**ACTIVIDAD 1 - INDIVIDUAL - POO**

**OBSERVACIONES IMPORTANTE ANTES DE INICIAR:**

Cada trabajo o actividad académica que sea desarrollado por usted como evidencia de su proceso de aprendizaje, debe estar 100% libre de frases copiadas desde cualquier fuente original (documento, libro, página web, proyecto de código fuente, video tutorial, etc.) de información.

En ningún momento se le está pidiendo que haga las cosas de 0 y olvide por completo los conocimientos sobre los cuales se basa algún tema de esta asignatura, por el contrario, lo estamos animando a que lea, investigue, comprenda, deduzca, analice, sintetice, concluya, interprete, argumente, domine, practique, critique, niegue o afirme con bases sólidas basadas en la lectura y el estudio autónomo y autodidacta.

Tenga muy presente que mientras estudias esta profesión y durante toda tu vida, siempre seras evaluado y medido de acuerdo por un docente, por un instructor, por tu equipo de trabajo, por tus jefes, por la sociedad o por tus clientes.

Antes de realizar cualquier acción indebida en el desarrollo de esta y cualquier otra actividad académica, recuerda que actualmente existen muchas herramientas modernas y avanzadas que existen analizar, buscar, identificar, rastrear y comparar la idoneidad y originalidad de cada contenidos digital (texto, imágenes, archivos, etc), con especial profundidad en archivos de documentos académico o que contienen fuentes de programas informáticos, las cuales permiten detectar plagio de una manera tremendamente simple, rápida y automatizada.

Evita que el pensamiento del mínimo esfuerzo y el facilismo invada tu mente y tus acciones, ya que este destruirá tus sueños de crecimiento y éxito profesional.

**OBSERVACIONES**:

* **actividad para desarrollar de forma individual**

● Se debe subir al aula virtual de la asignatura por sima-pesad (no se aceptan entregas por otro medio)

● Solo se evaluará y calificará a los estudiantes que hayan subido la actividad por sima-pesad en la fecha estipulada

● Existen muchísimas formas (lógicas y estructurales) de desarrollar un mismo ejercicio de programación, se evalúa con 1.0 cualquier evidencia de similitud del contenido del código o trabajo con otros contenidos entregados por los alumnos, encontrados por internet. recuerde que no es ilegal aprender de otros recursos, lo ilegal es tomar ese recurso para aprovecharse del trabajo hecho por otra persona para luego maquillarlo para tratar de hacerlo pasar por nuestro y ganar una buena calificación, es decir, realizar los siguientes trucos viejos y muy fáciles de evidenciar:

○ Reorganizar el código, colocando clases, variables y métodos en otros lugares diferentes al lugar donde los dejó el autor original.

○ Cambiar nombres de clases

○ Cambiar nombre de archivos

○ Cambiar nombre del proyecto

○ Cambiar nombres de variables

○ Cambiar el nombre de los métodos o funciones,

○ Cambiar mensajes de salida,

○ Cambiar System.out.print por JOptionPane

○ Cambiar color y apariencia de los componentes de GUI

○ etc

● Estudiar y practicar en grupo (cipa) no es sinónimo de entregar el mismo el mismo resultado, que una persona desarrolle el ejercicio y se lo pase a los compañeros de grupo para que lo presenten cuando la tarea es individual.

● Cada estudiante debe tener la capacidad, habilidad, competencia y destreza para poder desarrollar todos los ejercicios de forma individual y con su propia lógica.

**FORMATO DE ENTREGA:**

Un archivo comprimido en formato ZIP **(NO USAR OTRO FORMATO) - NO SE ACEPTA**

**Nombre del archivo**:

codigo\_nombre\_apellido\_asignatura\_act\_1\_2024-2**.zip**

**Por ejemplo**:

1234567\_john\_arrieta\_estructuras\_de\_datos\_act\_2\_2024\_2.**zip**

El archivo comprimido debe contener como mínimo los siguientes archivos.

■ Documento en formato EDITABLE .DOC (NO ES PDF) con el d**esarrollo y explicación detallada** del desarrollo de la actividad.

■ El desarrollo y **entrega del trabajo debe contener todos los ítem** típicos y característicos de un trabajo académico.

