Tecnológico Nacional de México
Instituto Tecnológico de Orizaba

Materia: Programación orientada a objetos

Carrera: Ingeniería en Informática

Tema: Clases y Objetos

Grupo: 1a2A

Integrantes del Equipo:
-Kevin Hernández Heredia (22010603)
-Gutiérrez Montaño Juan Luis (22010601)

Fecha de entrega: 27 de Marzo de 2023

Introducción:

El propósito principal del paradigma de la programación orientada a objetos es explicar los elementos básicos de la programación orientada a objetos (POO). Más específicamente, qué es una clase, cuáles son sus miembros (atributos y métodos), qué es un objeto, qué significa instanciación, cuál es el estado y el comportamiento de un objeto, describe qué es un objeto. Noticias.

Usando varios ejemplos, este tema explica cómo estos elementos se relacionan con el mundo real, haciendo que el diseño de aplicaciones sea más fácil y natural. Al mismo tiempo, se introduce UML (Lenguaje de modelado unificado), un lenguaje de modelado para sistemas de software. Más específicamente, UML se usa para representar gráficamente las clases de manera formal (diagramas de clases).

Competencia específica:

Comprende y aplica la estructura de clases para la creación de objetos y utiliza clases predefinidas para facilitar el desarrollo de aplicaciones.

Marco teórico:

El objeto es el elemento principal de la programación orientada a objetos y alrededor del cual gira este paradigma, de ahí el nombre.

¿Qué se entiende por «objeto» en la POO?

Para responder a estas preguntas lo haremos de manera formal e informal. La formal nos lleva a la siguiente definición.

Un objeto en POO representa alguna entidad de la vida real, es decir, alguno de los objetos únicos que pertenecen al problema con el que nos estamos enfrentando, y con el que podemos interactuar.

Cada objeto, de igual modo que la entidad de la vida real a la que representa, tiene:

Un estado (es decir, unos atributos con unos valores concretos) y un comportamiento (es decir, tiene funcionalidades o sabe hacer acciones concretas).

Informalmente, podemos decir que un objeto es cualquier elemento único del mundo real con el que se pueda interactuar.

¿Qué es una clase?

El concepto clase está íntimamente relacionado con el concepto objeto.

Podemos definir informalmente una clase como una plantilla (o esqueleto o plano) a partir de la cual se crean los objetos.

Por ejemplo, en el mundo hay millones de televisores de la misma marca y modelo. Cada uno de esos televisores ha sido montado a partir de un mismo plano/esqueleto/plantilla y, consecuentemente, todos ellos tienen los mismos componentes, conexiones y funcionalidades. Ese plano/esqueleto/plantilla es, en términos de programación orientada a objetos, una clase.

Por consiguiente, una clase describe las características y el comportamiento de un conjunto de objetos similares en un contexto determinado.

Relación entre la clase y los objetos

En este punto tiene que quedar claro que, cuando hablamos de objeto, hacemos referencia a una estructura que hay en un momento determinado de la ejecución del programa, mientras que cuando hablamos de clase hacemos referencia a una estructura que representa a un conjunto de objetos. Esto implica que:

El tipo de un objeto, en vez de ser de un tipo básico (por ejemplo, int, char, float, etc.), es una clase concreta definida por el programador (TIPO DE DATO ABSTRACTO/TIPO DE DATO DEFINIDO POR EL PROGRAMADOR).

Si estamos en una reunión familiar, cada persona de la familia es única y se puede interactuar con ella. Es decir, cada persona de esa reunión es un objeto. Así pues, Marina, su madre Elena y su padre David son, cada uno de ellos, objetos. Es más, si el bisabuelo y el abuelo paternos de Marina se llaman ambos Manuel, cada uno de ellos es un objeto diferente, puesto que, si bien se llaman igual, no son la misma persona (es decir, el mismo objeto). El bisabuelo y el abuelo, así como el resto de integrantes de la familia, son elementos con los que se puede interactuar de manera individual. Eso sí, todas los miembros de la familia son del mismo tipo Persona (que es la clase a la que pertenecen).

Así pues, si tenemos la clase Persona, en ella están representadas las propiedades que caracterizan a una persona (entidad del mundo real), mientras que los diferentes objetos (por ejemplo, Elena, Marina y David) representan a personas concretas. La siguiente clase llamada Persona agrupa las características comentadas en la clase:

public class Persona {

//Persona TDA= tipo de dato definido por el usuario Tipo de dato Abstracto // características esenciales= atributos dato miembro // ambito tipo de dato identificador private String nomPer; private String apePer; private byte edadPer; private char gnePer; private String tiposanPer;

}

Finalmente, veamos dos ejemplos que deberían servir para acabar de entender qué es un objeto y una clase, así como la relación entre los dos.

Otro Ejemplo:

Cuando vamos a una tienda de electrodomésticos a comprar un televisor, lo más probable es que nos encontremos no con una tele, sino con más de una. Cada uno de esos televisores es un objeto, aunque pertenezcan a la misma marca y al mismo modelo.

Pensemos que dos televisores del mismo modelo, aunque aparentemente parezcan idénticos, no tienen por qué ser iguales. Simplemente hay que pensar que uno puede estar encendido -porque está en el aparador- y el otro apagado porque está dentro del embalaje. Es decir, tienen estados diferentes. Es más, aunque ambos estén apagados y, por consiguiente, tengan el mismo estado, ipuedes tocarlos y ver que son dos objetos diferentes! Eso sí, ambos televisores pertenecen a la clase Televisor.

Lo mismo ocurre con la clase Televisor y con los objetos miTelevisor, tuTelevisor, elTelevisor DelVecino, etc. Por un lado, la clase Televisor representa el concepto abstracto de televisor (es decir, las características y acciones comunes de los televisores), mientras que los tres objetos declarados -miTelevisor, tuTelevisor y elTelevisorDelVecino- son televisores concretos.

Miembros de una clase

Como hemos visto, una clase está formada por unas variables y unas funciones. Estrictamente hablando en términos de POO, a las variables se les llama atributos y a las funciones se les llama métodos. A su vez, al binomio formado por los atributos y los métodos se le denomina miembros de una clase. Así pues, tanto un atributo como un método son miembros de la clase.

Atributos

Los atributos, también llamados campos, son variables que codifican el estado de un objeto. Si tenemos la clase Persona con los atributos:

private String nomPer;

private String apeper;

private byte edadPer;

private char gnePer; private String tiposanPer;

Cada objeto que se defina del tipo Persona tendrá estos atributos.

Métodos

Los métodos implementan el comportamiento de un objeto o, dicho de otro modo, las funcionalidades que un objeto es capaz de realizar.

Haciendo una analogía con la programación estructurada, los métodos serían como funciones (devuelvan algo o no). De ahí que un método, además de por el nombre, se caracteriza por los argumentos (o también llamados parámetros) de entrada que recibe y por el valor de retorno que resulta de ejecutar el comportamiento que implementa. La descripción de estos elementos se conoce como firma del método o signatura del método. En pseudocódigo sería:

Constructor

El constructor se llama de forma automática cuando se crea un objeto para situarlo en memoria e inicializar los atributos declarados en la clase. En la mayoría de lenguajes, el constructor tiene las siguientes características:

1) Normalmente, el nombre del constructor es el mismo que el de la clase.

2) El constructor no tiene tipo de retorno, ni siquiera void.

3) Puede recibir parámetros (o también llamados argumentos) con el fin de inicializar los atributos de la clase para el objeto que se está creando en ese momento.

4) En general suele ser público, pero algunos lenguajes permiten que sea privado.

En el lenguaje java permiten crear más de un constructor, en estos casos, al constructor sin parámetros/argumentos se le suele llamar constructor por defecto o

predeterminado o vacío, mientras que aquellos que tienen parámetros se les llama constructores con argumentos o constructor parametrizado. Como se puede apreciar, decir <<constructor por defecto>> y <con argumentos» es lo mismo que decir que se hace una sobrecarga del constructor. Debido a la sobrecarga, la única limitación cuando se quiere (y se puede) crear más de un constructor es que no pueden declararse varios constructores con el mismo número y el mismo tipo de argumentos.

Destructor

El destructor es el método que sirve para eliminar un objeto concreto definitivamente de memoria. Hay que tener en cuenta que:

1) No todos los lenguajes necesitan implementar un método destructor, este es el caso de Java

2) Por norma general, una clase tiene un solo destructor.

3) No recibe parámetros.

En Java, se usa el método especial finalize() que no devuelve nada (en este caso, sí tiene tipo de retorno, concretamente void). El compilador de Java no obliga a implementar el método finalize(). Así pues, solo se debe codificar si realmente es necesario.

Instancia

Como ya hemos comentado, los objetos son ejemplares de una clase. Así pues, a la hora de crear un objeto, debemos seguir los siguientes pasos:

1) Declarar el objeto.

1. Persona obj;

2) Instanciar el objeto (crear un objeto a partir de una clase).

1. obj = new Persona();

Para instanciar/crear un objeto nuevo debemos escribir la palabra new seguida de uno de los constructores de la clase. En este caso, hemos usado uno de los dos constructores con argumentos, pero también podríamos haber usado el constructor por defecto.

3) En este momento ya tenemos el objeto personal del tipo Persona creado en memoria. Así pues, ya podemos acceder a sus atributos y métodos.

Los pasos 1 y 2 se pueden agrupar en un único paso:

1. Persona obj = new Persona();

Debido a que la acción de crear un objeto a partir de una clase se le llama instanciar, muchas veces a los objetos se les llama instancia. Así pues, personal es una instancia o un objeto de Persona.

Estado y comportamiento

Como hemos leído en la definición formal de objeto, todo objeto en la POO tiene un estado y un comportamiento. Esto es así porque los objetos (o entidades) de la vida real comparten estas dos características.

Debemos entender que el estado de un objeto viene definido por los valores que toman en un instante determinado los atributos que definen ese objeto.

Por su parte, el comportamiento del objeto se puede entender como las funcionalidades que ese objeto es capaz de realizar. Estas funcionalidades que definen el comportamiento de un objeto las define la clase a la que pertenece dicho objeto mediante los métodos de la clase.

Veamos varios ejemplos para entender mejor estos dos conceptos y ver cómo cualquier entidad (u objeto) de la vida real tiene un estado y un comportamiento:

Un perro tiene un estado (nombre, raza, color, etc.) y un comportamiento (ladrar, enterrar huesos, mover la cola, etc.).

Un coche también tiene un estado (velocidad actual, marcha actual, color, longitud, ancho, etc.) y un comportamiento (subir marcha, bajar marcha, encender intermitente, etc.).

Un rectángulo tiene un estado (coordenadas de origen x e y, altura y ancho) así como un comportamiento (área, perímetro, modificar el valor de x, modificar el valor de y, modificar el valor de la altura, modificar el valor del ancho, etc.).

