Programación Orientada a Objetos

Ingeniería Informática

Métodos

Integrantes:

Gutiérrez Montaño Juan Luis 22010601

Hernández Heredia Kevin 22010603

Grupo:

2a3B

Fecha de entrega:

16/abril/2023

Introducción:

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en la utilización de objetos, los cuales se definen a través de atributos y métodos. En Java, uno de los lenguajes más utilizados para aplicaciones de POO, es importante entender cómo funcionan los métodos y cómo se pueden utilizar para mejorar el diseño de un programa.

En este sentido, el presente texto aborda los principales conceptos relacionados con la definición, estructura y uso de métodos en Java. Se explorarán temas como el valor de retorno, la declaración de métodos de clase e instancia, el ámbito y tiempo de vida de las variables, la sobrecarga de métodos y la utilización de constructores y destructores.

Con la comprensión de estos temas, se podrá ampliar el conocimiento sobre POO en Java y mejorar la capacidad para diseñar programas más eficientes y funcionales.

Competencia especifica:

Comprende y aplica los diferentes tipos de métodos, tomando en cuenta el ámbito y tiempo de vida de los datos durante la ejcucion de un programa.

Marco teórico:

Para empezar, es importante entender qué es un método y cómo se estructura. Un método se define como un conjunto de instrucciones que realizan una tarea específica y que pueden recibir parámetros de entrada y devolver un valor de salida. La estructura de un método en Java incluye el nombre del método, los parámetros de entrada, la declaración del tipo de valor de retorno y el cuerpo del método que contiene las instrucciones.

El valor de retorno es otro concepto importante en los métodos de Java. Este valor representa el resultado de la tarea que realiza el método y puede ser de cualquier tipo de datos en Java. En algunos casos, un método no devuelve ningún valor, y en su lugar, simplemente realiza una tarea.

La declaración de un método en Java puede ser de dos tipos: de clase o de instancia. Un método de clase se aplica a la clase en general, mientras que un método de instancia se aplica a una instancia específica de una clase. Además, es importante comprender el ámbito y el tiempo de vida de las variables que se utilizan en los métodos.

Los argumentos y el paso de parámetros son conceptos importantes en los métodos de Java, ya que permiten que los métodos reciban información para realizar su tarea. En Java, el paso de parámetros puede ser por valor o por referencia, lo que puede afectar el comportamiento del método.

El puntero this es otra característica importante de los métodos de Java. Este puntero se utiliza para hacer referencia a la instancia actual de la clase y se utiliza con frecuencia en los métodos de instancia.

La sobrecarga de métodos es un concepto avanzado en Java que permite que un método tenga varios parámetros diferentes. Los constructores y destructores son métodos especiales que se utilizan para inicializar y destruir objetos en Java, respectivamente.

Material y Equipo:

- ✓ Computadora
- ✓ Eclipse IDE for Java Developers – 2023 – 03
- ✓ Java SE Development kit 19.0.2

Desarrollo de la practica:

1)Diseñar un metodo de clase y probar en una unidad ejecutable.
Se tiene un salón en forma rectangular, de 7 metros de largo por 6.5 metros de ancho. Además, se cuenta con dos alfombras una de 3.8 de largo por 4.6 de ancho y 4.5 de largo por 2.3 de ancho, también rectangulares, que se colocarán sobre el piso de dicho salón. Se desea saber que parte del piso quedará cubierta y qué parte no, para ayudar a decidir si se compra o no otras alfombras.
PD: Le recuerdo que ya tiene una clase que puede reutilizar para dar solución al ejercicio.
Al reutilizar una clase que ya se tenia lista se identifican los atributos que esten completes, en este caso lo estan, se identifican los metodos para crear, en este caso solo se agregara un metodo el cual utilizara el metodo de calcularArea, se

crea otra clase donde sera probado y se crean 3 objetos, los cuales seran las alfombras, despues llama al metodo que se creo y se le introducen los 3 objetos y regresara el area cubierta y el area restante.

Rectangulo
-float base -float altura
+Rectangulo() +Rectangulo(float base, float altura) +float getBase() +void setBase(float base) +float getAltura() +void setAltura(float altura) +String toString() +float AreaRectangulo() +float PerimetroRectangulo() + static String AreaCubierta(Rectangulo salon, Rectangulo alf1, Rectangulo alf2)

TestSalon
+static void CapturaObjetos +static void main(String[] args)

2)Clase Película

Realizar un programa en Java que defina una clase Película con los siguientes atributos privados:

• Nombre: es el nombre de la película y es de tipo String.

• Director: representa el nombre del director de la película y es de tipo String.

• Género: es el género de la película, con los siguientes valores: ACCIÓN, COMEDIA, DRAMA y SUSPENSO.

• Duración: duración de la película en minutos, con los siguientes valores :60, 115,120, 150,180,191, 210,240, y 270.

• Año: representa el año de realización de la película.

• Calificación: es la valoración de la película por parte de sus usuarios y es de tipo byte, con los siguientes valores 0..10.

✓ Se debe definir un constructor público para la clase, que recibe como parámetros los valores de todos sus atributos. También se deben definir los siguientes métodos:

✓ Métodos get y set para cada atributo y con los derechos de acceso privados

para los set y públicos para los get.

✓ Un método que muestre en pantalla todos los valores de los atributos.

✓ Un método privado boolean esPeliculaEpica(), el cual devuelve un valor verdadero si la duración de la película es mayor o igual a tres horas, en caso contrario devuelve falso.

✓ Un método privado String calcularValoración(), el cual según la tabla 1 devuelve una valoración subjetiva.

TABLA VALORACION DE LA PELICULA

CALIFICACION	VALORACION	CALIFICACION	VALORACION
0-2	Muy mala	7-8	Buena
2-5	Mala	8-10	Excelente
5-7	Regular		

- ✓ El método privado boolean esSimilar() compara la película actual con otra película que se le pasa como parámetro. Si ambas películas son del mismo género y tienen la misma valoración, devuelve verdadero; en caso contrario, devuelve falso.
- ✓ Definir un método privado denominado imprimir Cartel que muestra en pantalla los datos de la película en el siguiente formato:

-------- Título ----------

*** (valoración)

Género, año

Director

La valoración se debe convertir a una cantidad determinada de asteriscos (*), recuerde que ya tiene un metodo usado en frecuencia de vocales.

En una clase que contenga una unidad ejecutable prueba lo siguiente: Diseñe un metodo de clase crear 2 instancias tipo película;

determinar si son películas épicas; calcular su respectiva valoración y determinar si son similares e imprimir ambos objetos. Mostrar en una ventana emergente los resultados de la ejecución del método main

Objeto	Nombre	Director	Genero	Duracion	año	Calificacion	Épica	Similares
peli1	Gandhi	Richard Attenborough	Drama	191	1982	8		
peli2	Thor	Kenneth Branagh	Acción	115	2011	7		

En primer lugar se identifican atributos con su respectivo tipo de dato, lo que serian nombre, director, genero, duracion, año y calificacion, despues cuantos constructores se requieren y con que atributos se hara cada uno, la creacion de los set y get de cada atributo, seguido de los metodos que se requieren y que recibe como parametros y si retornan algo o no, en este caso se hara un solo constructor con todos los atributos, el set y get de cada atributo, 9 metodos, 4 de ellos privados y 5 publicos, despues se crea la clase donde se ponen a prueba.