1. Presentación

2. Tabla de contenido

3. Introducción

4. Objetivos

5. Justificación

6. Desarrollo

a. Debe ser lo suficientemente explicativo,

b. Argumentativo

c. Ilustrativo

d. Enlace de GitHub del repositorio para descargar los proyectos o archivos de código fuente.

7. Bibliografía

Archivo comprimido con los códigos fuentes del desarrollo de la actividad.

**Normas APA para la elaboración de esta actividad**

<https://www.colconectada.com/normas-apa/>

**INFORMACIÓN SOBRE LA ACTIVIDAD ACADÉMICA**

Basados en lo aprendido por medio de las lecturas, investigación, estudio autónomo, práctica constante, para poder realizar esta actividad usted debe realizar los siguientes pasos:

1. Ir al foro de la unidad (FORO DE SELECCION DE TEMA PRÁCTICO) y revisar el número del ejercicio que le fue asignado
2. Desarrollar los puntos que se describen en la sección **Descripción de la actividad** de este documento.

**Material de apoyo:**

* [INSTALACIÓN DE JDK SOBRE WINDOWS 10](https://youtu.be/BoYRcKZbDb0?si=yYLnzKdqa6zHuwdd)
* [INSTALACIÓN DE NETBEANS SOBRE WINDOWS](https://youtu.be/oT1cUI984zU?si=D_VkvXcHJUG-62Ch)
* [INSTALACIÓN DE VSC SOBRE WINDOWS](https://youtu.be/TO4MnZCsONg?si=yY6R5U6i_yvdBE_g)
* [PROGRAMAR EN JAVA DESDE BLOCK NOTAS Y EL TERMINAL DE LINEAS DE COMANDOS](https://youtu.be/HNmwpdVzDYA?si=HdXNIPNLcszl_aot)
* [PROGRAMAR EN JAVA USANDO EL BLOC DE NOTAS](https://www.youtube.com/watch?v=hVSRIlm9D_s&list=PLfB6iylhfbYtmfgYtxlRuGGzNR6fT-MUV&index=5)
* [MISIONTIC - UDEA - JAVA - S9 - MODELADO DE CLASES CON UML](https://www.youtube.com/watch?v=GJQzYzZmTTE&t=311s)
* [9 - DIAGRAMA DE CLASES Y ANALISIS ORIENTADO A OBJETOS - CURSO DE JAVA](https://youtu.be/JPjIxermtos?si=m6Fxpd2e-amCl9Zf)
* [DE DIAGRAMA DE CLASES A CLASES EN JAVA](https://www.youtube.com/watch?v=pn6zZIWKnlk&t=1626s)[10 - DEL DIAGRAMA DE CLASES AL CODIGO JAVA - CURSO DE JAVA](https://www.youtube.com/watch?v=i5iuC_Jph9w)
* **PARTE1 -** [EJEMPLO DE ANALISIS Y DISEÑO ORIENTADO A OBJETOS (AYDOO) - PARTE 1: - DIAGRAMA DE CLASES - UML](https://www.youtube.com/watch?v=q7qF5o5Og9I)
* **PARTE2** [EJEMPLO DE ANALISIS Y DISEÑO ORIENTADO A OBJETOS - PARTE 2 - DEL MODELO AL CODIGO JAVA](https://www.youtube.com/watch?v=aKeMmGLVQb4&list=PLfB6iylhfbYtmfgYtxlRuGGzNR6fT-MUV&index=8)
* **PARTE3**

[EJEMPLO DE ANALISIS Y DISEÑO ORIENTADO A OBJETOS [PARTE 3] CONSTRUCTORES DE LOS OBJETOS O INSTANCIAS](https://youtu.be/GhKiD3UUV60?si=5HaUPiBMewjKXjLX)

* [7 - INTRODUCCION A LOS OBJETOS - CURSO DE JAVA](https://www.youtube.com/watch?v=xTZjyVWJtE4)
* [8 - INTRODUCCION A LA POO - CURSO DE JAVA](https://www.youtube.com/watch?v=P1L6yWjSXHs&t=144s)
* [10 - DEL DIAGRAMA DE CLASES AL CODIGO JAVA - CURSO DE JAVA](https://www.youtube.com/watch?v=i5iuC_Jph9w)
* [11 - CLASES, PROPIEDADES Y CONSTRUCTORES - CURSO DE JAVA](https://www.youtube.com/watch?v=mub8uJOsaT8)
* [12 - CLASES, PROPIEDADES Y METODOS - CURSO DE JAVA](https://www.youtube.com/watch?v=QC3Y5S94pPk)
* [MISIONTIC - JAVA - S8- G4- POO - DIAGRAMA DE CLASES UML Y HERENCIA](https://www.youtube.com/watch?v=fRTBF3Ddois)
* [MISIONTIC - JAVA - S8- G6- POO - DIAGRAMA DE CLASES UML Y HERENCIA](https://www.youtube.com/watch?v=j-aascOs-YI)