De igual modo, un televisor tiene un estado (encendido o apagado, canal actual, volumen actual, etc.) y un comportamiento (encender, apagar,cambiar a un canal concreto, incrementar el número de canal, decrementar el número de canal, aumentar el volumen, disminuir el volumen, sintonizar, etc.).

Incluso una factura tiene un estado (cobrada o no, importe total, paga y señal abonada, etc.) y un comportamiento (cambiar de no cobrada a cobrada y viceversa, modificar el valor del importe total, etc.).

Hasta algo más abstracto/intangible como un contacto del teléfono tiene un estado (nombre, apellido, teléfono, correo electrónico, etc.) y un comportamiento (introducir un atributo -es decir, nombre, apellido, teléfono, correo electrónico, etc.-, modificar el valor de un atributo y consultar el valor de un atributo).

Así pues, si un televisor concreto (el objeto) tiene el atributo canal actual igual a 5, ese televisor está en un estado diferente a si tuviera el canal actual igual al número 6.

Si nos fijamos en los ejemplos anteriores, nos daremos cuenta de que hay dos tipos de métodos:

1) Aquellos que hacen acciones que realiza la entidad real (por ejemplo, ladrar en el caso del perro, calcular el área de un rectángulo, la acción de encenderse de un televisor, etc.).

2) Aquellos que operan directamente sobre los atributos del objeto. Por un lado, están los métodos que modifican el valor de los atributos del objeto (por ejemplo, modificar el número de teléfono de un contacto o el canal actual de un televisor) y que, por consiguiente, cambian el estado del objeto. Por otro lado, están los métodos que consultan el valor de los atributos del objeto (por ejemplo, consultar el número de teléfono de un contacto o el canal actual de un televisor). Estos métodos de modificación llamados setter y getter, respectivamente. y consulta son

Mensaje

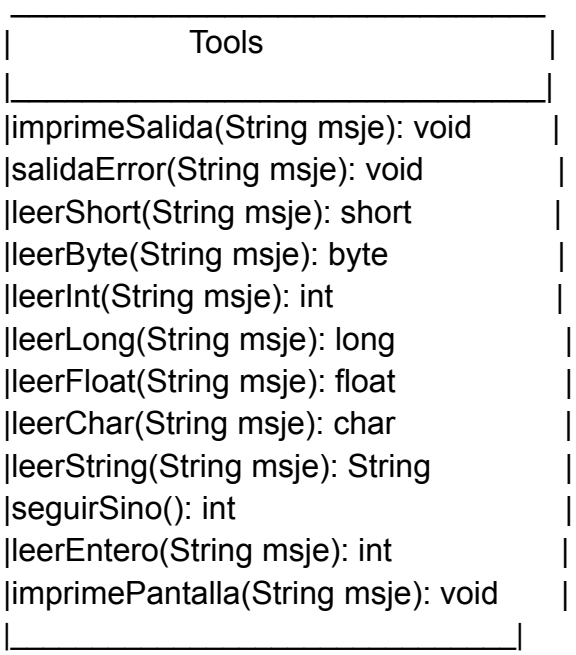
Cuando los objetos quieren interactuar entre ellos utilizan mensajes. Un mensaje es la manera que existe de acceder a los atributos y métodos de un objeto.

Material y equipo

- Computadora de escritorio o laptop
- Windows 10 u 11
- La versión más reciente de Eclipse como IDE para Java o NetBeans

Desarrollo de la practica

Se debe de crear un nuevo proyecto en Eclipse o NetBeans en dónde sé crearás paquetes que contendrán las clases que se realizará.
Dentro de la carpeta sourse se crean tres paquetes, el primero se llama "Entradas y Salidas" en dónde alojaremos una clase que nos ayudará a reutilizar código para el procesamiento de presentación de datos capturados.



En este diagrama, se puede ver que la clase "Tools" tiene los siguientes métodos públicos:

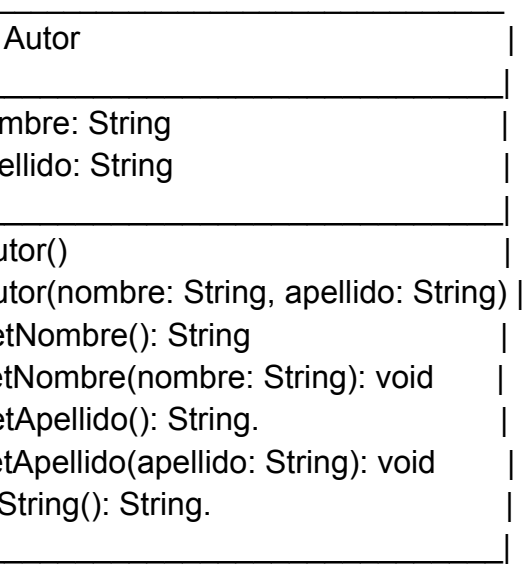
imprimeSalida(String msje): void
salidaError(String msje): void
leerShort(String msje): short
leerByte(String msje): byte
leerInt(String msje): int
leerLong(String msje): long
leerFloat(String msje): float
leerChar(String msje): char
leerString(String msje): String
seguirSino(): int
leerEntero(String msje): int
imprimePantalla(String msje): void

Cada método está representado por un rectángulo en el diagrama, con su nombre, parámetros y tipo de retorno. La flecha que apunta hacia fuera desde cada método indica que es un método público. El tipo de retorno se muestra después del nombre del método y se separa con dos puntos. Los parámetros de entrada se muestran entre paréntesis después del nombre del método, con el nombre del parámetro

seguido de dos puntos y luego el tipo del parámetro. Los métodos que no tienen parámetros se muestran con paréntesis vacíos. En general, el diagrama UML representa la misma información que el diagrama de clases que generamos anteriormente, pero con una presentación más formal y estructurada

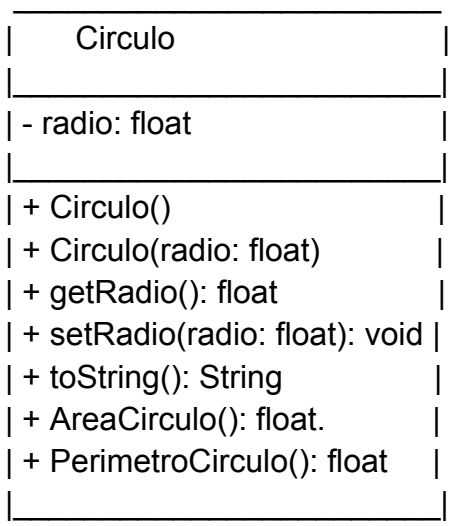
Después, creamos el siguiente paquete en dónde iremos poniendo las clases para cada problema.
Los siguientes son las clases que contiene el paquete TDA:

Autor.java



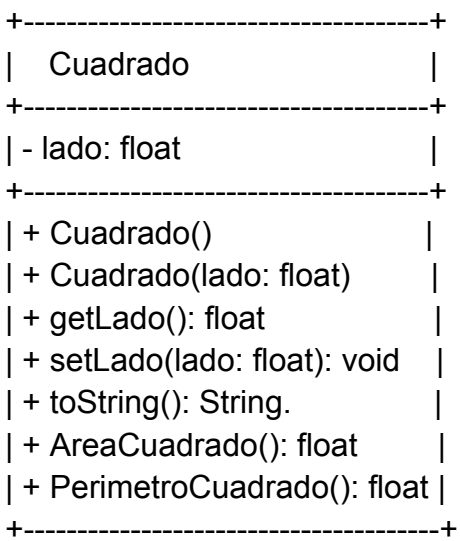
En este diagrama, se puede ver que la clase "Autor" tiene dos campos privados: nombre y apellido. También tiene dos constructores: uno sin argumentos y otro que acepta un nombre y un apellido como parámetros. Los métodos getNombre() y getApellido() devuelven el valor de los campos nombre y apellido, respectivamente. Los métodos setNombre() y setApellido() actualizan los campos correspondientes con un nuevo valor. El método toString() devuelve una cadena que representa el objeto "Autor" en forma de "Nombre=nombre, Apellido=apellido". Todos los métodos son públicos.

Círculo.java



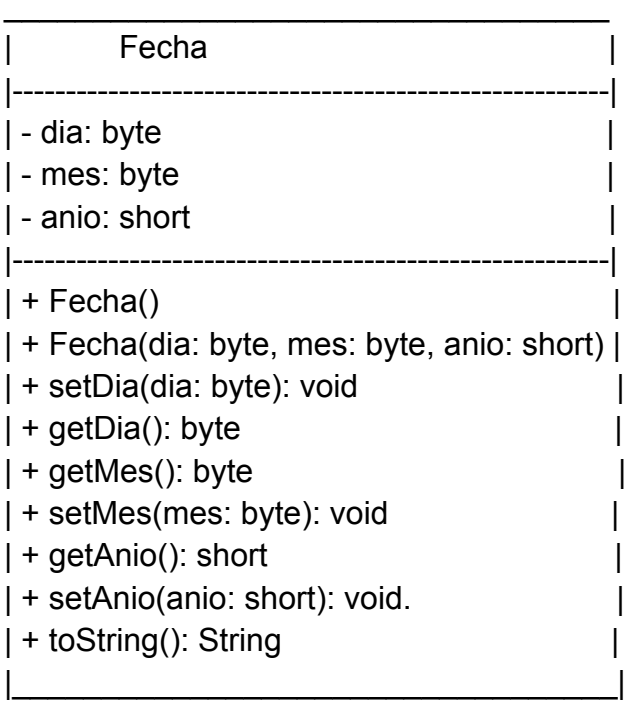
En este diagrama, se puede ver que la clase "Circulo" tiene un campo privado: radio. También tiene dos constructores: uno sin argumentos y otro que acepta un valor de radio como parámetro. Los métodos getRadio() y setRadio() devuelven y actualizan el valor del campo radio, respectivamente. El método toString() devuelve una cadena que representa el objeto "Circulo" en forma de "radio=valorDeRadio". Los métodos AreaCirculo() y PerimetroCirculo() devuelven el área y el perímetro del círculo, respectivamente. Todos los métodos son públicos

