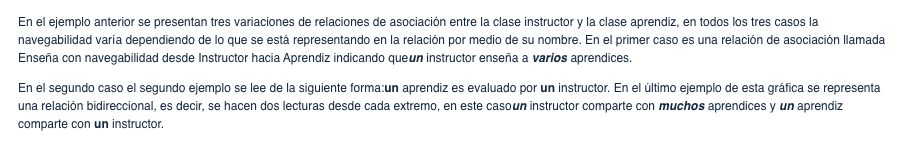
**2.2 Asociaciones**

**Figura 3**

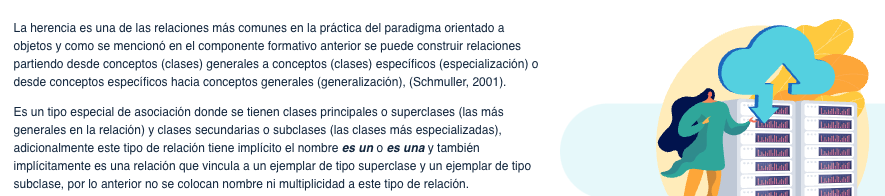
*Ejemplo de asociación*

****

****



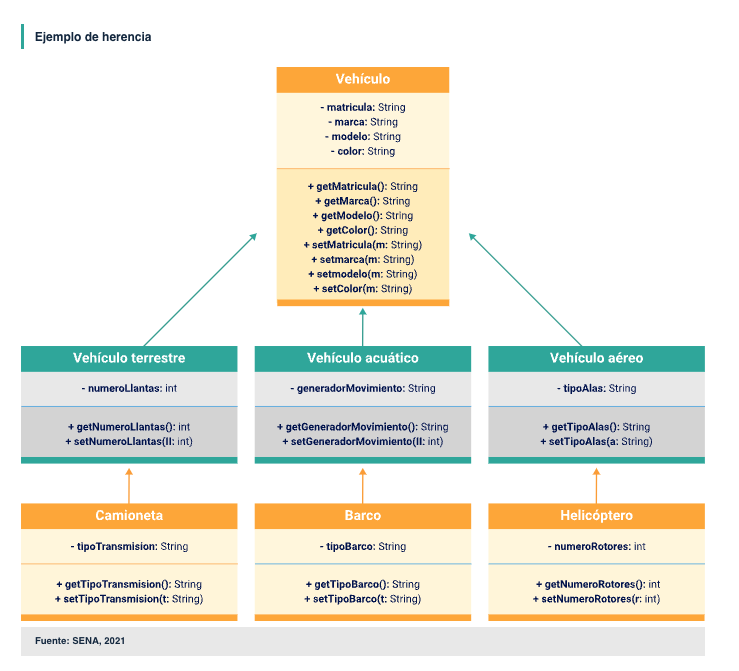
**2.3 Herencia**

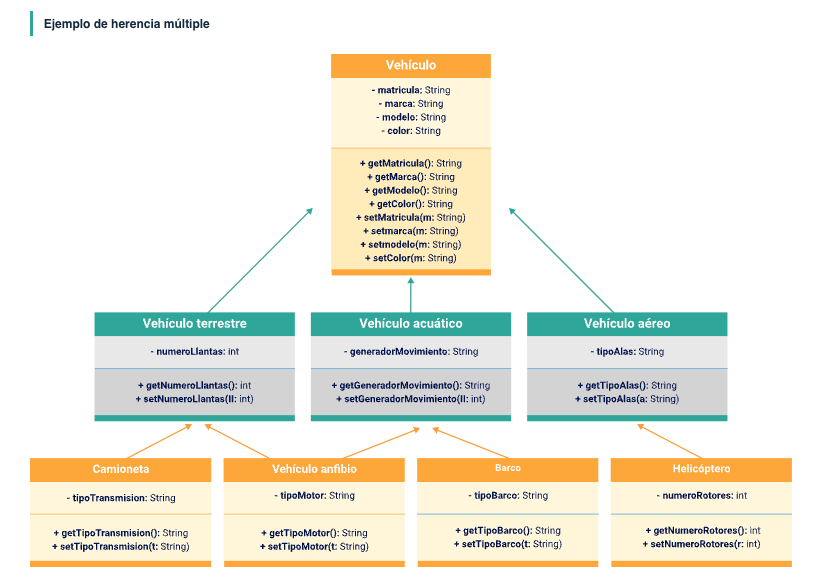


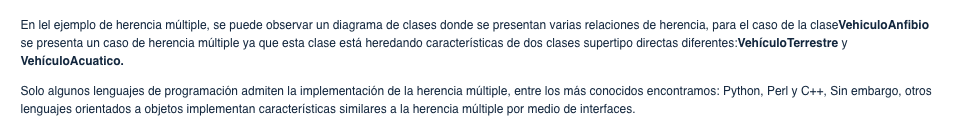
Reemplazar todo el texto:

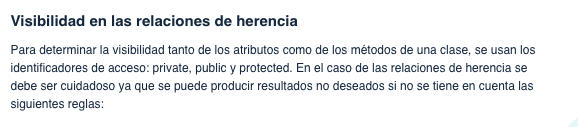
La herencia es una de las relaciones más comunes en la práctica del paradigma orientado a objetos y, como se mencionó en el componente formativo anterior, se puede construir relaciones partiendo desde conceptos (clases) generales a conceptos (clases) específicas (especialización) o desde conceptos específicos hacia conceptos generales (generalización) (Schmuller, 2001).