Pelicula
-String nomPel -String nomDirPel -String genPel -short durPel -short añoPel -byte calPel
+Pelicula(String nomPel, String nomDirPel, Stirng genPel, short durPel, short añoPel, byte calPel) +String getPel() -void setNomPel(String nomPel) +String getNomDirPel() -void setNomDirPel(String nomDirPel) +String getGenPel() -void setGenPel(Stirng genPel) +short getDurPel() -void setDurPel(short durPel) +short getAñoPel() -void setAñoPel(short añoPel) +byte getCalPel() -void setCalPel(byte calPel) +String toString() -boolean esPeliculaEpica() -String CalcularValoracion() -static boolean esSimilar(Pelicula pel2)

-String imprimirCartel() +static String imprimeFrecuencia(byte n) +boolean esPeliculaEpica2() +String CalcularValoracion2() +static esSimilar2(Pelicula pel2) +String imprimirCartel()

TestPeliculas
+static void main(String[] args) +static void CapturaPeliculas()

3)clase ConversorMetros

Realizar un programa en Java que permita realizar las siguientes conversiones de unidades de longitud, con el siguiente atributo privado y constantes :

• Atributo llamado metros (de tipo double) para definir la cantidad de metros a convertir a diferentes unidades de longitud

Constantes:

• METROS_CM = 100
• METROS_MILIM = 1000
• METROS_PULGADAS = 39.37
• METROS_PIES = 3.28
• METROS_YARDAS = 1.09361

La clase tiene un constructor para inicializar la cantidad de metros a convertir y un conjunto de métodos private para realizar diferentes conversiones, cada uno con su valor de retorno correspondiente.

• Metros a centímetros.
• Metros a milímetros.
• Metros a pulgadas.
• Metros a pies.
• Metros a yardas

En una clase que contenga una unidad ejecutable presentar un menú que permita crear instancias y seleccionar el tipo de conversión a realizar.

Comenzamos identificando que solo se requiere un atributo el cual es metros, pero se requieren 5 constantes, se realiza solo un constructor con el parámetro de metros de tipo double, se crean 5 métodos privados, uno para cada conversión de metros, para su uso en alguna otra clase se crea un puente o métodos públicos donde llamaran a los privados, después creamos la clase del test, donde pondremos a prueba la clase ConversorMetros, en este caso decidimos utilizar un menú desplegable y de botones.

ConversorMetros
-double metros final byte METROS_CM final short METROS_MILIM final double METROS_PULGADAS final double METROS_PIES final double METROS_YARDAS
+ConversorMetros(double metros) -double Metros_A_Centimetros() -double Metros_A_Milimetros() -double Metros_A_Pulgadas() -double Metros_A_Pies() -double Metros_A_Yardas() +double Metros_A_Centimetros2() +double Metros_A_Milimetros2() +double Metros_A_Pulgadas2() +double Metros_A_Pies2() +double Metros_A_Yardas2()

TestConversorMetros
+static void main(String[] args) +static void CapturaMetros(String menu) +static String Desplegable(String menu) +static String Boton(String menu)

4)clase Pedido

Realizar un programa en Java que permita calcular el pedido que realiza un cliente en un restaurante.

Los pedidos de un restaurante están conformados por las siguientes partes:

- Un primer plato(“Crema de espinacas ”,” Ensalada de verduras ”,”Crema de brócoli”,” Caldo tlalpeño”,” Sopa mixteca ”).
- Un segundo plato(“Filete de Pescado”, ”Milanesa de Pollo”,” Bistec a la mexicana”,”

Pollo en escabeche”,” Carne asada”,” Lomo relleno”,” Pollo a la plancha”).

- Una bebida (“Coca cola”,” Pepsi”,” Naranjada ”,”Limonada ”,” Agua de

sabor”).

• Un postre (“Pastel helado”,” Helado”,” Fresas con crema”,” Plátanos fritos ”,”Flan casero”,” Gelatina”).

Cada uno de dichas partes tiene un nombre y un precio. Se requiere definir métodos sobrecargados para calcular el valor del pedido dependiendo si el cliente solicita:

✓ Un primer plato y una bebida.

✓ Un primer plato, un segundo plato y una bebida.

✓ Un primer plato, un segundo plato, una bebida y un postre.

En una clase que contenga una unidad ejecutable presentar los costos del pedido solicitado (imprimir en una ventana emergente).

Comenzamos creando la clase Pedido, y en este caso no requiere de ningún atributo o constructor, utiliza métodos sobrecargados, se crean 3 metodos sobrecargados cada uno con diferentes parámetros de entrada, deben ser métodos estáticos para poder introducir los datos desde otra clase, después se requiere de métodos que calculen el precio de los platillos, son 4 metodos privados y estáticos y se crea la clase TestPedido donde se pondrá a prueba, para la prueba hice dos métodos que presentan menú desplegable donde uno lo utilice para saber que pedido quiere uno y el otro para que escogiera sus platillos.

Pedido
+static void CalcularPedido(String primPlato, String bebida) +static void CalcularPedido(String primPlato, String segPlato, String bebida) +static void CalcularPedido(String primPlato, String segPlato, String bebida, String bebida) -static byte calculaPrimPlato(String primPlato) -static byte calculaSegPlato(String segPlato) -static byte calculaBebida(String bebida) -static byte calculaPostre(String postre)

TestPedido
+static void main(String[] args) +static void CapturaPedido(String menu) +static String Desplegable(String menu) +static String Desplegable1(String menu)

5)clase ArtículoCientífico

Realizar un programa en Java que permita modelar un artículo científico. Los artículos científicos contienen los siguientes metadatos: nombre del artículo, autor, palabras claves, nombre de la publicación, año y resumen.
Se deben definir tres constructores sobrecargados:

✓ El primero inicializa un artículo científico con solo su título y autor.

✓ El segundo constructor, un artículo científico con su nombre, autor, palabras claves, nombre de la publicación y año. Debe invocar al primer constructor.

✓ El tercer constructor, un artículo científico con su nombre, autor, palabras claves, nombre de la publicación, año y resumen. Debe invocar al segundo constructor.

✓ Se requiere un método que imprima los atributos de un artículo en pantalla.

Realizar una unidad ejecutable que utilice el tercer constructor para instanciar un artículo científico e imprima los valores de sus atributos en pantalla.
Recuerde que: a veces es necesario inicializar un objeto de diferentes formas. Para ello, se realiza la sobrecarga de constructores. Los constructores sobrecargados deben tener diferente número o tipo de parámetros. Cada constructor realiza una tarea diferente. Se puede utilizar la palabra reservada this para llamar a un constructor desde otro. La llamada this debe ser la primera línea de dicho constructor. Ejemplo:

```
public ArtículoCientífico(String título, String autor, String[] palabrasClaves,
String publicación, int año) {
this(título, autor); // Invoca al constructor sobrecargado
this.palabrasClaves = palabrasClaves;
this.publicación = publicación;
this.año = año;
}
```

Se crea la clase ArtículoCientífico, se identifican los atributos y su tipo de dato, en este caso son 6 atributos, 5 String y 1 short, después se crean los constructores sobrecargados, pero en este caso hacen referencia a el anterior

constructo, después se crea el toString, creamos la clase TestArtículoCientífico donde probaremos la clase previamente hecha, en este caso solo son entradas de datos así que no requieren de un menú.

ArtículoCientífico
-String nomArti -String autorArti -String palaClaves -String nomPublica -short añoArti -String resumArti
+ArtículoCientífico(String nomArti, String autorArti) +ArtículoCientífico(String nomArti, String autorArti, String palaClaves, String nomPublica, short añoArti) + ArtículoCientífico(String nomArti, String autorArti, String palaClaves, String nomPublica, short añoArti, String resumArti) +String toString()

TestArtículoCientífico
+static void main(String[] args) +static void CapturaArtículos()

6) Clase AsistenteBoda (Control de registro de los asistentes)

Implemente una aplicación que permita controlar los asistentes que son mayores de edad y que asistirán a la boda; para ello deberá de ingresar el nombre, la edad, sexo (femenino, masculino), estado civil (solter@,casad@,viud@,separad@ y divorciad@). Deberá de capturar los datos y mostrar las estadísticas usando variables estáticas (solo los contadores) e imprimir en una sola ventana emergente las frecuencias y el valor obtenido.