Cuadrado.java



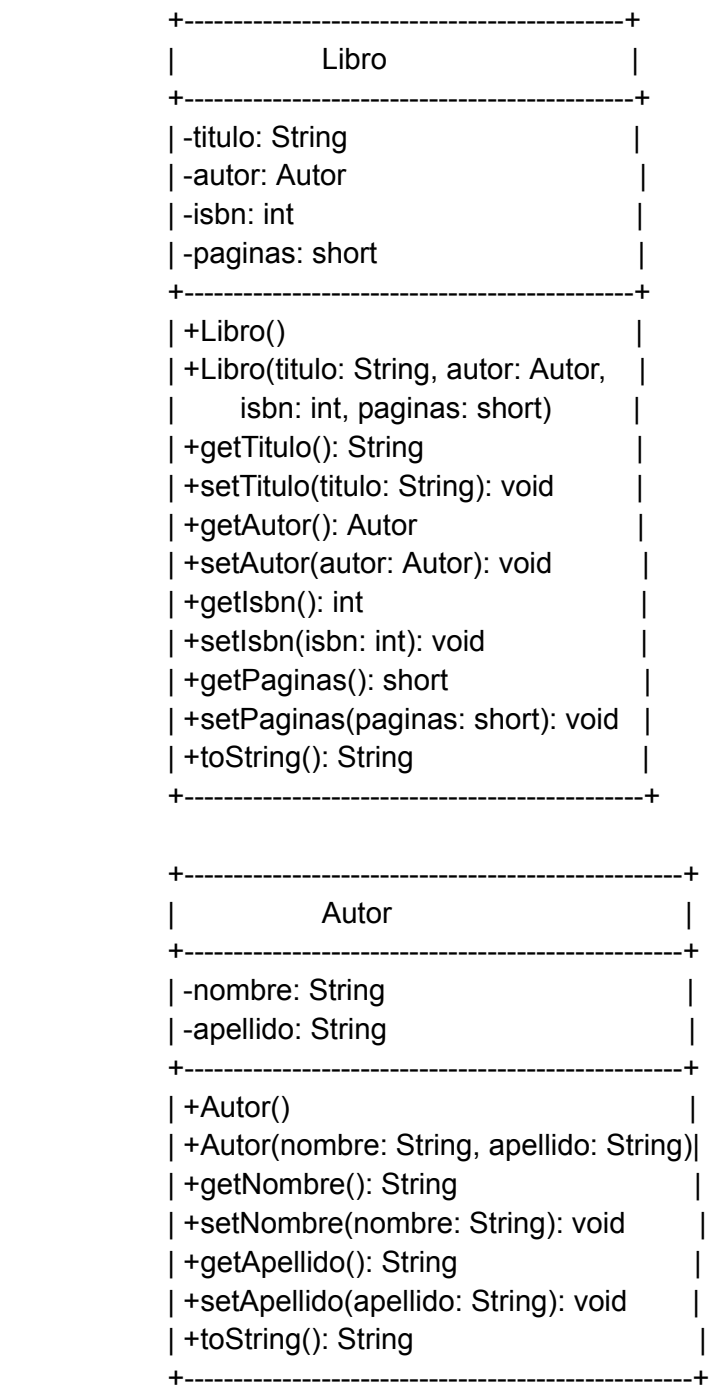
La clase Cuadrado tiene un atributo privado lado de tipo float.
Tiene dos constructores, uno por defecto y otro que recibe un parámetro lado.
Tiene los métodos públicos getLado y setLado para acceder y modificar el valor de lado.
El método toString devuelve una cadena con información del objeto Cuadrado.
Los métodos AreaCuadrado y PerimetroCuadrado calculan y devuelven el área y el perímetro del cuadrado, respectivamente.

Fecha.java



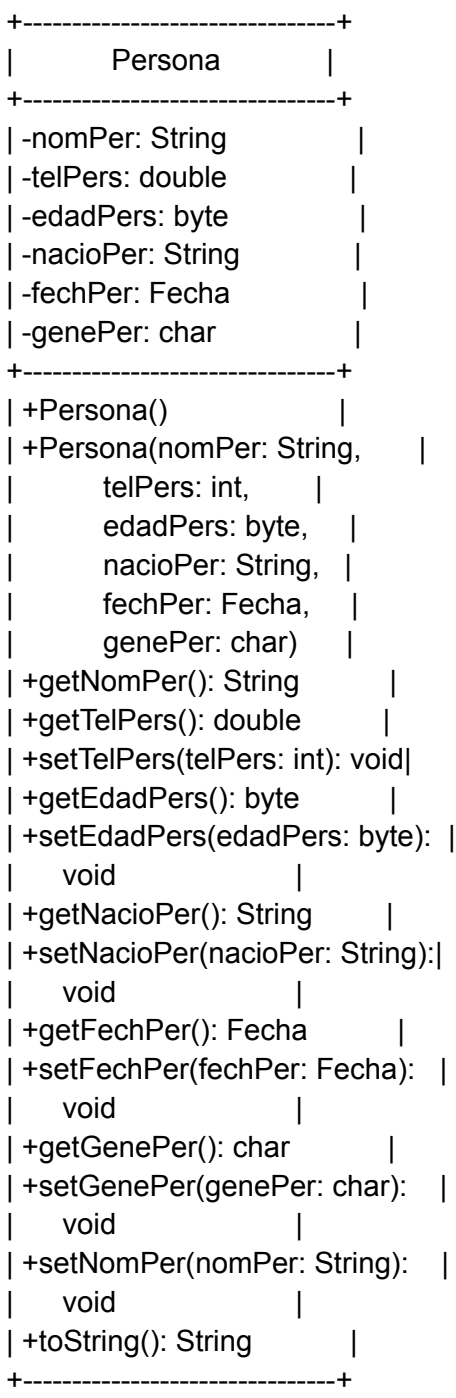
La clase Fecha tiene tres atributos privados: dia, mes y anio. Además, cuenta con un constructor vacío y otro que recibe los tres parámetros para inicializar los atributos. También dispone de los métodos set y get para cada uno de los atributos, así como el método toString() para mostrar la información de la fecha en una cadena de texto.

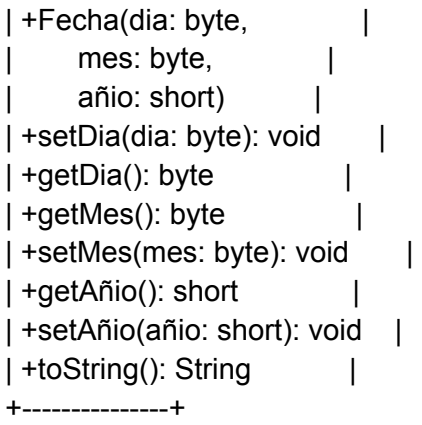
Libro.java



La clase Libro tiene una relación de composición con la clase Autor, lo que significa que un objeto de la clase Libro contiene una referencia a un objeto de la clase Autor. Además, la clase Libro tiene cuatro atributos: titulo, autor, isbn y paginas. La clase Autor tiene dos atributos: nombre y apellido. Ambas clases tienen constructores, métodos getters y setters, y el método toString para convertir los objetos en cadenas de texto.

Persona.java



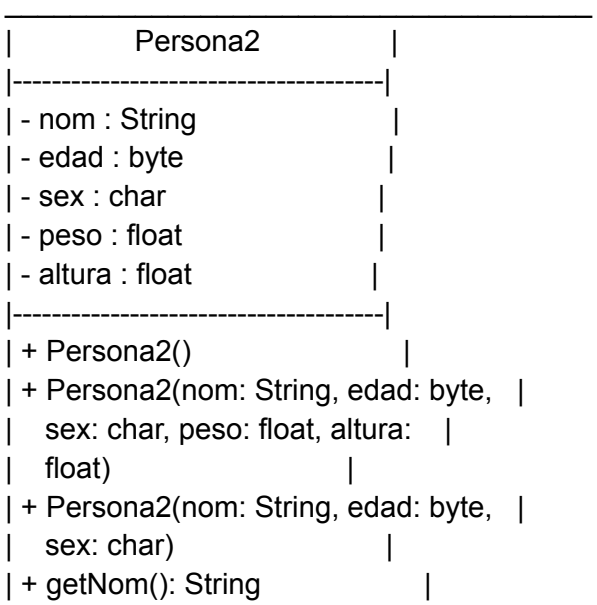


El diagrama muestra dos clases: Persona y Fecha. Persona tiene los atributos nomPer, telPers, edadPers, nacioPer, fechPer y genePer, mientras que Fecha tiene los atributos dia, mes y añio.

Persona tiene un constructor vacío y un constructor parametrizado que inicializa todos los atributos. También tiene métodos getters y setters para cada atributo y un método toString para imprimir los valores de todos los atributos.

Fecha también tiene un constructor vacío y un constructor parametrizado para inicializar los atributos. Tiene métodos getters y setters para cada atributo y un método toString para imprimir los valores de todos los atributos.

Persona2.java



```
| + setNom(nom: String): void          |
| + getEdad(): byte                    |
| + setEdad(edad: byte): void          |
| + getSex(): char                     |
| + setSex(sex: char): void            |
| + getPeso(): float                   |
| + setPeso(peso: float): void         |
| + getAltura(): float                 |
| + setAltura(altura: float): void     |
| + calculaIMC(peso: float, altura:    |
|   float): byte                       |
| + esMayorDeEdad(edad: byte): boolean |
| + toString(): String                 |
---------------------------------------
```

La clase Persona2 tiene cinco atributos privados (nom, edad, sex, peso, altura) y tres constructores. También tiene métodos getter y setter para todos sus atributos, así como dos métodos estáticos (calculaIMC y esMayorDeEdad) que devuelven un valor en función de los parámetros proporcionados. La clase Persona2 también tiene un método toString que devuelve una cadena que representa la información de la persona.

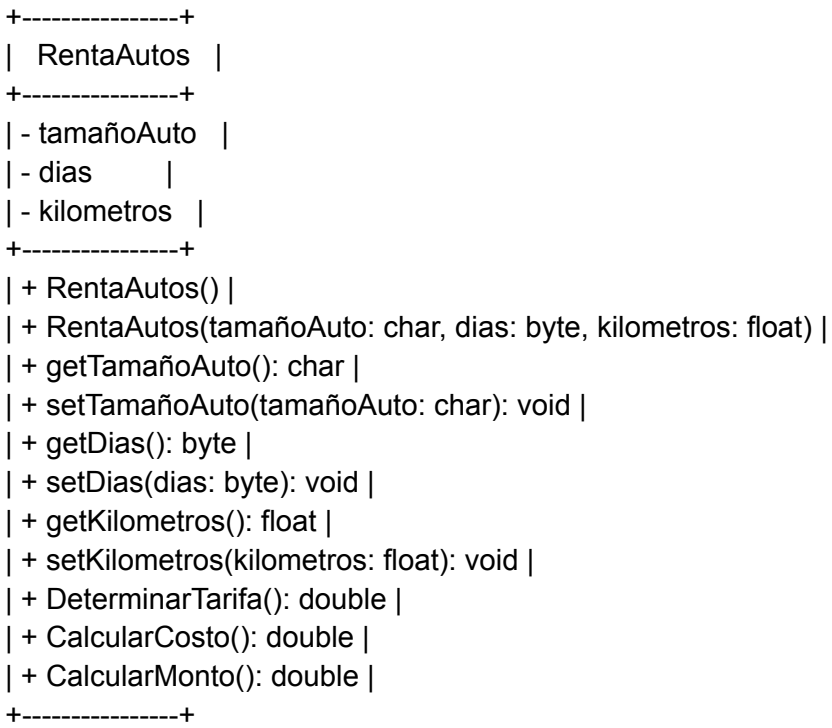
Rectángulo.java

```
+-----------------+
|   Rectangulo    |
+-----------------+
| - base: float   |
| - altura: float |
+-----------------+
| + Rectangulo()  |
| + Rectangulo(base: float, altura: float) |
| + getBase(): float |
| + setBase(base: float): void |
| + getAltura(): float |
| + setAltura(altura: float): void |
| + toString(): String |
| + AreaRectangulo(): float |
| + PerimetroRectangulo(): float |
+-----------------+
```

La clase Rectangulo tiene dos atributos privados: base y altura, y dos métodos constructores, uno vacío y otro parametrizado que recibe los valores de la base y la

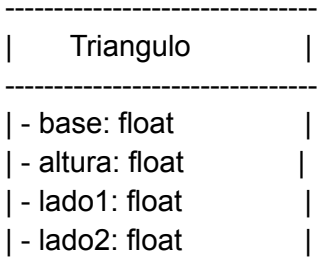
altura. Además, la clase tiene métodos públicos para obtener y establecer los valores de la base y la altura, y dos métodos para calcular el área y el perímetro del rectángulo. Por último, la clase tiene el método toString() para imprimir los valores de los atributos del objeto.