Es un tipo especial de asociación donde se tienen clases principales o superclases (las más generales en la relación) y clases secundarias o subclases (las clases más especializadas). Adicionalmente, este tipo de relación tiene implícito el nombre ***es un*** o ***es una*** y también, implícitamente, es una relación que vincula a un ejemplar de tipo superclase y un ejemplar de tipo subclase; por lo anterior, no se colocan nombre ni multiplicidad a este tipo de relación.

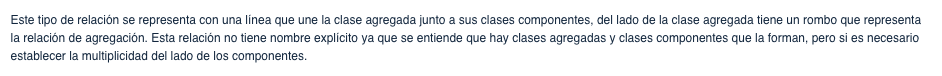


****

****

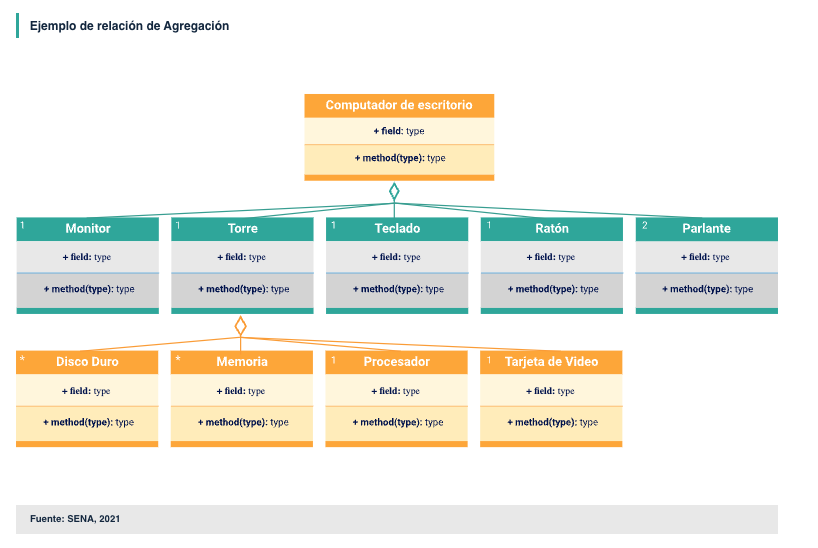


**2.4 Agregación**

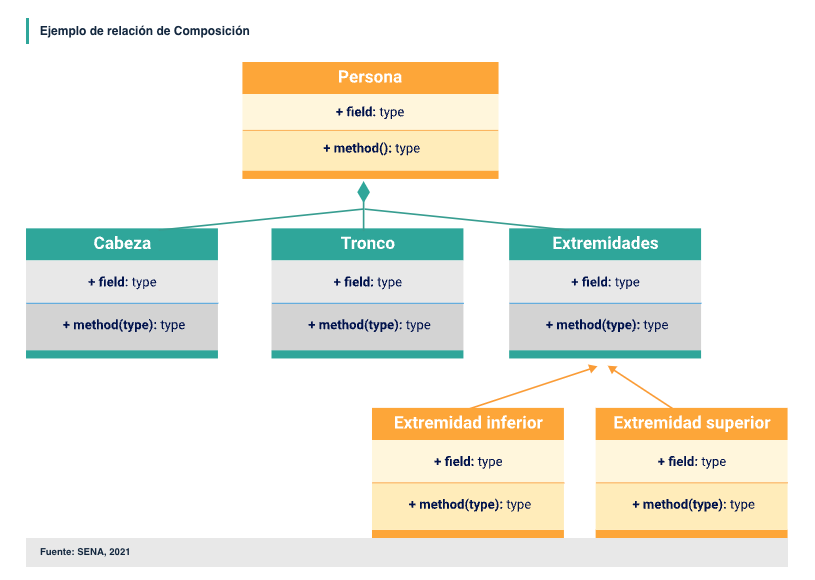


Reemplazar texto, así:

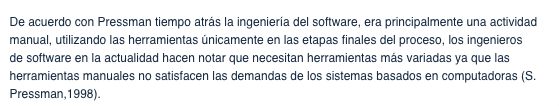
Este tipo de relación se representa con una línea que une la clase agregada junto a sus clases componentes; ahora, del lado de la clase agregada se tiene un rombo que representa la relación de agregación. Esta relación no tiene nombre explícito, ya que se entiende que hay clases agregadas y clases componentes que la forman. Es necesario establecer la multiplicidad del lado de los componentes.