[MisionTic Ciclo 2 - Java - Tutoría 43 - POO HERENCIA y POLOMORFISMO](https://www.youtube.com/watch?v=0uMnv022xJg)

[MisionTic Ciclo 2 - Java - Reto 2 - Tutoría 2](https://www.youtube.com/watch?v=DUPMuEI4iUg&t=2515s)

[TUTORIA POO - 2022-01-15](https://www.youtube.com/watch?v=-9bSApesUJo)

[**CURSO DE JAVA - DESDE INICIO HASTA GESTIÓN DE BASES DE DATOS**](https://www.youtube.com/playlist?list=PLfB6iylhfbYu6slj0cCAsMK2Ib3Ge9N7M)

**VIDEOS DESDE 1 AL 12**

**DESCRIPCIÓN DE LA ACTIVIDAD**

Tomando como base el enunciado que le fue asignado que usted y su compañero de grupo han seleccionado:

1. **Realizar un análisis detallado que implique el proceso de abstracción** que permita identificar los requerimientos funcionales, únicas e historias de usuario (leer e investigar sobre requerimientos funcionales)
2. Identificar las clases conceptuales inmersas dentro del enunciado o situación descrita en el ejercicio.
3. Identificar las relaciones que existen entre dichas clases.
4. Realizar y documentar el proceso de análisis y modelado del ejercicio usando la metodología Orientada a Objetos:
   1. Lista de requerimientos funcionales
   2. Listar las Clases
   3. Describir cada clase

NOMBRE DE LA CLASE:

* \_\_\_\_

DESCRIPCIÓN:

* \_\_\_\_

PROPIEDADES DE LA CLASE

* + - * Tipo de dato nombre de Propiedad.
      * \_\_\_
      * \_\_\_
      * \_\_\_

OPERACIONES (MÉTODOS) DE LA CLASE:

* + - * Descripcion del metodo:
      * Firma, declaración, prototipo o cabecera del método:

ámbito tipo\_de\_dato\_retorno nombre\_metodo(parámetros)

**Ejemplo ilustrativo**:

**NOMBRE CLASE**: Usuario:

**DESCRIPCIÓN**: Esta clase representa a todos los objetos de tipo usuario, los cuales tienen relaciones con otros objetos del sistema, tales como, Objetos de tipo X, de tipo Y, de tipo Z, etc.

Los objetos de tipo usuario también se puede subclasificar o subdividirse en UsuarioDocente, UsuarioEstudiante, UsuarioRector, UsuarioPolítico, UsuarioEgresado, UsuarioFuncionario, UsuarioContador, UsuarioAmacenista, UsuarioClinte, UsuarioCajero, etc, lo cual representa una clara relación de herencia entre la clase Usuario y sus derivados. (recordar que el párrafo anterior es solo un ejemplo ilustrativo, el resultado del proceso de análisis depende de cada caso particular)

**PROPIEDADES DE LAS PROPIEDADES DE LA CLASE**:

* código. Tipo de dato: Caracteres. Es requerido: SI
* clave. Tipo: Caracteres. Es requerido: SI
* nombre. Tipo: Caracteres. Es requerido: SI
* apellido. Tipo: Caracteres. Es requerido: SI
* email. Tipo: Caracteres. Es requerido: SI
* teléfono. Tipo: Caracteres. Es requerido: SI
* rol. Tipo: Caracteres. Es requerido: SI