RentaAutos.java



Las tres propiedades tamañoAuto, dias y kilometros tienen sus respectivos getters y setters. La clase cuenta con tres métodos: DeterminarTarifa(), CalcularCosto() y CalcularMonto(), que realizan cálculos para determinar la tarifa, costo y monto de la renta del automóvil, respectivamente.

Triangulo.java



| - catetoO: float |
| - catetoA: float |

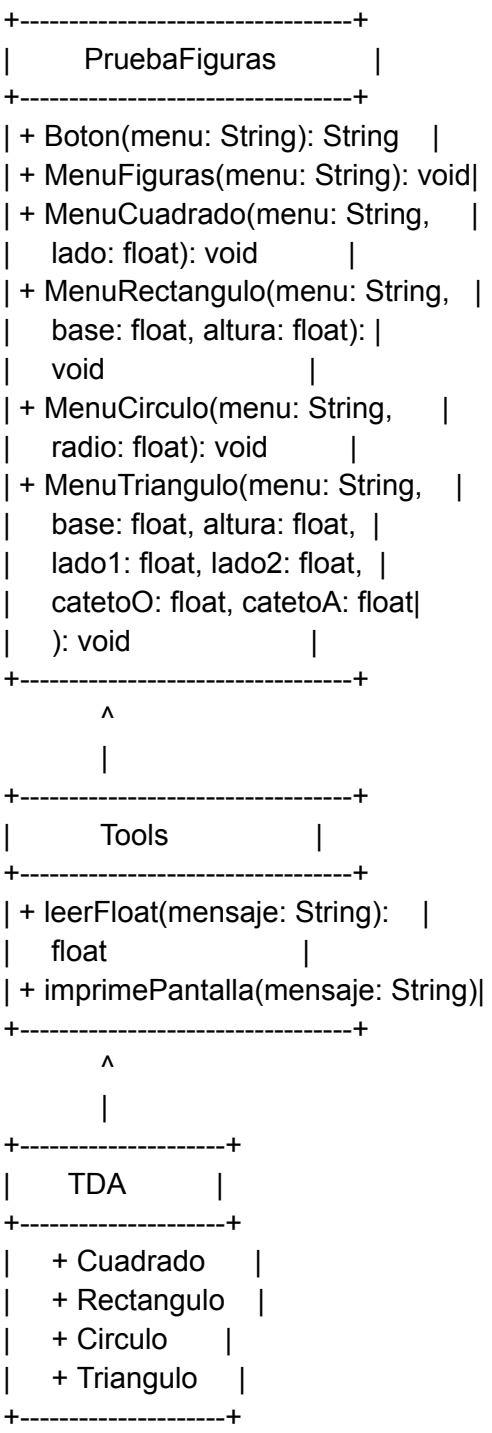
| + Triangulo() |
| + Triangulo(float, float, |
| float, float, |
| float, float) |
| + getBase(): float |
| + setBase(float): void |
| + getAltura(): float |
| + setAltura(float): void |
| + getLado1(): float |
| + setLado1(float): void |
| + getLado2(): float |
| + setLado2(float): void |
| + getCatetoO(): float |
| + setCatetoO(float): void |
| + getCatetoA(): float |
| + setCatetoA(float): void |
| + AreaTriangulo(): float |
| + PerimetroTriangulo(): float|
| + HipotenusaTriangulo(): float|
| + TipoTriangulo(): String |

Las variables base, altura, lado1, lado2, catetoO y catetoA son privadas, por lo que solo se pueden acceder a ellas a través de los métodos get y set correspondientes. Además, la clase cuenta con los siguientes métodos públicos:

Triangulo(): constructor vacío.
Triangulo(float, float, float, float, float, float): constructor que recibe como parámetros los valores iniciales de las variables de instancia.
AreaTriangulo(): método que devuelve el área del triángulo.
PerimetroTriangulo(): método que devuelve el perímetro del triángulo.
HipotenusaTriangulo(): método que devuelve la longitud de la hipotenusa del triángulo.
TipoTriangulo(): método que devuelve una cadena indicando el tipo de triángulo (equilátero, isósceles o escaleno).

Por último creamos el último paquete en dónde creamos las pruebas de las clases del paquete TDA, a este paquete lo nombraremos cómo TestClasesObjetos.
Las clases que contiene son:

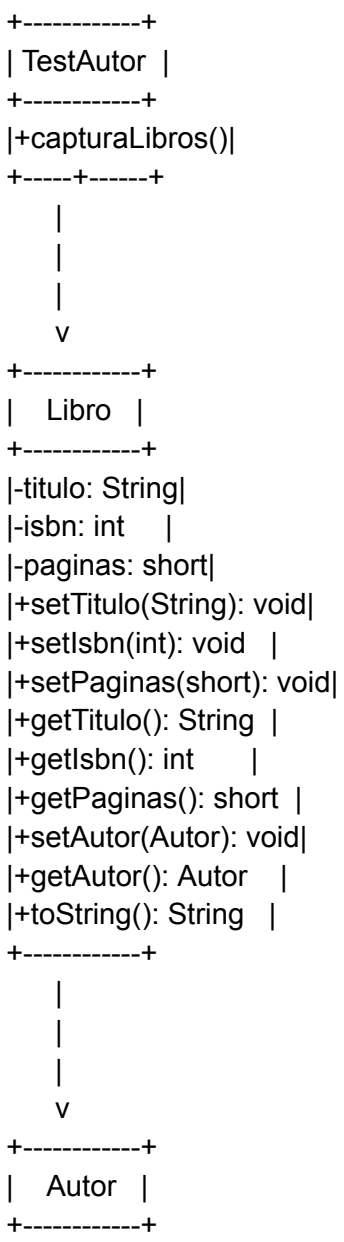
PruebaFiguras.java

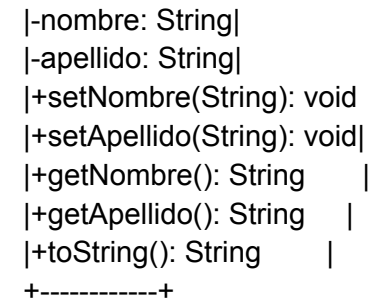


En el diagrama se muestran cuatro clases dentro del paquete TDA: Cuadrado, Rectangulo, Circulo y Triangulo. También se muestra la clase Tools y la clase PruebaFiguras que contiene los métodos Boton, MenuFiguras, MenuCuadrado,

MenuRectangulo, MenuCirculo y MenuTriangulo. La relación entre las clases PruebaFiguras y Tools indica que la clase PruebaFiguras utiliza los métodos de la clase Tools. También hay relaciones entre PruebaFiguras y las clases Cuadrado, Rectangulo, Circulo y Triangulo, indicando que la clase PruebaFiguras utiliza estas clases para crear objetos y realizar cálculos de área y perímetro. Las relaciones entre PruebaFiguras y las clases MenuCuadrado, MenuRectangulo, MenuCirculo y MenuTriangulo indican que la clase PruebaFiguras utiliza estos métodos para mostrar menús de opciones en la consola

TestAutor.java



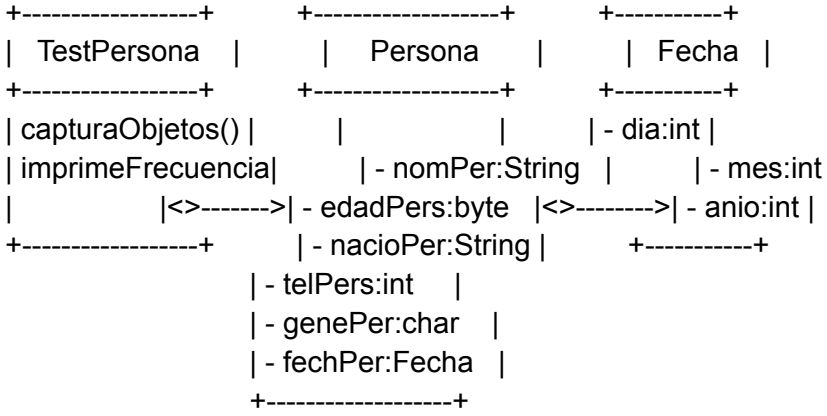


Este diagrama representa las tres clases involucradas en el código: TestAutor, Libro y Autor. La clase TestAutor es la clase principal que contiene el método main que se ejecuta al iniciar el programa. Este método llama al método capturaLibros, que es donde se lleva a cabo la captura de los libros.

La clase Libro representa un libro, con sus atributos titulo, isbn y paginas. También tiene métodos para establecer y obtener los valores de estos atributos, así como para establecer y obtener el autor del libro.