**2.5 Composición**

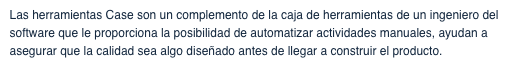
****

**3. Herramientas CASE**

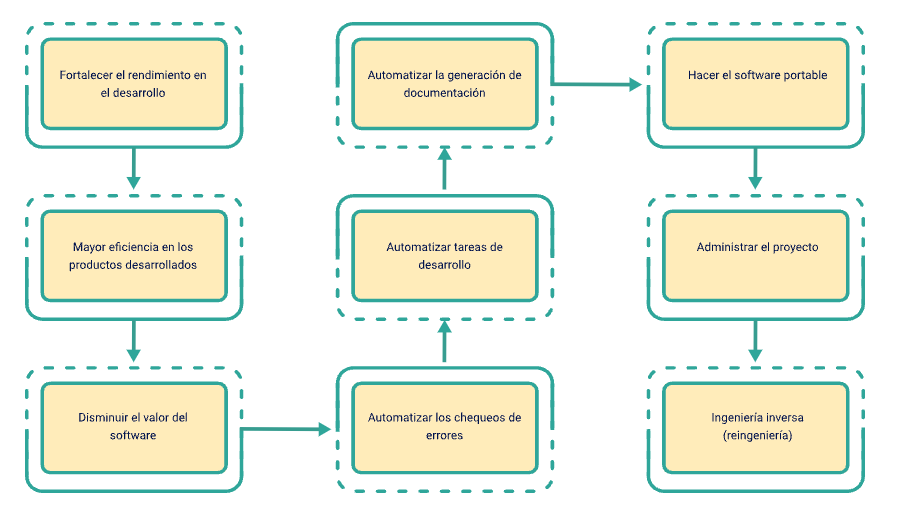
****

Reemplazar texto, así:

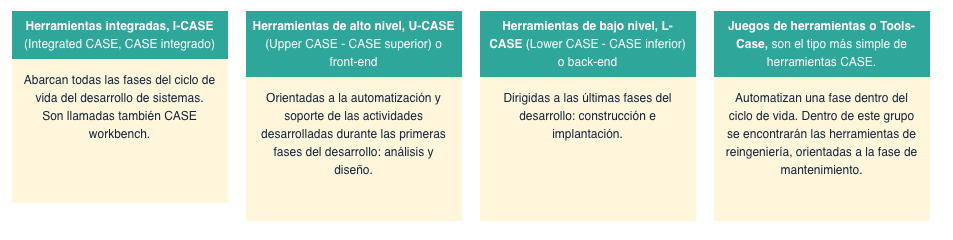
De acuerdo con Pressman (1998), tiempo atrás la ingeniería del *software* era principalmente una actividad manual, utilizando las herramientas únicamente en las etapas finales del proceso. Los ingenieros de *software* en la actualidad hacen notar que necesitan herramientas más variadas, ya que las herramientas manuales no satisfacen las demandas de los sistemas basados en computadoras (Pressman,1998).









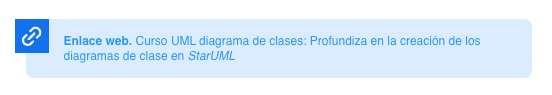




Reemplazar lo anterior por este bloque destacado:

Para profundizar en las herramientas CASE, se sugiere revisar el material complementario sobre esta temática.

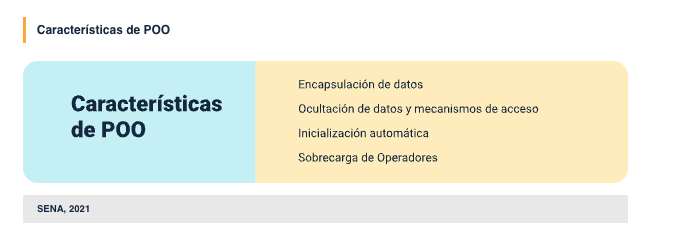




Reemplazar lo anterior por este bloque destacado:

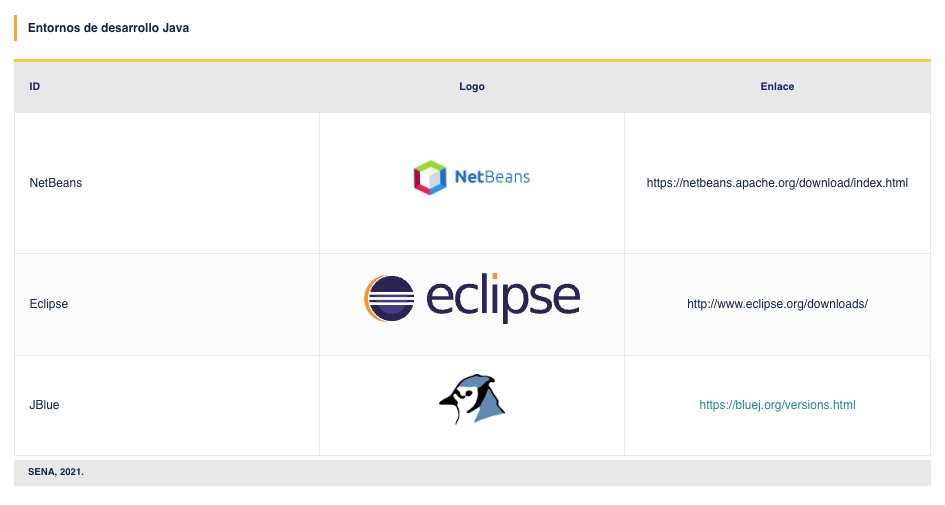
Para profundizar en la creación de los diagramas de clase en *StarUML*, se sugiere revisar en el material complementario el video sobre esta temática.