**MÉTODOS DE LA CLASE**:

Operaciones o funciones que realizan los objetos de la clase Usuario

* USuario():
  + Objetivo: Crear y retornar un objeto o instancia de tipo dándole valores por defecto a todas sus propiedades del nuevo objeto a crear.
  + Ámbito: Público
  + Tipo de retorno: Usuario
  + Parámetros: Ninguno
* USuario():
  + Objetivo: Crear y retornar un objeto o instancia de tipo dándole asignado el valor de los parámetros a los valores a las respectivas propiedades del nuevo objeto a crear.
  + Ámbito: Público
  + Tipo de retorno: Usuario
  + Parámetros: Caracteres
* getNombre():
  + Objetivo: Retornar el valor de la propiedad Nombre
  + Ámbito: Público
  + Tipo de retorno Caracteres
  + Parámetros: Ninguno
* setNombre():
  + Objetivo: Cambiar el valor de la propiedad Nombre
  + Ámbito: Público
  + Tipo de retorno: nada
  + Parámetros: Caracteres
* …
* …

1. **RELACIONES ENTRE CLASES:**

En esta sección deben identificar las relaciones que existen entre las diferentes clases y describir cada una defendiendo y representando su nivel o grado de cardinalidad o multiplicidad, según sea cada caso:

Recordar que existen diferentes tipos de tipos de relaciones entre clases:

* Relación de asociación con cardinalidad
* Relación de Herencia
* Relación de Composición
* Relación de Agregación
* Relación de clase Asociación
* Relación de Usabilidad

Ejemplo ilustrativo:

Relaciones de herencia:

Usuario <|--- UsuarioCajero

Relaciones de Asociación con cardinalidad

UsuarioCajero “\*” ——— “\*” Caja

(UsuarioCajero, Caja) : Atender

UsuarioCajero “1” ——— “\*” Venta

Articulo “1” ——— “1” Garantia

Relaciones de Composición:

Matricula <•>—— Asignatura

Casa <°>—— Ventana

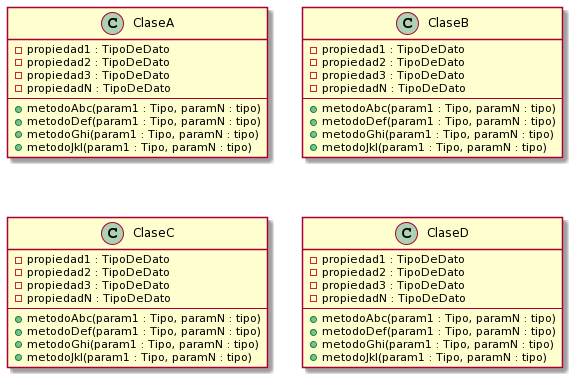
(UsuarioCajero, Caja) : Atender

Asistencia —-----> Falta

1. **MODELAR EN UML** **EL RESPECTIVO DIAGRAMA DE CLASES**

Tomando como base la información obtenida durante el desarrollo del proceso de análisis (pasos anteriores), realizar el proceso de modelado de las clases conceptuales, esto con el fin de obtener el Diagrama de Clases de Dominio o Diagrama de Clases Conceptual, puede elegir cualquier herramienta disponible en internet especializada en facilitar la construcción de Diagramas UML:

**a). EJEMPLO MODELO DE CADA UNA DE LAS CLASES CONCEPTUALES SIN RELACIONES:**



…

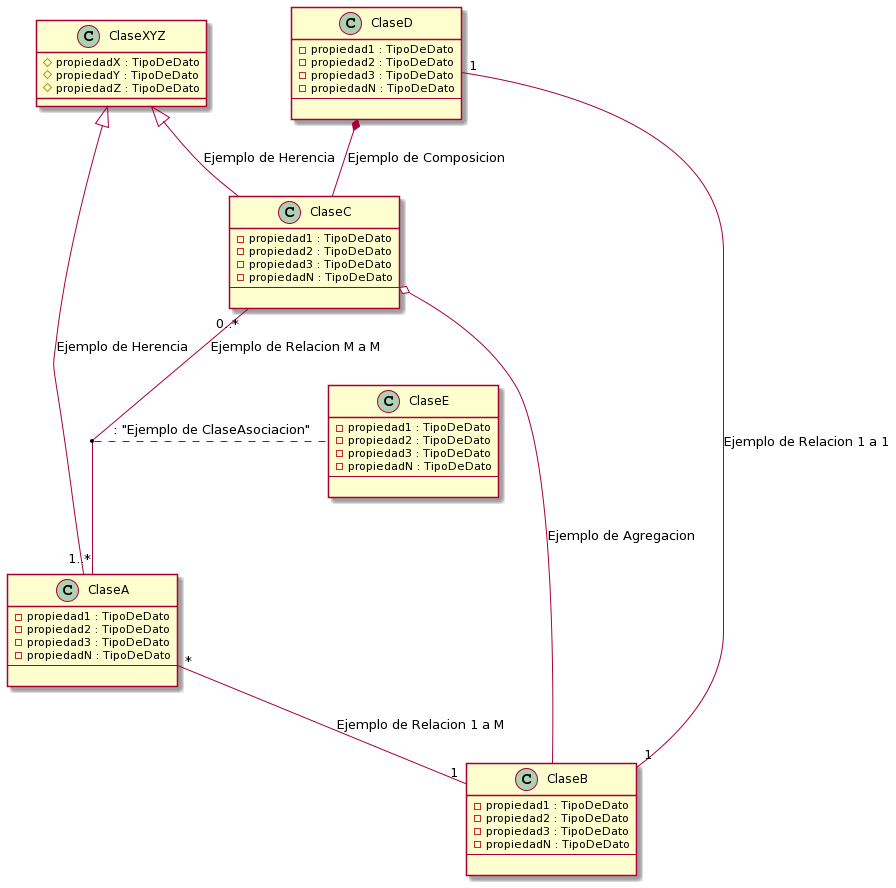
**b). Modelar el diagrama de clases de forma parcial (sin incluir los métodos)**