La clase Autor representa al autor de un libro, con sus atributos nombre y apellido. También tiene métodos para establecer y obtener los valores de estos atributos, así como para representar al autor como una cadena de caracteres

TestPersona.java



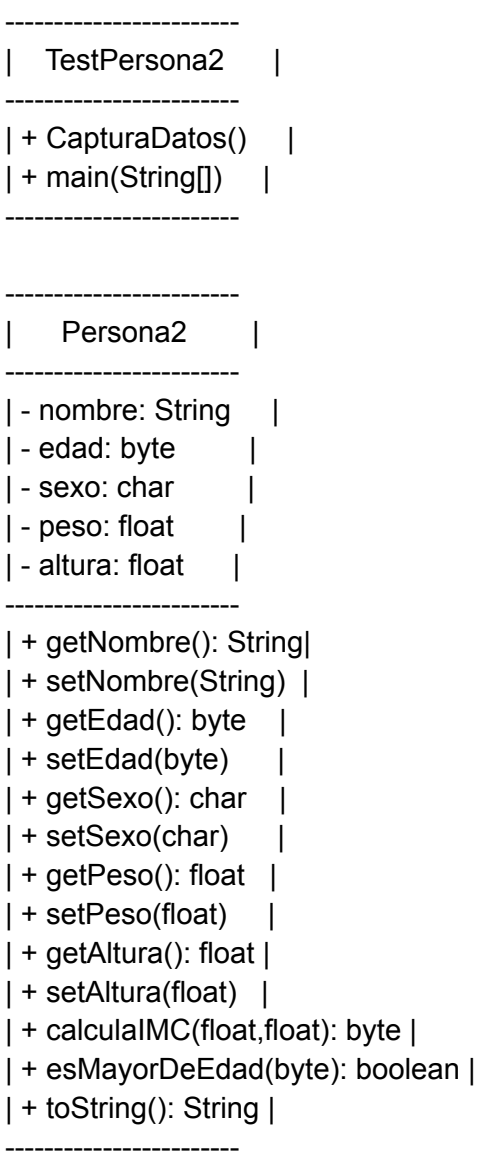
La clase TestPersona es la clase principal que tiene un método capturaObjetos() que instancia objetos de la clase Persona. Cada objeto de Persona tiene atributos como nomPer, edadPers, nacioPer, telPers, genePer y fechPer que representan el

nombre, edad, nacionalidad, teléfono, género y fecha de nacimiento de cada persona.

La clase Fecha representa la fecha de nacimiento y tiene los atributos dia, mes y anio.

Además, la clase TestPersona tiene un método adicional imprimeFrecuencia que se utiliza para imprimir el porcentaje de mujeres y hombres.

TestPersona2.java



La clase TestPersona2 tiene dos métodos, CapturaDatos() y main(), el cual utiliza la clase Persona2. La clase Persona2 tiene cinco atributos: nombre, edad, sexo, peso, y altura. Además, tiene varios métodos públicos, que permiten establecer y obtener los valores de los atributos, y realizar operaciones como el cálculo del índice de masa corporal (IMC) y verificar si la persona es mayor de edad. La clase también tiene un método toString() que devuelve una cadena de caracteres con todos los atributos del objeto en formato de texto.

RentaAutos.java

| +leerByte(m : String) : byte|
| +leerFloat(m : String) : float|
| +imprimePantalla(m : String) : void|
| +seguirSino() : int|
+------------------+

La clase TestRentaAutos es la clase principal que contiene el método main() y el método CapturaRentaAutos() que instancia y usa objetos de la clase RentaAutos.

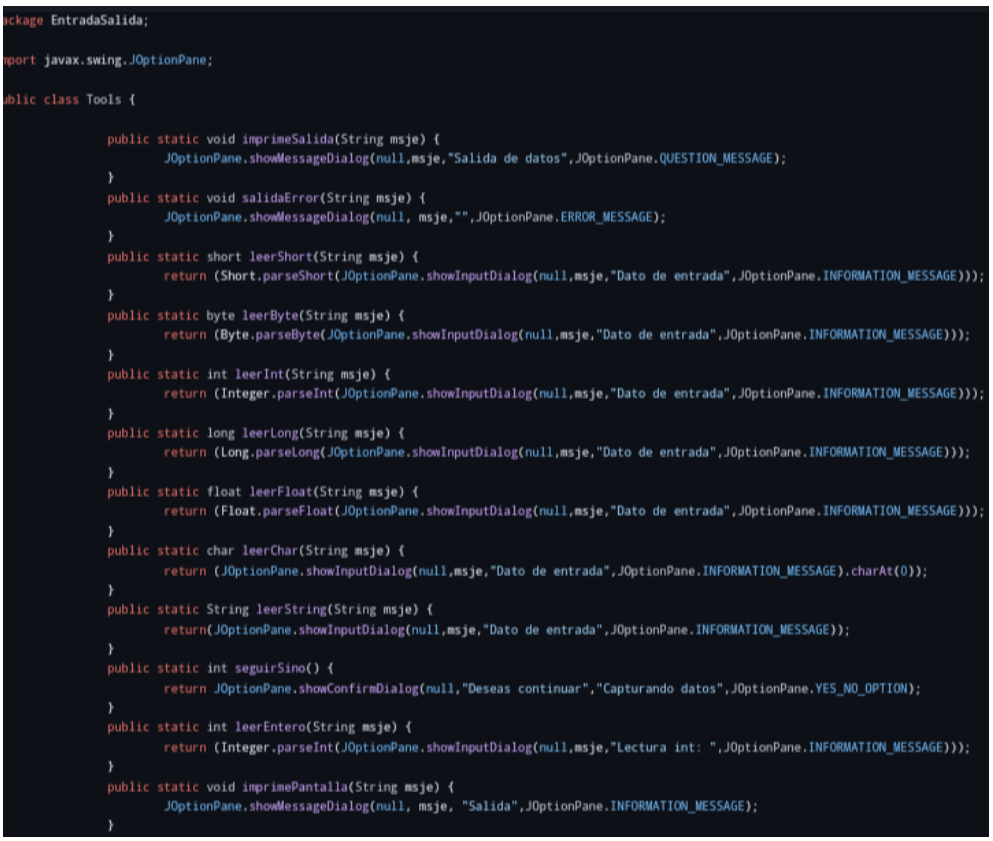
La clase RentaAutos tiene las propiedades tipoAuto, dias y kilometros, y los métodos para obtener y establecer el valor de cada propiedad, así como los métodos para determinar la tarifa, calcular el costo y calcular el monto total de la renta.

La clase Tools es una clase de utilidad que proporciona métodos para leer y escribir en la pantalla, así como para solicitar entrada del usuario y para solicitar confirmación.

Resultados

A continuación se explica el código fuente para cada paquete:

EntradaSalida



```java
ckage EntradaSalida;

mport javax.swing.JOptionPane;

blic class Tools {

	public static void imprimeSalida(String msje) {
		JOptionPane.showMessageDialog(null,msje,"Salida de datos",JOptionPane.QUESTION_MESSAGE);
	}
	public static void salidaError(String msje) {
		JOptionPane.showMessageDialog(null, msje,"",JOptionPane.ERROR_MESSAGE);
	}
	public static short leerShort(String msje) {
		return (Short.parseShort(JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE)));
	}
	public static byte leerByte(String msje) {
		return (Byte.parseByte(JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE)));
	}
	public static int leerInt(String msje) {
		return (Integer.parseInt(JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE)));
	}
	public static long leerLong(String msje) {
		return (Long.parseLong(JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE)));
	}
	public static float leerFloat(String msje) {
		return (Float.parseFloat(JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE)));
	}
	public static char leerChar(String msje) {
		return (JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE).charAt(0));
	}
	public static String leerString(String msje) {
		return(JOptionPane.showInputDialog(null,msje,"Dato de entrada",JOptionPane.INFORMATION_MESSAGE));
	}
	public static int seguirSino() {
		return JOptionPane.showConfirmDialog(null,"Deseas continuar","Capturando datos",JOptionPane.YES_NO_OPTION);
	}
	public static int leerEntero(String msje) {
		return (Integer.parseInt(JOptionPane.showInputDialog(null,msje,"Lectura int: ",JOptionPane.INFORMATION_MESSAGE)));
	}
	public static void imprimePantalla(String msje) {
		JOptionPane.showMessageDialog(null, msje, "Salida",JOptionPane.INFORMATION_MESSAGE);
	}
```

El paquete EntradaSalida contiene una sola clase que contiene varios métodos estáticos que permiten leer diferentes tipos de datos de entrada por consola, como números enteros, flotantes, caracteres y cadenas. También tiene un método para imprimir texto en la consola.

TDA

```java
package TDA;

public class Persona {
// atributos
	private String nomPer;
	private double telPers;
	private byte edadPers;
	private String nacioPer;
	private Fecha fechPer;
	private char genePer;
	//constructor vacio
	public Persona() {	}
	//constructor parametrizado
	public Persona(String nomPer, int telPers, byte edadPers, String nacioPer, Fecha fechPer, char genePer) {

		this.nomPer = nomPer;
		this.telPers = telPers;
		this.edadPers = edadPers;
		this.nacioPer = nacioPer;
		this.fechPer = fechPer;
		this.genePer = genePer;

	}
	//encapsulamiento
	public String getNomPer() {
		return nomPer;
	}
	public double getTelPers() {
		return telPers;
	}
	public void setTelPers(int telPers) {
		this.telPers = telPers;
	}
	public byte getEdadPers() {
		return edadPers;
	}
	public void setEdadPers(byte edadPers) {
		this.edadPers = edadPers;
	}
	public String getNacioPer() {
		return nacioPer;
	}
	public void setNacioPer(String nacioPer) {
		this.nacioPer = nacioPer;
	}
	public Fecha getFechPer() {
		return fechPer;
	}
	public void setFechPer(Fecha fechPer) {
		this.fechPer = fechPer;
	}
	public char getGenePer() {
		return genePer;
	}
	public void setGenePer(char genePer) {
		this.genePer = genePer;
	}
	public void setNomPer(String nomPer) {
		this.nomPer = nomPer;
	}
	//Generar el metodo toString
	@Override
	public String toString() {
		return "Persona [nomPer=" + nomPer + "telPers=" + telPers + "edadPers=" + edadPers + "nacioPer="
				+ nacioPer + "fechPer=" + fechPer + "genePer=" + genePer + "]";
	}

}
```

La clase Persona modela una persona, con atributos como nombre, edad, sexo, peso y altura. También tiene métodos para calcular el índice de masa corporal (IMC) y determinar si es mayor de edad.

```java
package TDA;

public class Circulo {
	private float radio;

	public Circulo() {}

	public Circulo(float radio) {
		this.radio=radio;
	}
	public float getRadio() {
		return radio;
	}
	public void setRadio(float radio) {
		this.radio = radio;
	}
	@Override
	public String toString() {
		return "Circulo [radio=" + radio + "]";
	}
	public float AreaCirculo() {
		return (float) (Math.PI*Math.pow(radio, 2));
	}
	public float PerimetroCirculo() {
		return (float) Math.PI*(radio*2);
	}
}
```

La clase Circulo modela un círculo, con atributos como el radio y el color. También tiene métodos para calcular el área y el perímetro del círculo.

```java
package TDA;

public class Rectangulo {
	private float base;
	private float altura;
	public Rectangulo() {}
	public Rectangulo(float base, float altura) {
		this.base=base;
		this.altura=altura;
	}
	public float getBase() {
		return base;
	}
	public void setBase(float base) {
		this.base = base;
	}
	public float getAltura() {
		return altura;
	}
	public void setAltura(float altura) {
		this.altura = altura;
	}
	@Override
	public String toString() {
		return "Rectangulo [base=" + base + ", altura=" + altura + "]
	}
	public float AreaRectangulo() {
		return (base*altura);
	}
	public float PerimetroRectangulo() {
		return ((base*altura)*2);
	}
}
```