**4. Características del lenguaje de programación orientada a objetos**

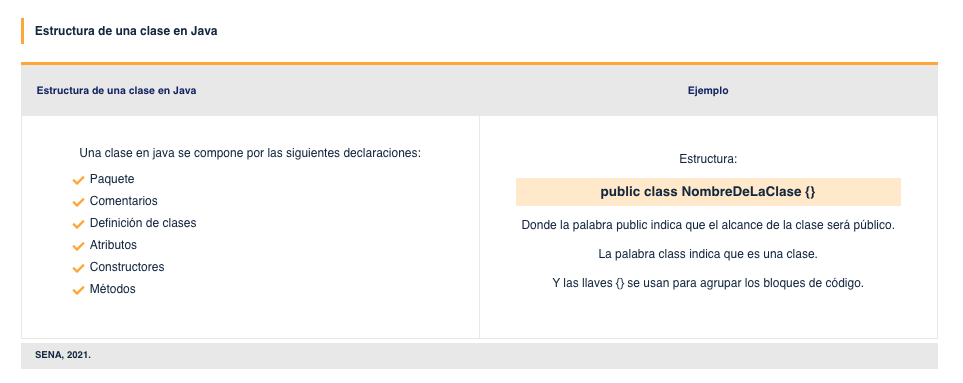
****

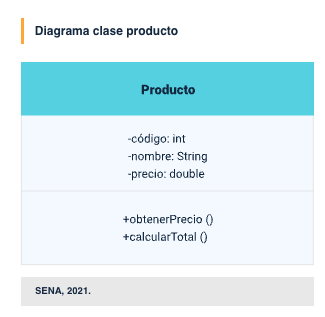


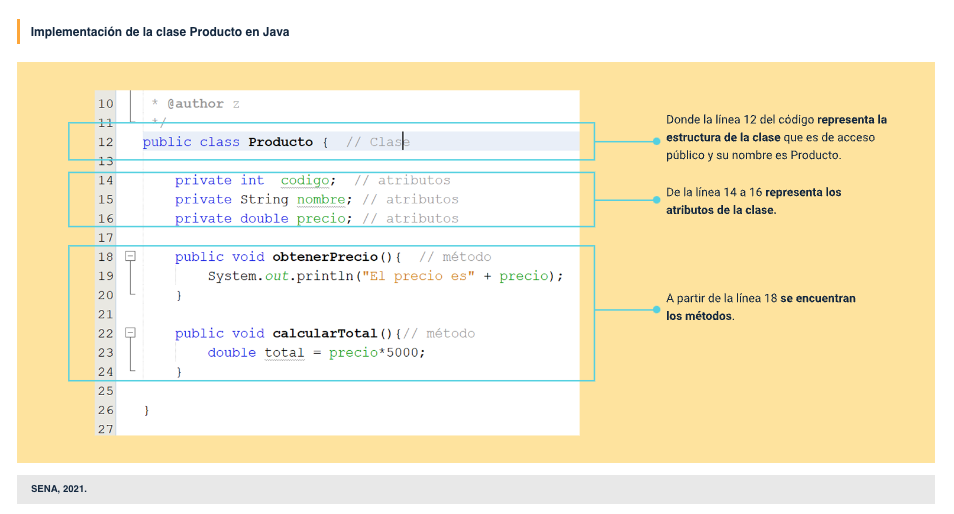
****



**5. Implementación de clases y objetos**



****

****

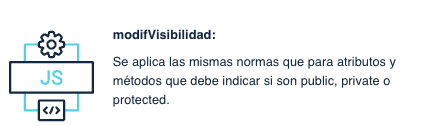
**5.1 Atributos y métodos de la clase**

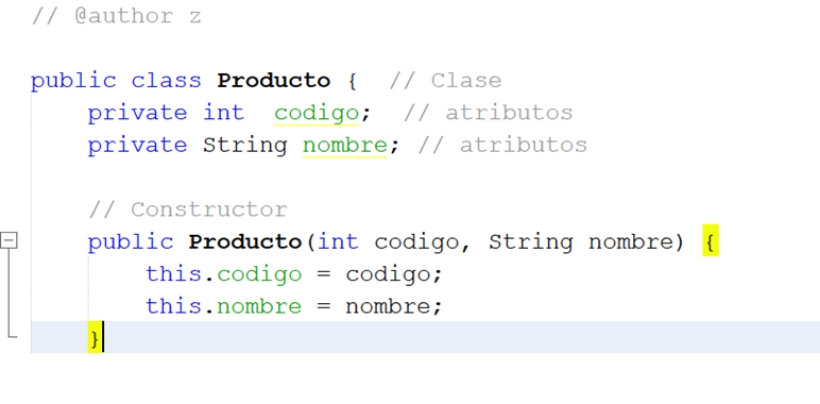


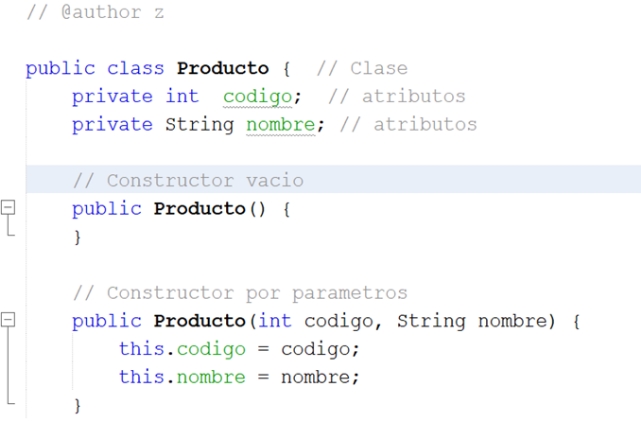




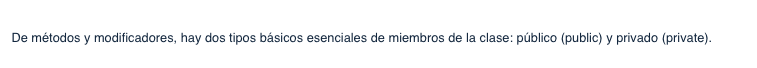
**5.2 Constructores y destructores**



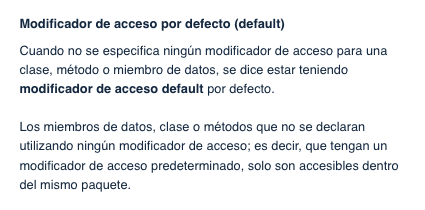