En caso de utilizar alguna de las siguientes herramientas: PlantUML, PlantText, Yuml o similares, es necesario adjuntar en esta sección del documento el script (código de PlantUML) utilizado para construir el diagrama.

En cualquier otro caso, se debe adjuntar la carpeta con los archivos generados por la herramienta de modelado utilizada, o incluir la dirección URL de donde se pueda descargar o visualizar la imagen del diagrama.

La imagen del diagrama debe agregarse a este documento (o presentarse claramente en el Video de sustentación) en caso de que decidas no elaborar el documento técnico.

**Ejemplo Ilustrativo del diagrama de clases**



1. **DEL DIAGRAMA AL CÓDIGO JAVA**
   * + - El IDE Netbeans y crear un proyecto tipo **Java Application With Ant**
       - El proyecto se debe llamar **TuNombreApellidoAct1Poo20232**
       - Estructura personal del Paquete Raíz
       - **co.edu.udec.poo.nombreapellido**
       - Crear la clase **Principal.java** dentro del paquete Raiz, esta clase debe contener el metodo main
       - Dentro del paquete raíz debe crear otro paquete llamado **dominio**
       - Dentro del paquete dominio debe crear cada una de las clases Java correspondientes a las clases del diagrama de clases conceptual
       - Ir compilando para ir corrigiendo cualquier error de sintaxis.
       - Dentro del método main (clase Principal.java):

Crear al menos un Objeto por cada clase usando el constructor por defecto

Crear al menos un Objeto por cada clase usando el constructor con parámetros

Usar los métodos SET para configurar el valor (estado del objeto) de cada propiedad de los objetos creados por cada clase.

Sobreescribir el método toString() de cada clase para que retorne los la información de cada Objeto:

Tipo: NombreDeLaClase

Propiedad: Valor

Propiedad: Valor

Propiedad: Valor

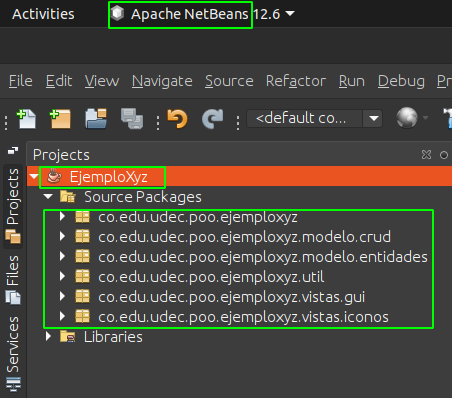
Propiedad: Valor

1. **Comprimir todos los archivos**

* Agregar en el archivo .ZIP lo siguiente:
  + Carpeta del proyecto hecho sobre Netbeans
  + Archivo WORD con el documento (informe técnico)
  + En caso de preferir hacer el video debe incluir el archivo de word con los enlaces necesarios.
    - Enlace de Youtube
    - Enlace de GitHub
    - Enlace con las Imágenes
    - Enlace con las referencias bibliográficas
    - El archivo PDF que contiene el enunciado del ejercicio.
  + Subir el archivo al aula virtual SIMA-PESAD correspondiente a la actividad de Unidad-1.
* Enviarme un correo [jarrietaa2@unicartagena.edu.co](mailto:jarrietaa2@unicartagena.edu.co) informando que ha entregado la actividad y está lista para ser revisada y evaluada.