La clase Rectangulo modela un rectángulo, con atributos como la base, la altura y el color. También tiene métodos para calcular el área y el perímetro del rectángulo.

```java
public class Triangulo {

	private float base;
	private float altura;
	private float lado1;
	private float lado2;
	private float catetoO;
	private float catetoA;

	public Triangulo() {}

	public Triangulo(float base, float altura, float lado1, float lado2, float catetoO, float catetoA) {
		this.base=base;
		this.altura=altura;
		this.lado1=lado1;
		this.lado2=lado2;
		this.catetoO=catetoO;
		this.catetoA=catetoA;
	}

	public float getBase() {
		return base;
	}
	public void setBase(float base) {
		this.base = base;
	}
	public float getAltura() {
		return altura;
	}
	public void setAltura(float altura) {
		this.altura = altura;
	}
	public float getLado1() {
		return lado1;
	}
	public void setLado1(float lado1) {
		this.lado1 = lado1;
	}
	public float getLado2() {
		return lado2;
	}
	public void setLado2(float lado2) {
		this.lado2 = lado2;
	}
	public float getCatetoO() {
		return catetoO;
	}

	public void setCatetoO(float catetoO) {
		this.catetoO = catetoO;
	}

	public float getCatetoA() {
		return catetoA;
	}

	public void setCatetoA(float catetoA) {
		this.catetoA = catetoA;
	}


	public float AreaTriangulo() {
		return (base*altura)/2;
	}
	public float PerimetroTriangulo() {
		return (base+lado1+lado2);
	}
	public float HipotenusaTriangulo() {
		return (float) (Math.sqrt(Math.pow(catetoO, 2)+Math.pow(catetoA, 2)));
	}
	public String TipoTriangulo() {
		if (base==lado1 && lado1==lado2)
			return "Equilatero";
		else if (base==lado1 || lado1==lado2 || base==lado2)
			return "Isoceles";
		else
			return "Escaleno";
```

La clase Triangulo modela un triángulo, con atributos como la base, la altura y el color. También tiene métodos para calcular el área y el perímetro del triángulo.

```java
package TDA;

public class Persona2 {
	private String nom;
	private byte edad;
	private char sex;
	private float peso;
	private float altura;
	public Persona2() {}
	public Persona2(String nom, byte edad, char sex, float peso, float altura) {
		this.nom=nom;
		this.edad=edad;
		this.sex=sex;
		this.peso=peso;
		this.altura=altura;
	}
	public Persona2(String nom, byte edad, char sex) {
		this.nom=nom;
		this.edad=edad;
		this.sex=sex;
	}
	public String getNom() {
		return nom;
	}
	public void setNom(String nom) {
		this.nom = nom;
	}
	public byte getEdad() {
		return edad;
	}
	public void setEdad(byte edad) {
		this.edad = edad;
	}
	public char getSex() {
		return sex;
	}
	public void setSex(char sex) {
		this.sex = sex;
	}
	public float getPeso() {
		return peso;
	}
	public void setPeso(float peso) {
		this.peso = peso;
	}
	public float getAltura() {
		return altura;
	}
	public void setAltura(float altura) {
		this.altura = altura;
	}
	public static byte calculaIMC(float peso, float altura) {
		float p;
		byte a=0;
		p=peso/((float)Math.pow(altura, 2));
		if(p<20)
			a=-1;
		else if (p>=20 && p<=25)
			a=0;
		else if (p>25)
			a=1;
		return a;
	}
	public static boolean esMayorDeEdad(byte edad) {
		return (edad>=18);
	}
	public String toString() {
		return "Persona [nom=" + nom + ", edad=" + edad + ", sex=" + sex + ", peso=" + peso + ", altura=" + altura
				+ "]";
	}
```

La clase Persona2 es similar a la clase Persona, pero tiene métodos adicionales para calcular el IMC y determinar si la persona está en su peso ideal.

```java
package TDA;

public class RentaAutos {
	private char tamañoAuto;
	private byte dias;
	private float kilometros;
	public RentaAutos() {}
	public RentaAutos(char tamañoAuto, byte dias, float kilometros) {
		this.tamañoAuto = tamañoAuto;
		this.dias = dias;
		this.kilometros = kilometros;
	}
	public char getTamañoAuto() {
		return tamañoAuto;
	}
	public void setTamañoAuto(char tamañoAuto) {
		this.tamañoAuto = tamañoAuto;
	}
	public byte getDias() {
		return dias;
	}
	public void setDias(byte dias) {
		this.dias = dias;
	}
	public float getKilometros() {
		return kilometros;
	}
	public void setKilometros(float kilometros) {
		this.kilometros = kilometros;
	}
	public double DeterminarTarifa() {
		double tarifa=0;
		if(tamañoAuto=='P'||tamañoAuto=='p')
			tarifa=200*dias;
		else if (tamañoAuto=='M'||tamañoAuto=='m')
			tarifa=350*dias;
		else if (tamañoAuto=='G'||tamañoAuto=='g')
			tarifa=450*dias;
		return tarifa;
	}
	public double CalcularCosto() {
		double costo=0;
		if(tamañoAuto=='P'||tamañoAuto=='p')
			costo=kilometros*20;
		else if (tamañoAuto=='M'||tamañoAuto=='m')
			costo=kilometros*30;
		else if (tamañoAuto=='G'||tamañoAuto=='g')
			costo=kilometros*40;
		return costo;
	}
	public double CalcularMonto() {
		double monto=CalcularCosto();
		if(kilometros>10)
			monto+=CalcularCosto()*0.025;
		monto+=DeterminarTarifa();
		return monto;
	}
}
```

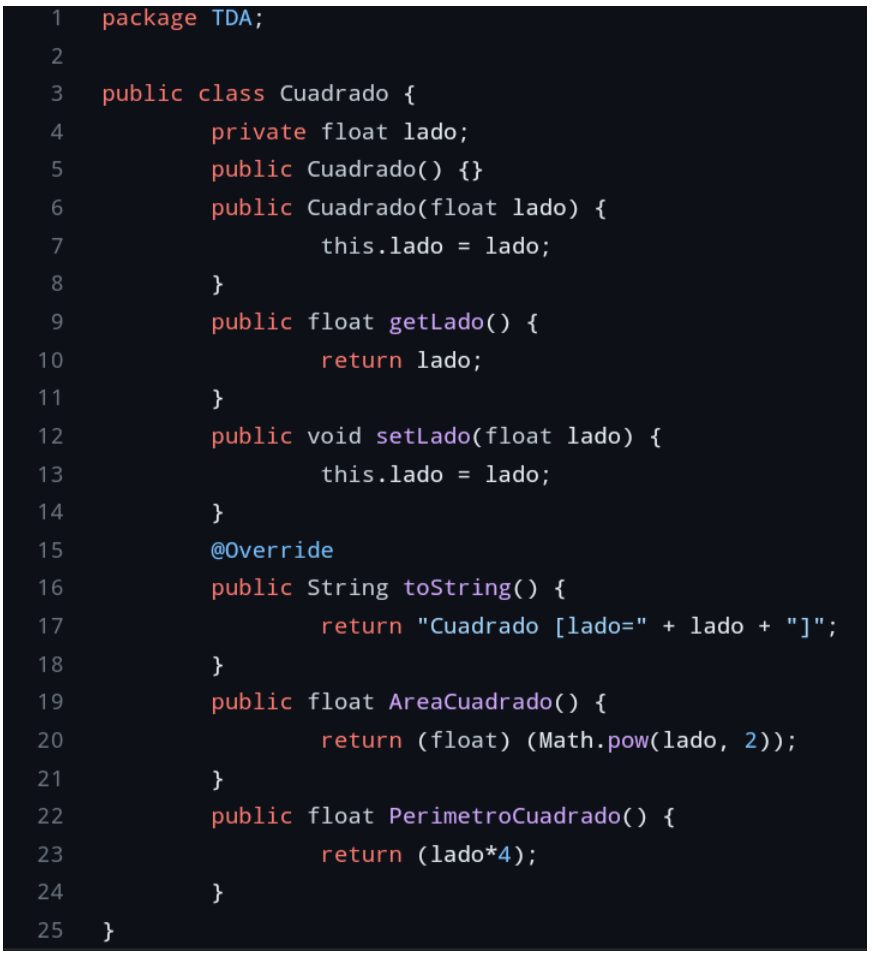
La clase RentaAutos modela una renta de autos, con atributos como el tamaño del auto, los días de renta y los kilómetros recorridos. También tiene métodos para determinar la tarifa, calcular el costo por kilómetro y el monto a pagar por la renta

```java
package TDA;

public class Autor {
	private String nombre;
	private String apellido;
	public Autor() {}
	public Autor(String nombre, String apellido) {
		this.nombre=nombre;
		this.apellido=apellido;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellido() {
		return apellido;
	}
	public void setApellido(String apellido) {
		this.apellido = apellido;
	}
	@Override
	public String toString() {
		return "Autor [Nombre= " + nombre + ", Apellido=" + apellido + "]";
	}
```

El código define una clase llamada Autor en el paquete TDA (acrónimo de Tipo de Dato Abstracto). Esta clase tiene dos atributos privados, nombre y apellido, que se pueden establecer mediante el constructor o los métodos de acceso (getters y setters). La clase también tiene un método toString() que devuelve una cadena que representa al objeto Autor con su nombre y apellido.

En resumen, la clase Autor es un tipo de dato que representa a un autor de un libro o cualquier otra entidad que requiera información de nombre y apellido.



```java
package TDA;

public class Cuadrado {
	private float lado;
	public Cuadrado() {}
	public Cuadrado(float lado) {
		this.lado = lado;
	}
	public float getLado() {
		return lado;
	}
	public void setLado(float lado) {
		this.lado = lado;
	}
	@Override
	public String toString() {
		return "Cuadrado [lado=" + lado + "]";
	}
	public float AreaCuadrado() {
		return (float) (Math.pow(lado, 2));
	}
	public float PerimetroCuadrado() {
		return (lado*4);
	}
}
```

El código define una clase llamada Cuadrado en el paquete TDA. La clase tiene un atributo privado llamado lado y dos constructores, uno sin argumentos y otro que acepta un argumento float para inicializar el atributo lado.

La clase también tiene métodos de acceso (getters y setters) para el atributo lado, y un método toString() que devuelve una cadena que representa al objeto Cuadrado con su lado.

Además, la clase define dos métodos específicos del cuadrado: el método AreaCuadrado() que calcula el área del cuadrado mediante el lado elevado al cuadrado, y el método PerimetroCuadrado() que calcula el perímetro del cuadrado multiplicando el lado por cuatro.