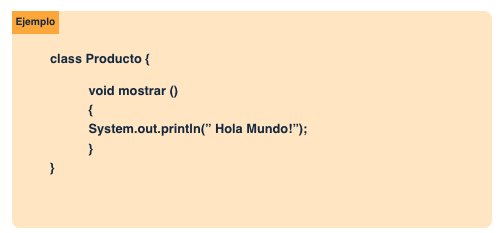


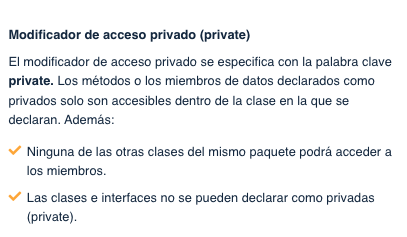
**5.3 Métodos accesores y modificadores**

****

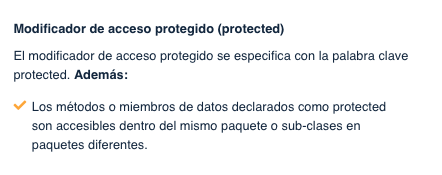
****

****

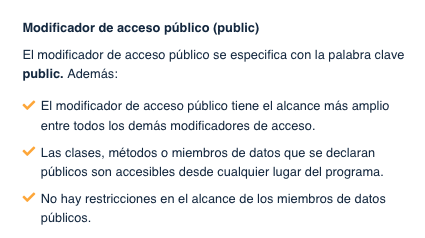


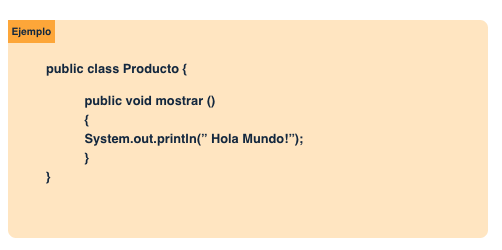
****

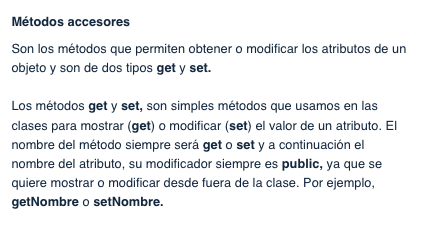


****

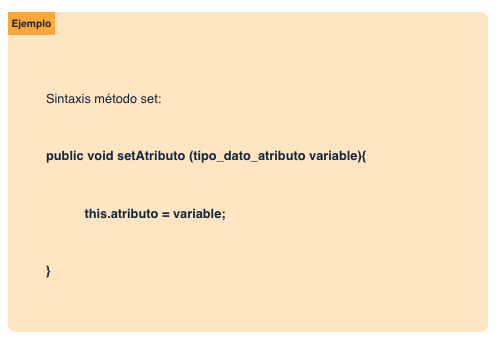


****



****

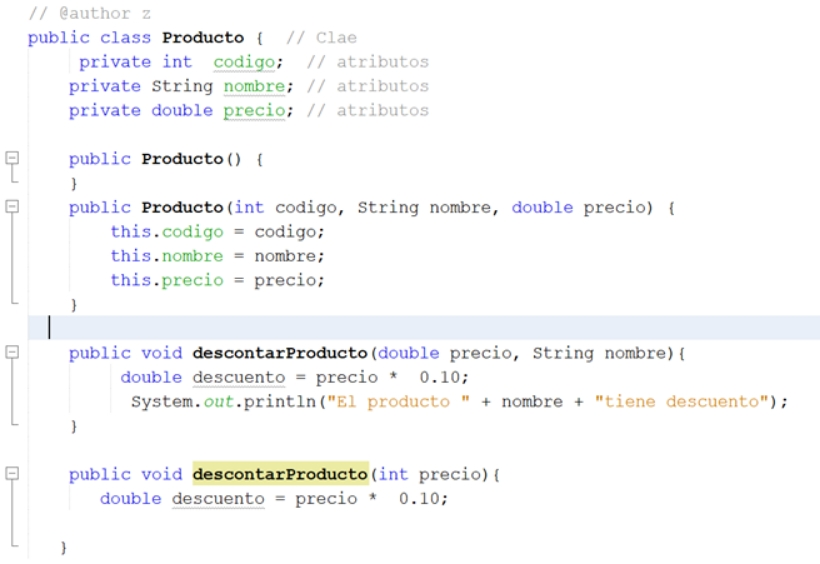


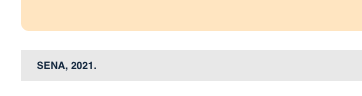
****

****

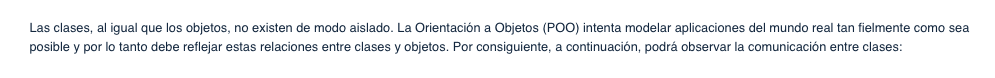
**5.4 Sobrecarga de métodos (*overload*)**