- ✓ Total de hombres
- ✓ Total de mujeres
- ✓ Total de casados
- ✓ Total de solteros
- ✓ Total de viudos
- ✓ Total de separados
- ✓ Total de divorciados.

✓ Imprimir los datos de la persona que tiene mayor edad y los datos de la persona que tiene menor edad.

✓ Imprimir todos los datos capturados usando una ventana emergente.

Probar en una unidad ejecutable que permita capturar los datos de tipo AsistenteBoda, y mostrar las frecuencias obtenidas , los datos de la persona que tiene mayor edad y los datos de la persona que tiene menor edad, y todos los datos capturados.

Creamos la clase AsistenteBoda, identificamos los atributos y contadores, para este caso son 4 atributos y 8 contadores, se requiere de dos constructores, uno vació y uno parametrizado, se crea cada set y get de los atributos y un get estático para el contador Asis, se genera el toString, se crean métodos de instancia para saber si es mayor, el tipo de asistente, el sexo, reutilizamos un método que creamos en la primera unidad llamado imprimeFrecuencia y otro método de clase donde se juntaran todos los contadores obtenidos para su uso, en otra clase se hace la prueba, para este caso es mejor utilizar un menú desplegable.

AsistenteBoda
-String nomAsis -byte edadAsis -char sexAsis -String estadoCivilAsis -static byte Asis -static byte HAsis -static byte MAsis -static byte CasadosAsis -static byte SolterosAsis -static byte ViudosAsis -static byte SeparadosAsis -static byte DivorciadosAsis
+AsistenteBoda() +AsistenteBoda(String nomAsis, byte edadAsis, char sexAsis, String estadoCivilAsis) +String getNomAsis() +void setNomAsis(String nomAsis) +byte getEdadAsis() +void setEdadAsis(byte edadAsis) +char getSexAsis() +void setSexAsis(char sexAsis) +String getEstadoCivilAsis() +void setEstadoCivilAsis(String estadoCivilAsis) +static byte getAsis()

+String toString() +boolean esMayor() +void tipoAsis() +void SexoAsistente() +static String imprimeFrecuencia(byte n) +static String Asistentes()

TestAsistenteBoda
+static void main(String[] args) +static void CapturaAsistentes() +static String Desplegable(String menu)

Declarar una clase Empleado con los atributos nombre, edad, horas trabajadas, pago por hora. Definir los siguientes métodos estáticos:

• Recibir como parámetros dos objetos de la clase Empleado y que retorne el que tiene una edad mayor, si son iguales retorne cualquiera de las dos.

• Recibir como parámetros dos objetos de la clase Empleado y que retorne el que trabajo mas horas si son iguales retorne cualquiera de las dos.

Se crea la clase Empleado, identificamos 5 atributos privados, se crea un constructor parametrizado con todos los atributos, se crean sus set y get, después creamos un método donde recibe como parámetro dos objetos de la clase Empleado y de este saca el mayor de los dos, el siguiente método recibe dos objetos de la clase empleado y de estos retorna el que tiene el mayor número de horas trabajadas, se le creo un test para probar la clase.

Posteriormente agregamos dos contadores que inicializamos, uno en 0 y el otro en 1000, introducimos estos contadores en el constructor para que contara los empleados y otro que les diera un numero de control a cada objeto y se le creo un get a conta.

Empleado
-int cveEmpleado -String nomEmp -byte edadEmp -byte hrstrabEmp

-double pagohrsEmp -static byte conta -static short inicial
+Empleado(String nombEmp, byte edadEmp, byte hrstrabEmp, double pagohrsEmp) +String getNombEmp() +void setNombEmp(String nomEmp) +byte getEdadEmp() +void setEdadEmp(byte edadEmp) +byte getHrstrabEmp() +void setHrstrabEmp(byte hrstrabEmp) +double getPagohrsEmp() +void setPagohrsEmp(double pagohrsEmp) +static byte getConta() +String toString() +static Empleado mayorEdad(Empleado emp1, Empleado emp2) +static Empleado mayorHrsTrabajadas(Empleado emp1, Empledo emp2)

TestEmpleado
+static void main(String[] args) +static void CapturaEmpleados()

Pinturas:

Esta clase es reutilizada de el examen de la primera unidad, en este no hay variables de instancia ni constructores, en este ocupamos dos métodos, el primero recibe como parámetros 1 String y 2 de tipo double, dentro de este se hacen las operaciones para calcular los litros de pintura que se requieren y el precio, el segundo método es la unidad ejecutable donde se piden los datos para llamar al método.

Pinturas

+static String cotizacionPintura(String nombreCliente, double largo, double alto) +static void main(String[] args)

Resultados:

Tools

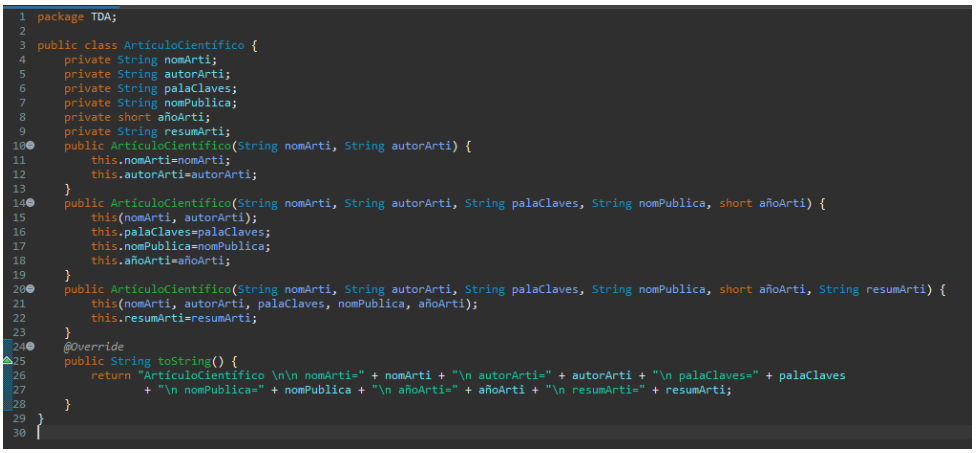
```java
package EntradaSalida;

import javax.swing.JOptionPane;

public 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 double leerDouble(String msje) {
        return (Double.parseDouble(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 código define una clase llamada "Tools" que contiene varios métodos estáticos que se pueden usar para interactuar con el usuario mediante ventanas emergentes (pop-up dialogs). Cada método tiene una tarea específica:

- "imprimeSalida" muestra un mensaje en una ventana emergente con un botón de pregunta (QUESTION_MESSAGE).
- "salidaError" muestra un mensaje de error en una ventana emergente con un icono de error (ERROR_MESSAGE).
- "leerShort", "leerByte", "leerInt", "leerLong", "leerFloat","leerDouble" y "leerChar" muestran un mensaje en una ventana emergente con un campo de entrada y un botón de confirmación. Estos métodos convierten la entrada del usuario en un tipo de datos específico (short, byte, int, long, float, double o char) y lo devuelven como resultado.
- "leerString" es similar a los métodos anteriores, pero devuelve una cadena de caracteres como resultado.

- "seguirSino" muestra un mensaje en una ventana emergente con dos botones (Sí y No) y devuelve el botón que el usuario seleccionó como resultado.
- "leerEntero" es similar a "leerInt", pero muestra un mensaje diferente en la ventana emergente.
- "imprimePantalla" muestra un mensaje en una ventana emergente con un icono de información (INFORMATION_MESSAGE).

ArtículoCientífico:



```java
package TDA;

public class ArtículoCientífico {
    private String nomArti;
    private String autorArti;
    private String palaClaves;
    private String nomPublica;
    private short añoArti;
    private String resumArti;
    public ArtículoCientífico(String nomArti, String autorArti) {
        this.nomArti=nomArti;
        this.autorArti=autorArti;
    }
    public ArtículoCientífico(String nomArti, String autorArti, String palaClaves, String nomPublica, short añoArti) {
        this(nomArti, autorArti);
        this.palaClaves=palaClaves;
        this.nomPublica=nomPublica;
        this.añoArti=añoArti;
    }
    public ArtículoCientífico(String nomArti, String autorArti, String palaClaves, String nomPublica, short añoArti, String resumArti) {
        this(nomArti, autorArti, palaClaves, nomPublica, añoArti);
        this.resumArti=resumArti;
    }
    @Override
    public String toString() {
        return "ArtículoCientífico \n\n nomArti=" + nomArti + "\n autorArti=" + autorArti + "\n palaClaves=" + palaClaves
                + "\n nomPublica=" + nomPublica + "\n añoArti=" + añoArti + "\n resumArti=" + resumArti;
    }
}
```

Esta es una clase llamada "ArtículoCientífico" que tiene varios atributos (nomArti, autorArti, palaClaves, nomPublica, añoArti, resumArti) que representan diferentes aspectos de un artículo científico, como su nombre, autor, palabras clave, nombre de la publicación, año de publicación y resumen. La clase tiene tres constructores, que permiten crear un objeto "ArtículoCientífico" con diferentes combinaciones de atributos. La clase también tiene un método "toString()" que devuelve una cadena que representa el objeto en forma legible de los datos capturados.

AsistenteBoda:

```java
package TDA;

public class AsistenteBoda {
    private String nomAsis;
    private byte edadAsis;
    private char sexAsis;
    private String estadoCivilAsis;
    private static byte Asis;
    private static byte HAsis;
    private static byte MAsis;
    private static byte CasadosAsis;
    private static byte SolterosAsis;
    private static byte ViudosAsis;
    private static byte SeparadosAsis;
    private static byte DivorciadosAsis;
    public AsistenteBoda() {}
    public AsistenteBoda(String nomAsis, byte edadAsis, char sexAsis, String estadoCivilAsis) {
        this.nomAsis=nomAsis;
        this.edadAsis=edadAsis;
        this.sexAsis=sexAsis;
        this.estadoCivilAsis=estadoCivilAsis;
        Asis++;
    }
    public String getNomAsis() {
        return nomAsis;
    }
    public void setNomAsis(String nomAsis) {
        this.nomAsis = nomAsis;
    }
    public byte getEdadAsis() {
        return edadAsis;
    }
    public void setEdadAsis(byte edadAsis) {
        this.edadAsis = edadAsis;
    }
    public char getSexAsis() {
        return sexAsis;
    }
    public void setSexAsis(char sexAsis) {
        this.sexAsis = sexAsis;
    }
    public String getEstadoCivilAsis() {
        return estadoCivilAsis;
    }
    public void setEstadoCivilAsis(String estadoCivilAsis) {
        this.estadoCivilAsis = estadoCivilAsis;
    }
```

```java
    public void setEstadoCivilAsis(String estadoCivilAsis) {
        this.estadoCivilAsis = estadoCivilAsis;
    }
    public static byte getAsis() {
        return Asis;
    }
    @Override
    public String toString() {
        return "Nombre= " + nomAsis + ", Edad= " + edadAsis + ", Sexo= " + sexAsis
                + ", Estado civil= " + estadoCivilAsis;
    }
    public boolean esMayor() {
        return edadAsis>=18;
    }
    public void tipoAsis() {
        switch (estadoCivilAsis) {
        case "Casad@": CasadosAsis++;break;
        case "Solter@": SolterosAsis++;break;
        case "Viud@": ViudosAsis++;break;
        case "Separad@": SeparadosAsis++;break;
        case "Divorciad@": DivorciadosAsis++;break;
        }
    }
    public void SexoAsistente() {
        if (sexAsis=='H')
            HAsis++;
        else
            MAsis++;
    }
    public static String imprimeFrecuencia(byte n) {
        String cad=" ";
        for(int i=1;i<=n;i++) {
            cad+="*";
        }
        return cad;
    }
    public static String Asistentes() {
        return "Total de hombres: "+imprimeFrecuencia(HAsis)+
                "\n Total de mujeres: "+imprimeFrecuencia(MAsis)+
                "\n Total de casados: "+CasadosAsis+
                "\n Total de solteros: "+SolterosAsis+
                "\n Total de viudos: "+ViudosAsis+
                "\n Total de separados: "+SeparadosAsis+
                "\n Total de divorciados: "+DivorciadosAsis;
    }
}
```

La clase "AsistenteBoda" representa a una persona que asiste a una boda y contiene:

Atributos: nomAsis, edadAsis, sexAsis y estadoCivilAsis

Atributos estaticos o contadores: Asis, HAsis, MAsis, CasadosAsis, SolterosAsis, ViudosAsis, SeparadosAsis y DivorciadosAsis.

“AsistenteBoda” no recibe parámetros.

“AsistenteBoda” recibe los parámetros nomAsis (nombre del asistente), edadAsis (edad del asistente), sexAsis (sexo del asistente) y estadoCivilAsis

(estado civil del asistente) y los asigna a las variables correspondientes de la instancia creada. Además, incrementa el valor de la variable estática Asis.

“getNomAsis” devuelve el valor de la variable nomAsis.

“setNomAsis” recibe un parámetro nomAsis y lo asigna a la variable nomAsis.

“getEdadAsis” devuelve el valor de la variable edadAsis.

“setEdadAsis” recibe un parámetro edadAsis y lo asigna a la variable edadAsis.

“getSexAsis” devuelve el valor de la variable sexAsis.

“setSexAsis” recibe un parámetro sexAsis y lo asigna a la variable sexAsis.

“getEstadoCivilAsis” devuelve el valor de la variable estadoCivilAsis.

“setEstadoCivilAsis” recibe un parámetro estadoCivilAsis y lo asigna a la variable estadoCivilAsis.

“getAsis” devuelve el valor de la variable estática Asis.

“toString” devuelve una cadena de caracteres que representa la información del asistente en formato de texto.

“esMayor” devuelve true si la edad del asistente es mayor o igual a 18, y false en caso contrario.

“tipoAsis” actualiza las variables estáticas CasadosAsis, SolterosAsis, ViudosAsis, SeparadosAsis y DivorciadosAsis según el estado civil del asistente.

“SexoAsistente” actualiza las variables estáticas HAsis y MAsis según el sexo del asistente.

“imprimeFrecuencia” recibe un parámetro n y devuelve una cadena de caracteres que representa la frecuencia de * según el valor de n.

“Asistentes” devuelve una cadena de caracteres que representa la información de los asistentes en formato de texto, incluyendo la cantidad de hombres, mujeres, casados, solteros, viudos, separados y divorciados.

ConversorMetros:

```java
package TDA;

public class ConversorMetros {
    private double metros;
    final byte METROS_CM=100;
    final short METROS_MILIM=1000;
    final double METROS_PULGADAS=39.37;
    final double METROS_PIES=3.28;
    final double METROS_YARDAS=1.09361;
    public ConversorMetros(double metros) {
        this.metros = metros;
    }
    private double Metros_A_Centimetros() {
        return metros*METROS_CM;
    }
    private double Metros_A_Milimetros() {
        return metros*METROS_MILIM;
    }
    private double Metros_A_Pulgadas() {
        return metros*METROS_PULGADAS;
    }
    private double Metros_A_Pies() {
        return metros*METROS_PIES;
    }
    private double Metros_A_Yardas() {
        return metros*METROS_YARDAS;
    }
    public double Metros_A_Centimetros2() {
        return Metros_A_Centimetros();
    }
    public double Metros_A_Milimetros2() {
        return Metros_A_Milimetros();
    }
    public double Metros_A_Pulgadas2() {
        return Metros_A_Pulgadas();
    }
    public double Metros_A_Pies2() {
        return Metros_A_Pies();
    }
    public double Metros_A_Yardas2() {
        return Metros_A_Yardas();
    }
}
```

La clase llamada "ConversorMetros" es una clase que tiene la funcionalidad de convertir una cantidad de metros a diferentes unidades de medida como centímetros, milímetros, pulgadas, pies y yardas.

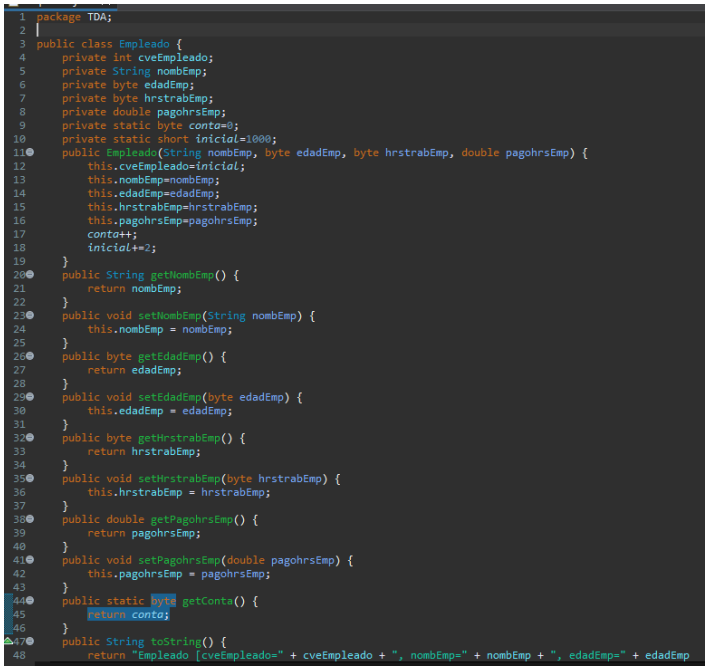
La clase tiene una variable privada de tipo "double" llamada "metros" que representa la cantidad de metros que se desea convertir. También hay cinco constantes finales que representan la relación de conversión entre metros y las otras unidades de medida.

El constructor de la clase toma como parámetro una cantidad de metros y asigna este valor a la variable "metros".

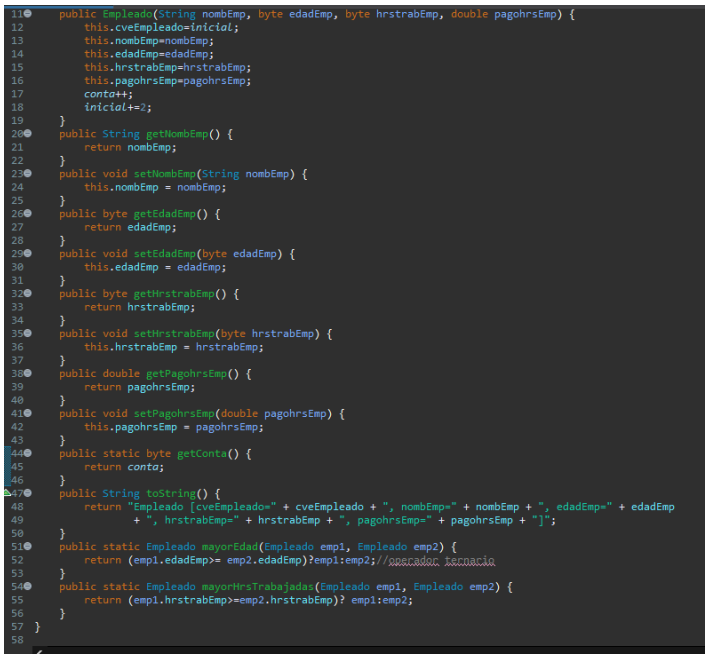
La clase también tiene cinco métodos privados que realizan las conversiones a cada una de las unidades de medida mencionadas anteriormente. Cada método toma la cantidad de metros almacenada en la variable "metros" y la multiplica por la relación de conversión correspondiente.

Además, la clase tiene cinco métodos públicos que devuelven el resultado de cada conversión en una nueva variable de tipo "double". Cada uno de estos métodos simplemente llama al método privado correspondiente para realizar la conversión y devuelve su resultado.

Empleado:



```java
package TDA;

public class Empleado {
    private int cveEmpleado;
    private String nombEmp;
    private byte edadEmp;
    private byte hrstrabEmp;
    private double pagohrsEmp;
    private static byte conta=0;
    private static short inicial=1000;
    public Empleado(String nombEmp, byte edadEmp, byte hrstrabEmp, double pagohrsEmp) {
        this.cveEmpleado=inicial;
        this.nombEmp=nombEmp;
        this.edadEmp=edadEmp;
        this.hrstrabEmp=hrstrabEmp;
        this.pagohrsEmp=pagohrsEmp;
        conta++;
        inicial+=2;
    }
    public String getNombEmp() {
        return nombEmp;
    }
    public void setNombEmp(String nombEmp) {
        this.nombEmp = nombEmp;
    }
    public byte getEdadEmp() {
        return edadEmp;
    }
    public void setEdadEmp(byte edadEmp) {
        this.edadEmp = edadEmp;
    }
    public byte getHrstrabEmp() {
        return hrstrabEmp;
    }
    public void setHrstrabEmp(byte hrstrabEmp) {
        this.hrstrabEmp = hrstrabEmp;
    }
    public double getPagohrsEmp() {
        return pagohrsEmp;
    }
    public void setPagohrsEmp(double pagohrsEmp) {
        this.pagohrsEmp = pagohrsEmp;
    }
    public static byte getConta() {
        return conta;
    }
    public String toString() {
        return "Empleado [cveEmpleado=" + cveEmpleado + ", nombEmp=" + nombEmp + ", edadEmp=" + edadEmp
```