En resumen, la clase Cuadrado es un tipo de dato que representa un cuadrado y proporciona métodos para calcular su área y perímetro

```java
package TDA;

public class Fecha {
	//atributos
		private byte dia;
		private byte mes;
		private short año;
		//Constructor vacio
		public Fecha() {}
		//Constructor parametrizado
		public Fecha(byte dia,byte mes,short año)
		{
			this.dia=dia;
			this.año=año;
			this.mes=mes;
		}
		//Encapsular
		public void setDia(byte dia) {
			this.dia=dia;
		}
		public byte getDia() {
			return dia;
		}
		public byte getMes() {
			return mes;
		}
		public void setMes(byte mes) {
			this.mes = mes;
		}
		public short getAño() {
			return año;
		}
		public void setAño(short año) {
			this.año = año;
		}
		@Override
		public String toString() {
			return "Fecha [dia=" + dia + ", mes=" + mes + ", año=" + año + "]
		}
}
```

El código define una clase llamada Fecha en el paquete TDA. La clase tiene tres atributos privados de tipo byte para el día y el mes, y un atributo privado de tipo short para el año. Además, la clase tiene dos constructores, uno sin argumentos y otro que acepta tres argumentos de tipo byte y short para inicializar los atributos de día, mes y año.

La clase también tiene métodos de acceso (getters y setters) para los tres atributos de fecha, y un método toString() que devuelve una cadena que representa al objeto Fecha con su día, mes y año.

En resumen, la clase Fecha es un tipo de dato que representa una fecha y proporciona métodos para acceder y modificar sus atributos

```java
package TDA;

public class Libro {
	private String titulo;
	private Autor autor;
	private int isbn;
	private short paginas;
	public Libro() {}
	public Libro(String titulo, Autor autor, int isbn, short paginas) {
		this.titulo=titulo;
		this.autor=autor;
		this.isbn=isbn;
		this.paginas=paginas;
	}
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public Autor getAutor() {
		return autor;
	}
	public void setAutor(Autor autor) {
		this.autor = autor;
	}
	public int getIsbn() {
		return isbn;
	}
	public void setIsbn(int isbn) {
		this.isbn = isbn;
	}
	public short getPaginas() {
		return paginas;
	}
	public void setPaginas(short paginas) {
		this.paginas = paginas;
	}
	@Override
	public String toString() {
		return "Libro [titulo= " + titulo + ", autor= " + autor + ", isbn= " + isbn + ", paginas= " + paginas + "]"
	}
}
```

El código define una clase llamada Libro en el paquete TDA. La clase tiene cuatro atributos privados: un atributo de tipo String llamado titulo para el título del libro, un atributo de tipo Autor que representa el autor del libro, un atributo de tipo int para el número ISBN del libro, y un atributo de tipo short para la cantidad de páginas del libro.

La clase tiene dos constructores, uno sin argumentos y otro que acepta cuatro argumentos para inicializar los atributos del libro.

La clase también tiene métodos de acceso (getters y setters) para cada uno de los atributos del libro, y un método toString() que devuelve una cadena que representa al objeto Libro con su título, autor, número ISBN y cantidad de páginas.

En resumen, la clase Libro es un tipo de dato que representa un libro y proporciona métodos para acceder y modificar sus atributos. El objeto Autor se utiliza como un atributo de la clase Libro para representar al autor del libro.

TestClasesObjetos

PruebaFigura.java

```java
port javax.swing.JOptionPane;
port EntradaSalida.Tools;
port TDA.Cuadrado;
port TDA.Rectangulo;
port TDA.Circulo;
port TDA.Triangulo;

blic class PruebaFiguras {
	public static String Boton(String menu) {
		String valores[]=menu.split(",");
		int n;
		n=JOptionPane.showOptionDialog(null, "Seleccione dando clic","Menu de figuras",JOptionPane.NO_OPTION,
				JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
		return valores[n];
	}
	public static void MenuFiguras(String menu) {
		String sel="";
		do {
			sel=Boton(menu);
			switch(sel) {
			case "Cuadrado":
				Cuadrado cuadrado=new Cuadrado();
				cuadrado.setLado(Tools.leerFloat("Dame el lado"));
				MenuCuadrado("Perimetro,Area,Salir",cuadrado.getLado());
				break;
			case "Rectangulo":
				Rectangulo rectangulo=new Rectangulo();
				rectangulo.setBase(Tools.leerFloat("Dame la base"));
				rectangulo.setAltura(Tools.leerFloat("Dame la altura"));
				MenuRectangulo("Perimetro,Area,Salir", rectangulo.getBase(), rectangulo.getAltura());
				break;
			case "Circulo":
				Circulo circulo=new Circulo();
				circulo.setRadio(Tools.leerFloat("Dame el radio:"));
				MenuCirculo("Perimetro,Area,Salir", circulo.getRadio());
				break;
			case "Triangulo":
				Triangulo triangulo=new Triangulo();
				triangulo.setBase(Tools.leerFloat("Dame la base: "));
				triangulo.setAltura(Tools.leerFloat("Dame la altura: "));
				triangulo.setLado1(Tools.leerFloat("Dame el primer lado: "));
				triangulo.setLado2(Tools.leerFloat("Dame el segundo lado: "));
				triangulo.setCatetoO(Tools.leerFloat("Dame el cateto opuesto: "));
				triangulo.setCatetoA(Tools.leerFloat("Dame el cateto adyacente: "));
				MenuTriangulo("Area,Perimetro,Hipotenusa,Tipo de triangulo,Salir",triangulo.getBase(),triangulo.getAltura(),trian
						triangulo.getLado2(),triangulo.getCatetoO(),triangulo.getCatetoA());
				break;
			case "Salir":
				Tools.imprimePantalla("Adiós");
				break;
			}
		}while(!sel.equalsIgnoreCase("Salir"));
	}
	public static void main(String[] args) {
		MenuFiguras("Cuadrado,Rectangulo,Circulo,Triangulo,Salir");
	}
	public static void MenuCuadrado(String menu, float lado) {
		String sel="";
		Cuadrado cua=new Cuadrado(lado);
		do {
			sel=Boton(menu);
			switch (sel) {
			case "Perimetro":
				Tools.imprimePantalla("El Perimetro es: \n"+cua.PerimetroCuadrado());
				break;
			case "Area":
				Tools.imprimePantalla("El Area es: \n"+cua.AreaCuadrado());
				break;
			case "Salir":
				Tools.imprimePantalla("Puedes seguir probando otra figura");
				break;
			}
		}while(!sel.equalsIgnoreCase("Salir"));
	}
	public static void MenuRectangulo(String menu, float base, float altura) {
		String sel="";
		Rectangulo rec=new Rectangulo(base, altura);
		do {
			sel=Boton(menu);
			switch (sel) {
			case "Perimetro":
				Tools.imprimePantalla("El perimetro es: \n"+rec.PerimetroRectangulo());
				break;
			case "Area":
				Tools.imprimePantalla("EL area es: \n"+rec.AreaRectangulo());
				break;
			case "Salir":
				Tools.imprimePantalla("Puedes seguir probando otra figura");
				break;
			}
		}while(!sel.equalsIgnoreCase("Salir"));
```

```
                        triangulo.setCatetoA(Tools.leerFloat("Dame el cateto adyacente: "));
                        MenuTriangulo("Area,Perimetro,Hipotenusa,Tipo de triangulo,Salir",triangulo.getBase(),triangulo.getAltura(),t
                                        triangulo.getLado2(),triangulo.getCatetoO(),triangulo.getCatetoA());
                        break;
                case "Salir":
                        Tools.imprimePantalla("Adiós");
                        break;
                }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
    public static void main(String[] args) {
        MenuFiguras("Cuadrado,Rectangulo,Circulo,Triangulo,Salir");
    }
    public static void MenuCuadrado(String menu, float lado) {
        String sel="";
        Cuadrado cua=new Cuadrado(lado);
        do {
                sel=Boton(menu);
                switch (sel) {
                case "Perimetro":
                        Tools.imprimePantalla("El Perimetro es: \n"+cua.PerimetroCuadrado());
                        break;
                case "Area":
                        Tools.imprimePantalla("El Area es: \n"+cua.AreaCuadrado());
                        break;
                case "Salir":
                        Tools.imprimePantalla("Puedes seguir probando otra figura");
                        break;
                }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
    public static void MenuRectangulo(String menu, float base, float altura) {
        String sel="";
        Rectangulo rec=new Rectangulo(base, altura);
        do {
                sel=Boton(menu);
                switch (sel) {
                case "Perimetro":
                        Tools.imprimePantalla("El perimetro es: \n"+rec.PerimetroRectangulo());
                        break;
                case "Area":
                        Tools.imprimePantalla("EL area es: \n"+rec.AreaRectangulo());
                        break;
                case "Salir":
                        Tools.imprimePantalla("Puedes seguir probando otra figura");
                        break;
                }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
    public static void MenuCirculo(String menu, float radio) {
        String sel="";
        Circulo cir=new Circulo(radio);
        do {
                sel=Boton(menu);
                switch (sel) {
                case "Perimetro":
                        Tools.imprimePantalla("El perimetro es: \n"+cir.PerimetroCirculo());
                        break;
                case "Area":
                        Tools.imprimePantalla("EL area es: \n"+cir.AreaCirculo());
                        break;
                case "Salir":
                        Tools.imprimePantalla("Puedes seguir probando otra figura");
                        break;
                }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
    public static void MenuTriangulo(String menu, float base, float altura, float lado1, float lado2, float catetoO, float catetoA) {
        String sel="";
        Triangulo tri=new Triangulo(base, altura, lado1, lado2, catetoO, catetoA);
        do {
                sel=Boton(menu);
                switch (sel) {
                case "Area":
                        Tools.imprimePantalla("El area es: \n"+tri.AreaTriangulo());
                        break;
                case "Perimetro":
                        Tools.imprimePantalla("El perimetro es: \n"+tri.PerimetroTriangulo());
                        break;
                case "Hipotenusa":
                        Tools.imprimePantalla("La hipotenusa es: \n"+tri.HipotenusaTriangulo());
                        break;
                case "Tipo de triangulo":
                        Tools.imprimePantalla("El triangulo es: \n"+tri.TipoTriangulo());
                        break;
                case "Salir":
                        Tools.imprimePantalla("Puedes seguir probando otra figura");
                        break;
                }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
```