****

****

****

**6. Comunicación entre clases**

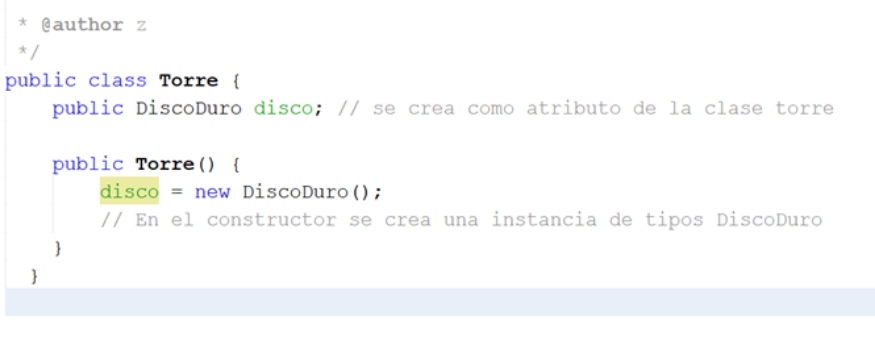
****

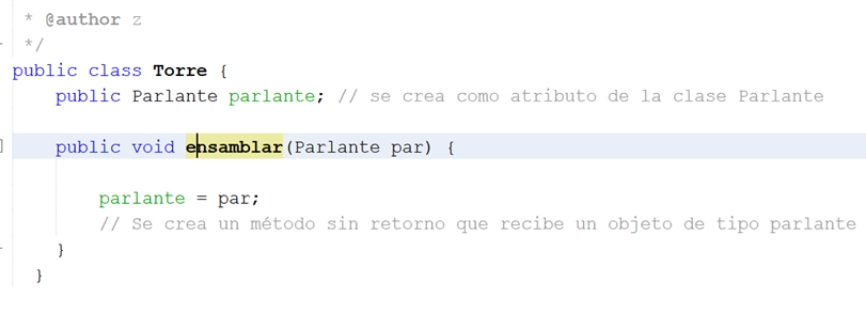


****

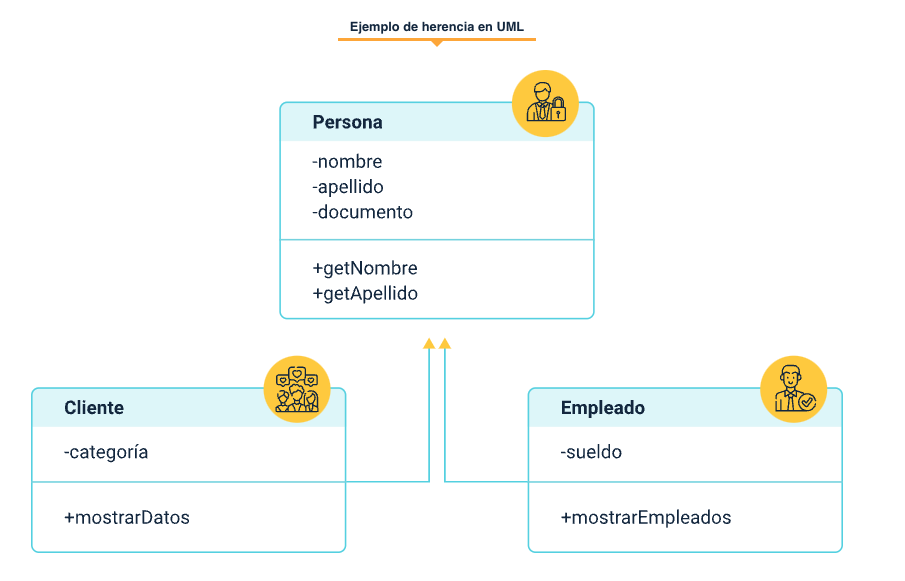
**6.1 Implementación de asociaciones**

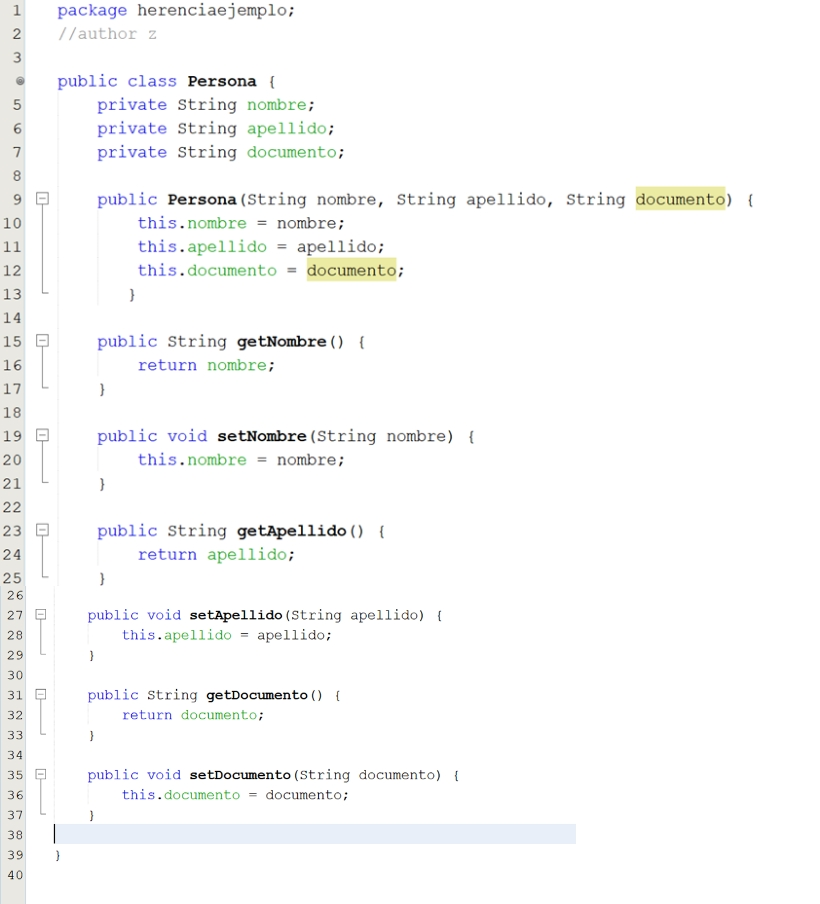
**6.2. Implementación de composición/agregación**