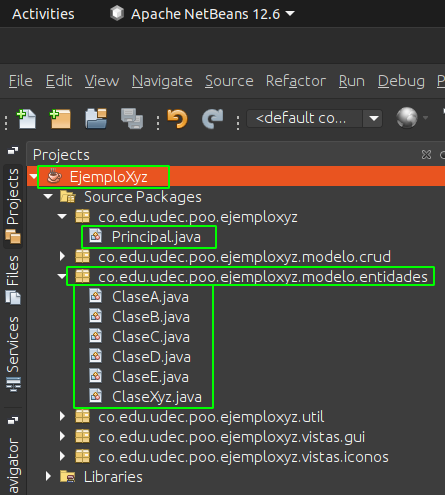
1. Crear un proyecto en Java usando el IDE Netbeans, el cual se debe llamar igual al nombre del archivo del ejercicio que han seleccionado.
2. El proyecto debe tener los siguientes paquetes
   1. paquete raíz:
      * **co.edu.ude.poo.nombreejercicio**
   2. paquete modelo:
      * co.edu.ude.poo.nombreejercicio.**modelo**
   3. paquete entidades
      * co.edu.ude.poo.nombreejercicio.modelo.**entidades**
   4. paquete dao o crud
      * co.edu.ude.poo.nombreejercicio.modelo.**crud**
   5. paquete de servicio o lógica de negocio
      * co.edu.ude.poo.nombreejercicio.**servicio**
   6. paquete de vistas o GUI (Interfaz gráfica de usuario)
      * co.edu.ude.poo.nombreejercicio.**vistas**
   7. paquete de imágenes
      * co.edu.ude.poo.nombreejercicio.vistas.**iconos**
   8. paquete de utilidades
      * co.edu.ude.poo.nombreejercicio.**util**

**Por ejemplo**, supongamos que han seleccionado el ejercicio con el archivo llamado EjemploXyz.pdf, entonces el proyecto debe tener la siguientes estructura:



1. Tomar como insumo principal el **diagrama de clases modelo conceptual** desarrollado en la actividad 1 - Unidad 1 y construir las clases Java según la estructura de cada clase Java y las relaciones representadas en el diagrama.

**Ejemplo**: De acuerdo al ejemplo dado en la actividad #1.

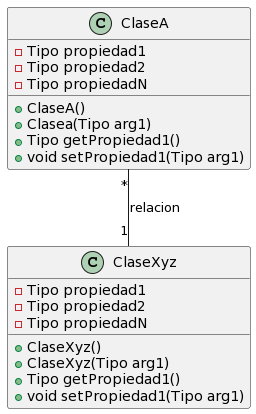


**Principal.java** corresponde a la clase que tiene el método main (punto de inicio o ejecución del programa).

**ClaseA**.java, **ClaseB**.java, **ClaseC**.java, **ClaseD**.java, **ClaseE**.java y **ClaseXyz**.java corresponden a los archivos que contienen el código fuente Java de cada clase que se encuentra en el diagrama de clases conceptual, teniendo en cuenta sus relaciones.

**(Estudiar relaciones entre Objetos, Diagrama de Clases, - Unidad 1)**

Ejemplo: Sea ClaseA y ClaseXyz dos clases en el diagrama de clases conceptual, supongamos que entre ellas existe una relacion de uno a muchos como la siguiente:



Instrucción tomada como referente:

package co.edu.ude.poo.johnarrieta.ejercicioxyc.modelo.entidades;

import java.util.\*;

public class ClaseXyz {

// Propiedades (Estudiar sobre propiedades)

// Estudiar sobre ambito de propiedad

// publico, protegido, amigable o de paquete y privado

// (estudiar sobre encapsulamiento POO)

private Tipo propiedad1;

private Tipo propiedad2;

private Tipo propiedad3;

private Tipo propiedadN;

private List<ClaseA> listaObjetosClaseA;

// Constructor (Estudiar sobre constructores)