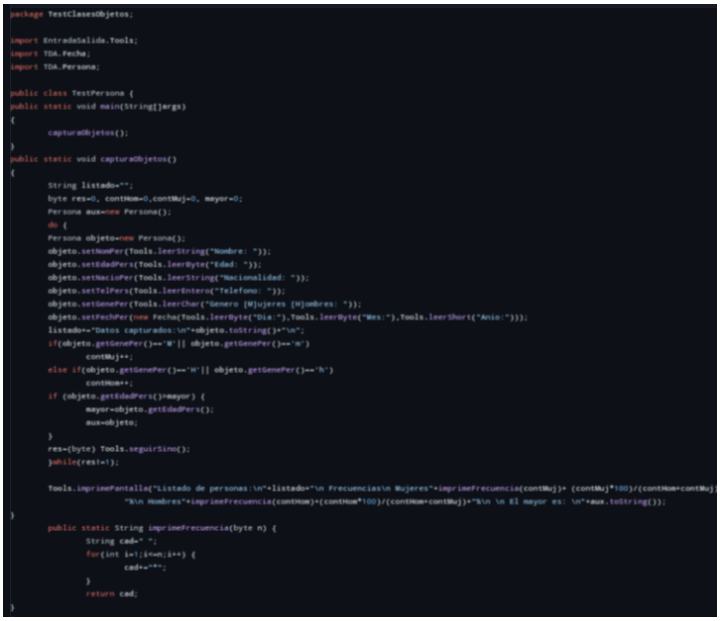
TestAutor.java

```
ackage TestClasesObjetos;

mport EntradaSalida.Tools;
mport TDA.Autor;
mport TDA.Libro;

ublic class TestAutor {
	public static void main(String[]args) {
		capturaLibros();
	}
	public static void capturaLibros() {
		String listado="";
		byte res=0;
		short menor=32767;
		Libro libMayor=new Libro();
		do {
			Libro libros=new Libro();
			libros.setTitulo(Tools.leerString("Titulo: "));
			libros.setAutor(new Autor(Tools.leerString("Nombre: "),Tools.leerString("Apellido: ")));
			libros.setIsbn(Tools.leerEntero("ISBN: "));
			libros.setPaginas(Tools.leerShort("Paginas: "));
			listado+="Libros capturados:\n"+libros.toString()+"\n";
			res=(byte) Tools.seguirSino();
			if (libros.getPaginas()<menor) {
				menor=libros.getPaginas();
				libMayor=libros;
			}
		}while (res!=1);
		Tools.imprimePantalla("Listado de libros"+listado+"\n \n El Libro con el mayor numero de paginas es:\n"+libMayor.toString());
	}
```

TestPersona.java



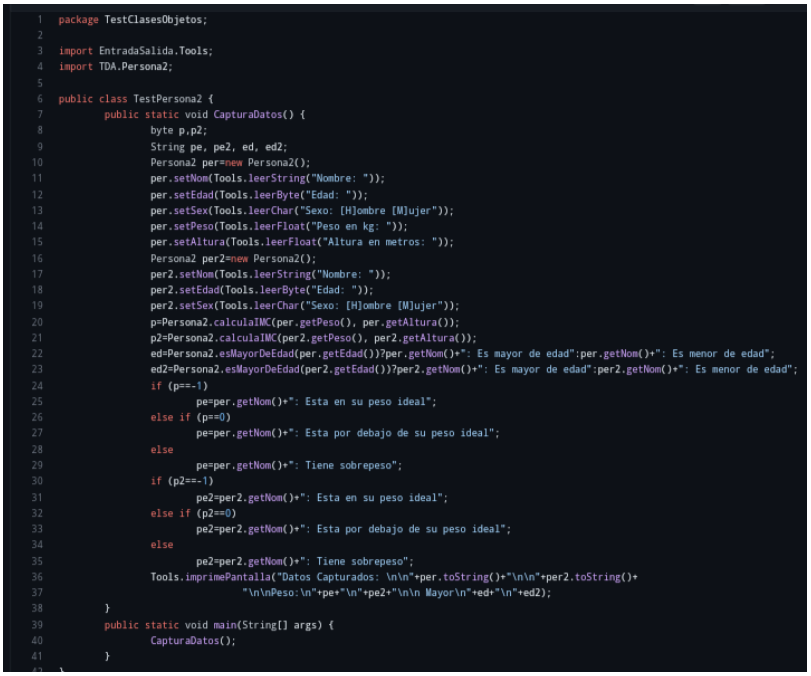
```
package TestClasesObjetos;

import EntradaSalida.Tools;
import TDA.Fecha;
import TDA.Persona;

public class TestPersona {
public static void main(String[]args)
{
        capturaObjetos();
}
public static void capturaObjetos()
{
        String listado="";
        byte res=0, contHom=0,contMuj=0, mayor=0;
        Persona aux=new Persona();
        do {
        Persona objeto=new Persona();
        objeto.setNomPer(Tools.leerString("Nombre: "));
        objeto.setEdadPers(Tools.leerByte("Edad: "));
        objeto.setNacioPer(Tools.leerString("Nacionalidad: "));
        objeto.setTelPers(Tools.leerEntero("Telefono: "));
        objeto.setGenePer(Tools.leerChar("Genero [M]ujeres [H]ombres: "));
        objeto.setFechPer(new Fecha(Tools.leerByte("Dia:"),Tools.leerByte("Mes:"),Tools.leerShort("Anio:")));
        listado+="Datos capturados:\n"+objeto.toString()+"\n";
        if(objeto.getGenePer()=='M'|| objeto.getGenePer()=='m')
                contMuj++;
        else if(objeto.getGenePer()=='H'|| objeto.getGenePer()=='h')
                contHom++;
        if (objeto.getEdadPers()>mayor) {
                mayor=objeto.getEdadPers();
                aux=objeto;
        }
        res=(byte) Tools.seguirSino();
        }while(res!=1);

        Tools.imprimePantalla("Listado de personas:\n"+listado+"\n Frecuencias\n Mujeres"+imprimeFrecuencia(contMuj)+ (contMuj*100)/(contHom+contMuj)
                        "%\n Hombres"+imprimeFrecuencia(contHom)+(contHom*100)/(contHom+contMuj)+"%\n \n El mayor es: \n"+aux.toString());
}
        public static String imprimeFrecuencia(byte n) {
                String cad=" ";
                for(int i=1;i<=n;i++) {
                        cad+="*";
                }
                return cad;
}
```

TestPersona2.java



```
1  package TestClasesObjetos;
2
3  import EntradaSalida.Tools;
4  import TDA.Persona2;
5
6  public class TestPersona2 {
7          public static void CapturaDatos() {
8                  byte p,p2;
9                  String pe, pe2, ed, ed2;
10                 Persona2 per=new Persona2();
11                 per.setNom(Tools.leerString("Nombre: "));
12                 per.setEdad(Tools.leerByte("Edad: "));
13                 per.setSex(Tools.leerChar("Sexo: [H]ombre [M]ujer"));
14                 per.setPeso(Tools.leerFloat("Peso en kg: "));
15                 per.setAltura(Tools.leerFloat("Altura en metros: "));
16                 Persona2 per2=new Persona2();
17                 per2.setNom(Tools.leerString("Nombre: "));
18                 per2.setEdad(Tools.leerByte("Edad: "));
19                 per2.setSex(Tools.leerChar("Sexo: [H]ombre [M]ujer"));
20                 p=Persona2.calculaIMC(per.getPeso(), per.getAltura());
21                 p2=Persona2.calculaIMC(per2.getPeso(), per2.getAltura());
22                 ed=Persona2.esMayorDeEdad(per.getEdad())?per.getNom()+": Es mayor de edad":per.getNom()+": Es menor de edad";
23                 ed2=Persona2.esMayorDeEdad(per2.getEdad())?per2.getNom()+": Es mayor de edad":per2.getNom()+": Es menor de edad";
24                 if (p==-1)
25                         pe=per.getNom()+": Esta en su peso ideal";
26                 else if (p==0)
27                         pe=per.getNom()+": Esta por debajo de su peso ideal";
28                 else
29                         pe=per.getNom()+": Tiene sobrepeso";
30                 if (p2==-1)
31                         pe2=per2.getNom()+": Esta en su peso ideal";
32                 else if (p2==0)
33                         pe2=per2.getNom()+": Esta por debajo de su peso ideal";
34                 else
35                         pe2=per2.getNom()+": Tiene sobrepeso";
36                 Tools.imprimePantalla("Datos Capturados: \n\n"+per.toString()+"\n\n"+per2.toString()+
37                                 "\n\nPeso:\n"+pe+"\n"+pe2+"\n\n Mayor\n"+ed+"\n"+ed2);
38         }
39         public static void main(String[] args) {
40                 CapturaDatos();
41         }
42 }
```

TestRentaAutos.java

```java
package TestClasesObjetos;

import EntradaSalida.Tools;
import TDA.RentaAutos;

public class TestRentaAutos {
	public static void main(String[] args) {
		CapturaRentaAutos();
	}
	public static void CapturaRentaAutos() {
		String cad= "Renta de autos\n";
		byte res=0;
		do {
		RentaAutos auto=new RentaAutos();
		auto.setTamañoAuto(Tools.leerChar("¿Que tipo de auto va a rentar?: [P]equeño, [M]ediano, [G]rande"));
		auto.setDias(Tools.leerByte("¿Cuantos dias lo va a rentar?:"));
		auto.setKilometros(Tools.leerFloat("Kilometros recorridos:"));
		cad+="Tipo de Auto: "+auto.getTamañoAuto()+"\n";
		cad+="Dias alquilados: "+auto.getDias()+"\n";
		cad+="Kilometros Recorridos: "+auto.getKilometros()+"\n";
		cad+="Tarifa: "+auto.DeterminarTarifa()+"\n";
		cad+="Costo por kilometro: "+auto.CalcularCosto()+"\n";
		cad+="Monto a pagar: "+auto.CalcularMonto();
		res=(byte)Tools.seguirSino();
		}while(res!=1);
		Tools.imprimePantalla("Listado de persona:\n"+cad);
```

El paquete TestClasesObjetos es un conjunto de clases que se utilizan para probar el funcionamiento de otras clases, en particular, de las clases que se encuentran en los paquetes TDA y EntradaSalida.

Cada clase en este paquete contiene uno o varios métodos que se encargan de crear objetos de las clases que se quieren probar y realizar operaciones sobre ellos. Estos métodos se pueden ejecutar desde la línea de comandos o desde un programa externo que invoque a la clase principal.

Los nombres de las clases en este paquete suelen indicar qué clase están probando. Por ejemplo, TestPersona y TestPersona2 están probando la clase Persona, mientras que TestRentaAutos está probando la clase RentaAutos

Conclusión:

La practica pone en aplicación los conceptos que ibtevrab a la programación orientada a objetos, por ello forman una experiencia en la

que los conocimientos interfieren para resolver los diversos problemas que se pueden resolver, siempre tomando en cuenta buenas prácticas

Bibliografía:

KevinDCCsHeOs. (2023). TemaClasesObjetos. GitHub. https://github.com/KevinDCCsHeOs/TemaClasesObjetos

MA.J.MTZ CASTILLO.(2023). TEMA 2 CLASES Y OBJETOS.Padlet.

https://padlet.com/mtzcastillo2023/tema-2-clases-y-objetos-f5ath9k63vv88uta