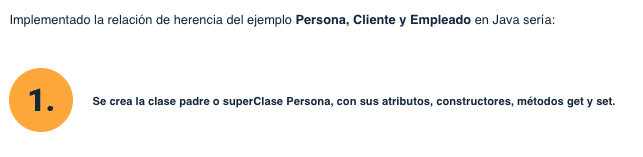
****

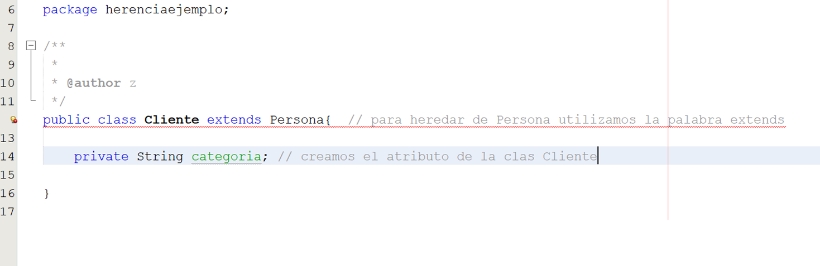


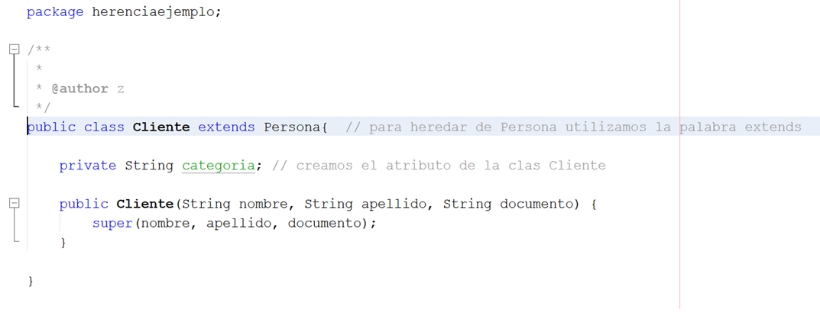
**6.3 Implementación de herencia**

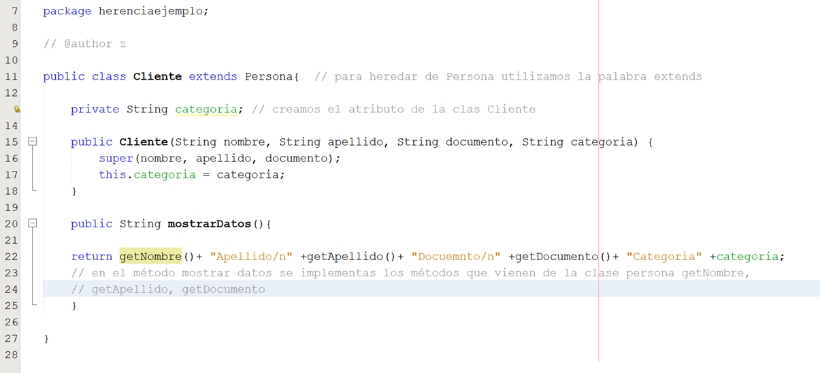
****

****

****





****

****

1. **SÍNTESIS**

A continuación, se presenta a modo de diagrama las temáticas desarrolladas en este componente formativo.

|  |
| --- |
| **CF11\_Gráfico\_Síntesis** |

1. **ACTIVIDADES DIDÁCTICAS**

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Diagramas de clase |
| Objetivo de la actividad | Reconozca |
| Tipo de actividad sugerida | Afianzar conceptos fundamentales de los diagramas de clases a través de su reconocimiento relacional. |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | [*https://ecored-sena.github.io/APLICACION\_SERVICIOS\_NUBE\_M4\_CF2/#/actividad*](https://ecored-sena.github.io/APLICACION_SERVICIOS_NUBE_M4_CF2/#/actividad) |

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Declaraciones en Java |
| Objetivo de la actividad | Identificar y relacionar los conceptos de implementación de clases, objetos, métodos y relaciones en Java. |
| Tipo de actividad sugerida | Emparejamiento |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | *CF11\_Actividad didáctica 02* |