// Constructor por defecto

public ClaseXyz(){

}

// Constructor por con argumentos

public ClaseXyz(Tipo parametro1, Tipo Parametro2){

this.propiedad1 = parametro1;

propiedad2 = parametro1;

// (estudiar sobre el operador this)

}

// Métodos SETTER

// (estudiar sobre metodos SET y encapsulamiento POO)

public void setPropiedad1(Tipo parametro){

this.propiedad1 = parametro;

}

// ... el resto de métodos SETTER

// Métodos GETTER

// (estudiar sobre métodos GET y encapsulamiento POO)

public Tipo getPropiedad1(){

return propiedad1 ;

}

// ... el resto de métodos GETTER

// Sobre escribir el método toString para

// que retorne los valores de las propiedades

// del objeto (instancia) que lo invoque.

// (Estudiar sobre sobreescritura de métodos)

@Override

public String toString(){

// código para que retorne los valores

// de todas las propiedades del objeto

}

}

// —------------------------------------------------------------

package co.edu.ude.poo.johnarrieta.ejercicioxyc.modelo.entidades;

public class ClaseA {

// Propiedades (Estudiar sobre propiedades)

// Estudiar sobre ambito de propiedad

// publico, protegido, amigable o de paquete y privado

// (estudiar sobre encapsulamiento POO)

private Tipo propiedad1;

private Tipo propiedad2;

private Tipo propiedad3;

private Tipo propiedadN;

private ClaseXyz objetoClaseXyz;

// Constructor (Estudiar sobre constructores)

// Constructor por defecto

// ...

// Constructor por con argumentos

// ...

// Métodos SETTER

// (estudiar sobre metodos SET y encapsulamiento POO)

// ...

// ... el resto de métodos SETTER

// Métodos GETTER

// (estudiar sobre métodos GET y encapsulamiento POO)

// ...

// ... el resto de métodos GETTER

// Sobre escribir el método toString para

// que retorne los valores de las propiedades

// del objeto (instancia) que lo invoque.

// (Estudiar sobre sobreescritura de métodos)

// ...

}

**Estudiar los temas de Unidad 1 Actividad 1: Clases, Objetos, Propiedades, diagrama de clases, relaciones entre clases.**

**Temas de unidad 2:**

* **Encapsulamiento y ámbito en POO para propiedades y métodos Constructores,**
* **Todo sobre métodos en Java: ámbito, retorno, nombre, parámetros, variables locales, el operador this, llamada o invocación de métodos, métodos Set y Get, sobreescritura del método, el método toString.**
* **Instanciar objetos de una clase en Java**
* **Uso Colecciones de objetos (listas y diccionarios) en Java**

1. El método main debe crear al menos una instancia (objeto) mostrando ejemplos de cómo instanciar nuevos objetos de cada clase usando o invocando **constructores** por defecto y **constructores** con parámetros.

**(Estudiar cómo instanciar o crear un objeto de una clase en Java)**

Instrucción tomada como referente:

**ClaseA objeto1A = new ClaseA();**

**ClaseA objeto2A = new ClaseA(parametros);**

**ClaseXyz objeto1Xyz = new ClaseXyz();**

**ClaseXyz objeto2Xyz = new ClaseXyz(parametros);**

//...

1. Utilizar las instancias u objetos creados para poder invocar sus métodos **set** y poder configurar los valores de cada una de sus propiedades. En este caso, para simplificar el desarrollo del ejercicio, se pueden usar variables locales del método main para asignar los respectivos valores de manera literal en el código.

Instrucción tomada como referente:

objeto1A.setPropiedad1(parametro);

//...

1. De acuerdo a las relaciones representadas en el diagrama de clases, se deben settear (configurar) las propiedades de tipo Referencia a objetos con los que se relaciona cada instancia creada.

Instrucción tomada como referente:

objetoA1.setObjetoClaseXyz(objeto2Xyz);

List<ClaseA> listaA = new ArrayList<>();

listaA.add(objeto1A);

listaA.add(objeto2A);

objeto1Xyz.setListaObjetosClaseA(listaA);

//...

1. Utilizar las instancias u objetos creados para poder invocar sus métodos **get** y poder obtener los valores de cada propiedad e imprimir por consola la información propia de cada objeto.

Instrucción tomada como referente:

Tipo variable = objeto1A.getPropiedad2();

//…

1. Ejecutar el programa para ver el funcionamiento final.