1. **MATERIAL COMPLEMENTARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| 1. Introducción a UML | Lucid Software Español. (2019). *Tutorial - Diagrama de Clases UML.* [Archivo de video]. Youtube. | Video | <https://www.youtube.com/watch?v=Z0yLerU0g-Q> |
| 1. Introducción a UML | Todo Calls. (2017). *Curso UML Diagramas de Clases.* [Archivo de video]. Youtube | Video | <https://www.youtube.com/watch?v=7WRYH2ei5Rw&t=106s> |
| 2. Diagrama de Clase | SENA. (2021). *Diagrama de clases - Ejercicios resueltos*. | Documento | <https://ecored-sena.github.io/APLICACION_SERVICIOS_NUBE_M4_CF2//downloads/ANEXO_DIAGRAMA_CLASES_EJERCICIOS.pdf> |
| 2. Diagrama de Clase | Robles, S. (2016). *HERRAMIENTAS Y ENTORNOS DE PROGRAMACIÓN*. [Pdf]. UCLM. | Página web | <https://silo.tips/download/herramientas-y-entornos-de-programacion> |
| 3. Herramientas CASE | Herramienta CASE. (s.f.) EcuRED. | Blog | <https://www.ecured.cu/Herramienta_CASE#Ejemplos_de_Herramientas_Case_m.C3.A1s_utilizadas>. |
| 3. Herramientas CASE | StarUML. (s.f.). *StarUML.* | Página web | <https://staruml.io/> |
| 4. Características del lenguaje de programación orientada a objetos | Universidad Autónoma del Estado de Hidalgo. (2014). *1.1 Lenguajes de Programación Orientados a Objetos*. | Página web | <http://200.57.56.254/lcc/mapa/PROYECTO/libro15/11_lenguajes_de_programacin_orientados_a_objetos.html> |
| 5. Implementación de clases y objetos | ARIMETRICS. (2020). *Qué es Entorno de desarrollo*. | Página web | <https://www.arimetrics.com/glosario-digital/entorno-de-desarrollo> |
| 5. Implementación de clases y objetos | Codigofacilito. (2012). *Clases, Objetos y Métodos en Java*. | Video | <https://www.youtube.com/watch?v=AEXLtATMkZM> |
| 6. Comunicación entre clases | Oracle Java Technologies | Oracle. (2021). Java Is the Language of Possibilities | Página web | <https://www.oracle.com/java/technologies/> |
| 6. Comunicación entre clases | Systemoff. (2020). *Cómo implementar relaciones de asociación de los diagramas de clases en Java.* [Archivo de video]. YouTube. | Video | <https://www.youtube.com/watch?v=BGStNPa0qRk> |

1. **GLOSARIO**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| camelCase | es una convención de Java que consiste en escribir palabras o frases compuestas de modo que cada palabra o abreviatura comienza con una letra mayúscula o una primera palabra con una letra mayúscula y el resto con minúscula. |
| CASE: | ingeniería de *software* asistida por computación. |
| Herramienta CASE: | una herramienta *software* que automatiza una parte de las fases del ciclo del desarrollo de sw. |
| IDE: | entorno de desarrollo integrado que se utiliza para el desarrollo de un código fuente o programa. |
| JVM: | máquina virtual de Java. |
| Tecnología CASE: | conjunto de instrumentos y técnicas *software* para automatizar una disciplina de la ingeniería, incluyendo metodologías estructuradas y herramientas. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

Andaluza, U. & Garcilaso, I. (2012). *Programación en Java Francisco Javier Cruz Vílchez.*

Cuevas Agustín, G. (1991). *Ingeniería del Software. Práctica de la programación*. Editorial RA-MA.

Deitel, H. M. & Deitel, P. J. (2004). *Como programar en C/C++ y Java*. (4ta Ed.)

Flores Cueto, J. J., & Bertolotti Zuñiga, C. (2013). Diagrama de clases en uml. 6.

Holt, R. (2009). *Introduction to UML: Unified Modeling Language*. U Waterloo <https://cupdf.com/document/introduction-to-uml-unified-modeling-language-ric-holt-u-waterloo-march-2009.html>

Pressman, R. (1998). *Ingeniería del Software. Un enfoque práctico.* (4ta ed.). Editorial Mc. Grraw-Hill

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *El Lenguaje Unificado de Modelado*. Elements, 30. <http://portal.acm.org/citation.cfm?id=993859&dl=>

Schmuller, J. (2001). *Aprendiendo UML en 24 horas*. Pearson Educación.

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia**  ***(Para el SENA indicar Regional y Centro de Formación)*** | **Fecha** |
| **Autor (es)** |  |  |  |  |
|  |  |  |  |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| Autor (es) | Miroslava González Hernández | Diseñadora Instruccional | Regional Santander - Centro industrial del diseño y la manufactura. | Agosto 2023 | Completar elementos del nuevo formato y realizar ajustes sobre el contenido. |
| Rafael Neftalí Lizcano Reyes | Responsable desarrollo curricular Ecosistema RED Santander |

**Nota:**Para la propuesta instruccional se deben tener en cuenta las métricas desarrolladas en el equipo:

<https://drive.google.com/drive/u/1/folders/1UiJvaklSCICR4BaQ7ga_q04JFa53h_u_>