```java
    public Empleado(String nombEmp, byte edadEmp, byte hrstrabEmp, double pagohrsEmp) {
        this.cveEmpleado=inicial;
        this.nombEmp=nombEmp;
        this.edadEmp=edadEmp;
        this.hrstrabEmp=hrstrabEmp;
        this.pagohrsEmp=pagohrsEmp;
        conta++;
        inicial+=2;
    }
    public String getNombEmp() {
        return nombEmp;
    }
    public void setNombEmp(String nombEmp) {
        this.nombEmp = nombEmp;
    }
    public byte getEdadEmp() {
        return edadEmp;
    }
    public void setEdadEmp(byte edadEmp) {
        this.edadEmp = edadEmp;
    }
    public byte getHrstrabEmp() {
        return hrstrabEmp;
    }
    public void setHrstrabEmp(byte hrstrabEmp) {
        this.hrstrabEmp = hrstrabEmp;
    }
    public double getPagohrsEmp() {
        return pagohrsEmp;
    }
    public void setPagohrsEmp(double pagohrsEmp) {
        this.pagohrsEmp = pagohrsEmp;
    }
    public static byte getConta() {
        return conta;
    }
    public String toString() {
        return "Empleado [cveEmpleado=" + cveEmpleado + ", nombEmp=" + nombEmp + ", edadEmp=" + edadEmp
                + ", hrstrabEmp=" + hrstrabEmp + ", pagohrsEmp=" + pagohrsEmp + "]";
    }
    public static Empleado mayorEdad(Empleado emp1, Empleado emp2) {
        return (emp1.edadEmp>= emp2.edadEmp)?emp1:emp2;//operador ternario
    }
    public static Empleado mayorHrsTrabajadas(Empleado emp1, Empleado emp2) {
        return (emp1.hrstrabEmp>=emp2.hrstrabEmp)? emp1:emp2;
    }
}
```

La clase "Empleado" es una clase que representa a un empleado en una empresa. Tiene varios atributos como el número de empleado, el nombre, la edad, las horas trabajadas y el pago por hora. También tiene algunos métodos para obtener y establecer los valores de estos atributos y algunos métodos estáticos para comparar los atributos de dos empleados.

El atributo "cveEmpleado" es un identificador único para cada empleado y se inicializa automáticamente con un valor incremental al construir un objeto de la clase. Los demás atributos representan información personal y laboral del empleado.

El constructor de la clase toma como parámetros el nombre del empleado, su edad, las horas trabajadas y el pago por hora. El número de empleado se inicializa automáticamente con un valor incremental y se asigna a la variable "cveEmpleado". El contador de empleados "conta" se incrementa para llevar la cuenta de la cantidad de empleados creados. La variable "inicial" se incrementa en 2 para asegurar que el siguiente número de empleado sea único.

La clase tiene métodos de acceso para obtener y establecer los valores de los atributos, como "getNombEmp", "setNombEmp", "getEdadEmp", "setEdadEmp", "getHrstrabEmp", "setHrstrabEmp", "getPagohrsEmp" y "setPagohrsEmp". Estos métodos permiten obtener y establecer los valores de los atributos de un objeto de la clase.

La clase también tiene un método estático llamado "mayorEdad" que toma dos objetos de la clase "Empleado" como parámetros y devuelve el objeto que tiene una edad mayor. El método utiliza un operador ternario para comparar las edades de los dos empleados.

Además, la clase tiene otro método estático llamado "mayorHrsTrabajadas" que toma dos objetos de la clase "Empleado" como parámetros y devuelve el objeto que tiene más horas trabajadas. Este método también utiliza un operador ternario para comparar las horas trabajadas de los dos empleados.

Por último, la clase tiene un método "toString" que devuelve una cadena de texto que representa al objeto de la clase en forma de texto. Este método se utiliza para imprimir información del objeto en la consola de Java.

Pedido:

```java
package TDA;

import EntradaSalida.Tools;

public class Pedido {
    public static void CalcularPedido(String primPlato, String bebida) {
        short total=(short)(calculaPrimPlato(primPlato)+calcularBebida(bebida));
        Tools.imprimePantalla("Total a pagar: "+total);
    }
    public static void CalcularPedido(String primPlato, String segPlato,String bebida) {
        short total=(short)(calculaPrimPlato(primPlato)+calculaSegPlato(segPlato)+calcularBebida(bebida));
        Tools.imprimePantalla("Total a pagar: "+total);
    }
    public static void CalcularPedido(String primPlato, String segPlato,String bebida, String postre) {
        short total=(short)(calculaPrimPlato(primPlato)+calculaSegPlato(segPlato)+calcularBebida(bebida)+calcularPostre(postre));
        Tools.imprimePantalla("Total a pagar: "+total);
    }
    private static byte calculaPrimPlato(String primPlato) {
        byte pr1=0;
        switch (primPlato) {
        case "Crema de espinacas": pr1=20;break;
        case "Ensalada de verduras": pr1=22;break;
        case "Crema de brocoli": pr1=25;break;
        case "Caldo tlalpeño": pr1=30;break;
        case "Sopa mixteca": pr1=27;break;
        }
        return pr1;
    }
    private static byte calculaSegPlato(String segPlato) {
        byte pr2=0;
        switch (segPlato) {
        case "Filete de pescado": pr2=60;break;
        case "Milanesa de pollo": pr2=45;break;
        case "Bistec a la mexicana": pr2=55;break;
        case "Pollo en escabeche": pr2=35;break;
        case "Carne asada": pr2=40;break;
        case "Lomo relleno": pr2=56;break;
        case "Pollo a la plancha": pr2=49;break;
        }
        return pr2;
    }
    private static byte calcularBebida(String bebida) {
        byte prb=0;
        switch (bebida) {
        case "Coca cola": prb=20;break;
        case "Pepsi": prb=18;break;
        case "Naranjada": prb=15;break;
        case "Limonada": prb=15;break;
```

```java
        byte pr1=0;
        switch (primPlato) {
        case "Crema de espinacas": pr1=20;break;
        case "Ensalada de verduras": pr1=22;break;
        case "Crema de brocoli": pr1=25;break;
        case "Caldo tlalpeño": pr1=30;break;
        case "Sopa mixteca": pr1=27;break;
        }
        return pr1;
    }
    private static byte calculaSegPlato(String segPlato) {
        byte pr2=0;
        switch (segPlato) {
        case "Filete de pescado": pr2=60;break;
        case "Milanesa de pollo": pr2=45;break;
        case "Bistec a la mexicana": pr2=55;break;
        case "Pollo en escabeche": pr2=35;break;
        case "Carne asada": pr2=40;break;
        case "Lomo relleno": pr2=56;break;
        case "Pollo a la plancha": pr2=49;break;
        }
        return pr2;
    }
    private static byte calcularBebida(String bebida) {
        byte prb=0;
        switch (bebida) {
        case "Coca cola": prb=20;break;
        case "Pepsi": prb=18;break;
        case "Naranjada": prb=15;break;
        case "Limonada": prb=15;break;
        case "Agua de sabor":prb=16;break;
        }
        return prb;
    }
    private static byte calcularPostre(String postre) {
        byte prp=0;
        switch (postre) {
        case "Pastel helado": prp=70;break;
        case "Helado": prp=25;break;
        case "Fresas con crema": prp=40;break;
        case "Platanos fritos": prp=30;break;
        case "Flan casero": prp=35;break;
        case "Gelatina": prp=45;break;
        }
        return prp;
    }
}
```

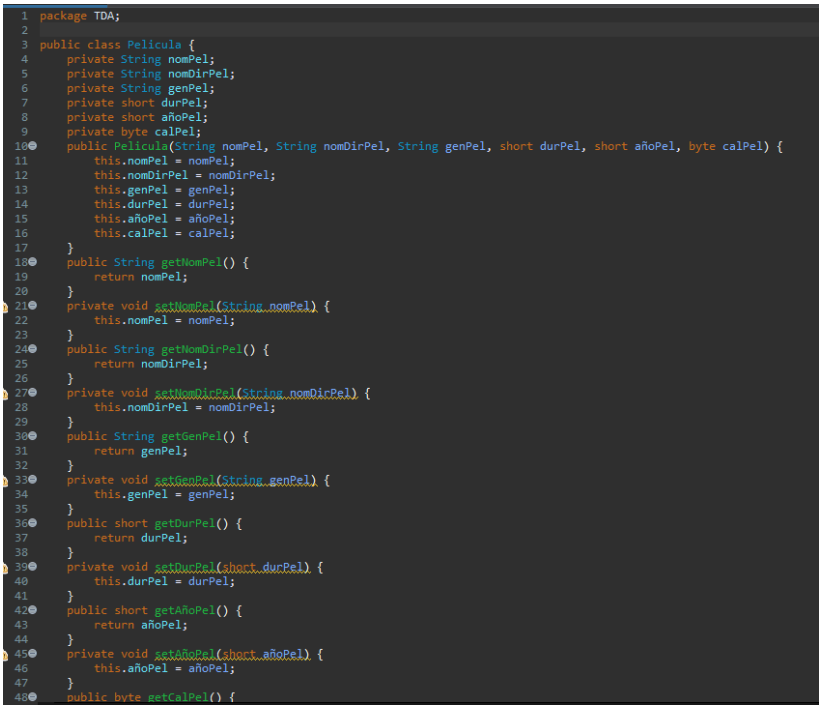
La clase llamada "Pedido" tiene varios métodos que permiten calcular el costo total de un pedido según los platillos y bebidas seleccionados. Los métodos son "CalcularPedido" y hay tres versiones diferentes, cada una con diferentes parámetros. El primer método "CalcularPedido" recibe dos parámetros de tipo String, que representan un plato principal y una bebida. El segundo método "CalcularPedido" recibe tres parámetros de tipo String, que representan un plato principal, un segundo plato y una bebida. El tercer método "CalcularPedido" recibe cuatro parámetros de tipo String, que representan un plato principal, un segundo plato, una bebida y un postre.

Cada uno de estos métodos utiliza otros métodos privados de la clase para calcular el costo de cada uno de los elementos del pedido. Por ejemplo, el método privado "calculaPrimPlato" recibe como parámetro un String que

representa el nombre del plato principal y devuelve un valor numérico que representa su costo. El método "calcularBebida" y "calcularPostre" funcionan de manera similar, recibiendo como parámetro un String que representa el nombre de la bebida o postre, respectivamente, y devolviendo su costo en forma de un valor numérico.

Una vez que se ha calculado el costo total del pedido utilizando los métodos privados correspondientes, los métodos "CalcularPedido" imprimen el costo total en la pantalla utilizando un método llamado "imprimePantalla", que es proporcionado por otra clase llamada "Tools".

Pelicula:



```java
package TDA;

public class Pelicula {
    private String nomPel;
    private String nomDirPel;
    private String genPel;
    private short durPel;
    private short añoPel;
    private byte calPel;
    public Pelicula(String nomPel, String nomDirPel, String genPel, short durPel, short añoPel, byte calPel) {
        this.nomPel = nomPel;
        this.nomDirPel = nomDirPel;
        this.genPel = genPel;
        this.durPel = durPel;
        this.añoPel = añoPel;
        this.calPel = calPel;
    }
    public String getNomPel() {
        return nomPel;
    }
    private void setNomPel(String nomPel) {
        this.nomPel = nomPel;
    }
    public String getNomDirPel() {
        return nomDirPel;
    }
    private void setNomDirPel(String nomDirPel) {
        this.nomDirPel = nomDirPel;
    }
    public String getGenPel() {
        return genPel;
    }
    private void setGenPel(String genPel) {
        this.genPel = genPel;
    }
    public short getDurPel() {
        return durPel;
    }
    private void setDurPel(short durPel) {
        this.durPel = durPel;
    }
    public short getAñoPel() {
        return añoPel;
    }
    private void setAñoPel(short añoPel) {
        this.añoPel = añoPel;
    }
    public byte getCalPel() {
```

La clase Pelicula representa una película y tiene los siguientes atributos:

Atributos: nomPel, nomDirPel, genPel, durPel, añoPel y calPel.

La clase tiene un constructor que recibe todos los atributos mencionados anteriormente y un conjunto de métodos get y set para obtener y modificar los valores de los atributos.

Además, la clase tiene varios métodos adicionales, que incluyen:

“esPeliculaEpica”: un método privado que devuelve true si la duración de la película es de 180 minutos o más.

“CalcularValoracion”: un método privado que devuelve una cadena de caracteres que representa la valoración de la película, según su calificación. La valoración puede ser "Muy mala", "Mala", "Regular", "Buena", "Excelente" o "No definida".

“esSimilar”: un método privado que recibe otra película como parámetro y devuelve true si ambas películas tienen el mismo género y la misma calificación.

“imprimirCartel”: un método privado que devuelve una cadena de caracteres que representa el cartel de la película, incluyendo su nombre, frecuencia de calificación, género, año de lanzamiento y nombre del director.

La clase también tiene métodos adicionales que son versiones públicas de los métodos privados antes mencionados. Los métodos públicos son:

“esPeliculaEpica2”: un método público que devuelve una cadena de caracteres que indica si la película es épica o no.

“CalcularValoracion2”: un método público que devuelve una cadena de caracteres que representa la valoración de la película.

“esSimilar2”: un método público que recibe otra película como parámetro y devuelve una cadena de caracteres que indica si ambas películas son similares o no.

“imprimirCartel2”: un método público que devuelve una cadena de caracteres que representa el cartel de la película.

Pinturas:

```java
package TDA;

import EntradaSalida.Tools;

public class Pinturas {
    public static String cotizacionPintura(String nombreCliente, double largo, double alto) {
        double area = largo * alto;
        double rendimiento = 3.0;
        double litros;
        String cad="";
        if (area % rendimiento == 0) {
            litros = area / rendimiento;
        } else {
            litros = Math.ceil(area / rendimiento);
        }
        double costo = litros * 75;
        cad="Cliente: "+nombreCliente+"\n"+"Metros Cuadrados a pintar: "+area+"\n"+"Litros de pintura que se requieren: "+litros+"\n"+"Costo Total a pagar: "+costo;
        return cad;
    }
    public static void main(String []args) {
        Tools.imprimePantalla(cotizacionPintura(Tools.leerString("Nombre del Cliente: "),
                Tools.leerDouble("Dame el largo: "), Tools.leerDouble("Dame el alto: ")));
    }
}

```

La clase "Pinturas" tiene un método estático llamado "cotizacionPintura" que calcula la cantidad de pintura y el costo total para pintar una superficie de tamaño especificado. Los parámetros son el nombre del cliente, el largo y el alto de la

superficie a pintar. El cálculo de la cantidad de pintura se basa en un rendimiento de 3 metros cuadrados por litro y se redondea hacia arriba si la división no es exacta. El costo total se calcula multiplicando la cantidad de litros necesarios por el precio unitario de $75.

El método "main" utiliza la función "Tools.leerString", "Tools.leerDouble" y "Tools.imprimePantalla" para solicitar información del usuario y mostrar el resultado de la cotización en la pantalla.

Rectangulo:

```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);
    }
    public static String AreaCubierta(Rectangulo salon, Rectangulo alf1, Rectangulo alf2) {
        String a="";
        if ((alf1.AreaRectangulo()+alf2.AreaRectangulo())-salon.AreaRectangulo()==salon.AreaRectangulo())
            a="Toda el area esta cubierta";
        else
            a="El area cubierta es= "+(alf1.AreaRectangulo()+alf2.AreaRectangulo())+
                "\nEl area descubierta es= "+(salon.AreaRectangulo()-(alf1.AreaRectangulo()+alf2.AreaRectangulo()));
        return a;
    }
}
```

La clase "Rectangulo" es una clase que representa un rectángulo, la cual contiene los atributos "base" y "altura", así como los métodos "get" y "set" para acceder y modificar estos atributos. También tiene un constructor que toma los valores iniciales de la base y la altura y los asigna a los atributos correspondientes. Además, tiene métodos para calcular el área y el perímetro del rectángulo.

La clase también tiene un método estático "AreaCubierta" que toma tres objetos de la clase Rectangulo como argumentos: "salon", "alf1" y "alf2". Este método calcula y devuelve una cadena que indica si todo el área del salón está cubierta por los dos rectángulos "alf1" y "alf2" o si hay alguna área descubierta. Si toda el

área está cubierta, la cadena devuelta indica que "Toda el area esta cubierta". De lo contrario, la cadena devuelta incluye información sobre el área cubierta y el área descubierta.

TestArtículoCientífico:

```java
package Test;

import EntradaSalida.Tools;

public class TestArtículoCientífico {
    public static void main(String[] args) {
        CapturaArtículos();
    }
    public static void CapturaArtículos() {
        ArtículoCientífico arti=new ArtículoCientífico( Tools.leerString("Nombre del Artículo Científico: "),
                                                        Tools.leerString("Autor del Artículo Científico:"),
                                                        Tools.leerString("Palabras Clave del Artículo Científico: "),
                                                        Tools.leerString("Nombre de la publicación del Artículo Científico: "),
                                                        Tools.leerShort("Año de publicación del Artículo Científico: "),
                                                        Tools.leerString("Resumen del Artículo Científico: "));
        Tools.imprimePantalla("Datos del Artículo Científico capturados:\n\n"+arti.toString());
    }
}
```

La clase “TestArtículoCientífico” contiene un método main que llama al método CapturaArtículos. Este último método captura los datos de un artículo científico utilizando el constructor de la clase ArtículoCientífico y los métodos leerString y leerShort de la clase Tools. Después, se imprime en la pantalla los datos del artículo capturados utilizando el método toString de la clase ArtículoCientífico. En resumen, esta clase es un ejemplo de cómo utilizar la clase ArtículoCientífico para crear y mostrar un objeto de esa clase.

TestAsistenteBodega:

```java
package Test;

import javax.swing.JOptionPane;

public class TestAsistenteBoda {
    public static void main(String[] args) {
        CapturaAsistentes();
    }
    public static void CapturaAsistentes() {
        byte res,x;
        String cad="", cadMay="", cadMen="";
        do {
            AsistenteBoda As=new AsistenteBoda();
            As.setNomAsis(Tools.leerString("Nombre: "));
            As.setEdadAsis(Tools.leerByte("Edad: "));
            do {
                As.setSexAsis(Tools.leerChar("Sexo: [H]ombre  [M]ujer"));
                x=(As.getSexAsis()=='H'||As.getSexAsis()=='M')?(byte)1:(byte)0;
            }while(x!=1);
            As.setEstadoCivilAsis(Desplegable("Solter@,Casad@,Viud@,Separad@,Divorciad@"));
            if (As.esMayor()) cadMay+=As.toString()+"\n";
            else cadMen+=As.toString()+"\n";
            As.tipoAsis();
            As.SexoAsistente();
            cad+=As.toString()+"\n";

            res=(byte)Tools.seguirSino();
        }while(res!=1);
        Tools.imprimePantalla("Asistentes Capturados: \n\n"+AsistenteBoda.Asistentes()+
                "\n\n Asistentes mayores: \n\n"+cadMay+
                "\n\n Asistentes menores: \n\n"+cadMen+
                "\n\n Todos los asistentes \n\n"+cad);
    }
    public static String Desplegable(String menu) {
        String valores[]=menu.split(",");
        String res=(String)JOptionPane.showInputDialog(null,"Estado Civil: ","M E N U",
                JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
        return res;
    }
}
```

La clase “TestAsistenteBoda” contiene un método main que llama al método CapturaAsistentes. Dentro del método CapturaAsistentes, se utiliza un bucle do-while para capturar los datos de los asistentes a una boda.

En cada iteración del bucle, se crea un objeto AsistenteBoda y se establecen sus propiedades (nombre, edad, sexo, estado civil) utilizando métodos set y métodos de entrada de datos (leerString, leerByte, leerChar, Desplegable). Luego, se llama a varios métodos de la clase AsistenteBoda para determinar el tipo y el sexo del asistente y para generar una representación en formato de cadena del objeto.

La cadena resultante se agrega a diferentes variables de cadena (cad, cadMay, cadMen) según corresponda, según la edad del asistente. El bucle continúa hasta que el usuario indica que no desea ingresar más asistentes.

Finalmente, se utiliza el método imprimePantalla de la clase Tools para imprimir una lista de todos los asistentes capturados, una lista de los asistentes mayores y otra lista de los asistentes menores. Todo esto se hace dentro del método CapturaAsistentes.

TestConversorMetros:

```
package Test;

import javax.swing.JOptionPane;

public class TestConversorMetros {
    public static void main(String[] args) {
        CapturaMetros("Convertir Metros a...,Salir");
    }
    public static void CapturaMetros(String menu) {
        double resul=0;
        String sel="", con="";
        do {
            sel=Boton(menu);
            switch (sel) {
            case "Convertir Metros a...":
                ConversorMetros metros=new ConversorMetros(Tools.leerDouble("Dame los metros: "));
                con=Desplegable("Centimetros,Milimetros,Pulgadas,Pies,Yardas");
                switch (con) {
                case "Centimetros": resul=metros.Metros_A_Centimetros2();break;
                case "Milimetros": resul=metros.Metros_A_Milimetros2();break;
                case "Pulgadas": resul=metros.Metros_A_Pulgadas2();break;
                case "Pies": resul=metros.Metros_A_Pies2();break;
                case "Yardas": resul=metros.Metros_A_Yardas2();break;
                }
                Tools.imprimePantalla("El resultado es= "+resul);
                break;
            case "Salir":
                break;
            }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
    public static String Desplegable(String menu) {
        String valores[]=menu.split(",");
        String res=(String)JOptionPane.showInputDialog(null,"¿A que unidad de medida quieres convertir?","M E N U",
                JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
        return res;
    }
    public static String Boton(String menu) {
        String valores[]=menu.split(",");
        int n;
        n=JOptionPane.showOptionDialog(null,"SELECCIONA DANDO CLICK","M E N U",JOptionPane.NO_OPTION,
                JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
    return(valores[n]);
    }
}

```

La clase TestConversorMetros implementa un programa que permite convertir medidas de metros a diferentes unidades de medida.

El método main llama al método CapturaMetros pasando como parámetro una cadena que representa las opciones de menú que se mostrarán en el programa.

El método CapturaMetros contiene un bucle que se ejecutará mientras la selección del usuario no sea "Salir". Dentro del bucle se muestra un menú de opciones utilizando el método Boton. Si la opción seleccionada por el usuario es "Convertir Metros a...", se pide al usuario que ingrese la cantidad de metros que desea convertir y se muestra otro menú de opciones utilizando el método Desplegable, donde se le pide al usuario que seleccione a qué unidad de medida desea convertir los metros ingresados. Luego, se utiliza un switch para determinar qué método de conversión debe llamarse en la clase ConversorMetros y se muestra el resultado en pantalla utilizando el método imprimePantalla de la clase Tools.

El método Desplegable recibe una cadena con las opciones de menú separadas por comas y devuelve la opción seleccionada por el usuario utilizando un cuadro de diálogo JOptionPane.

El método Boton recibe una cadena con las opciones de menú separadas por comas, muestra un cuadro de diálogo con botones correspondientes a las opciones y devuelve la opción seleccionada por el usuario.

TestEmpleado:

```java
package Test;

import EntradaSalida.Tools;

public class TestEmpleado {
    public static void main(String[] args) {
        CapturaEmpleados();
    }
    public static void CapturaEmpleados() {
        Empleado carlos = new Empleado("Carlos Perez",(byte)25, (byte)40, 180.5);
        Empleado sonia = new Empleado("Sonia Alvarez",(byte)19, (byte)45, 180.5);
        Empleado alma = new Empleado("Alma Alvarez",(byte)19, (byte)45, 180.5);
        String cad="Empleado con mayor edad: "+Empleado.mayorEdad(carlos, sonia)+"\n";
        cad+="Empleado que trabajo mas horas: "+Empleado.mayorHrsTrabajadas(carlos, sonia);
        Tools.imprimePantalla(cad+"\n"+carlos.toString()+"\n"+sonia.toString()+"\n"+alma.toString()+
                "\n\n Total de objetos creados: \n"+Empleado.getConta());
    }
}
```

La clase TestEmpleado crea tres objetos Empleado con distintos atributos y luego llama a dos métodos estáticos de la clase Empleado para obtener información sobre estos objetos. El método mayorEdad devuelve el objeto

Empleado con la mayor edad y el método mayorHrsTrabajadas devuelve el objeto Empleado que ha trabajado más horas. La información resultante se concatena en una cadena de texto que luego se muestra en la consola junto con la representación de cadena de cada objeto Empleado creado. También se imprime el total de objetos Empleado creados.

TestPedido:

```java
package Test;

import javax.swing.JOptionPane;

public class TestPedido {
    public static void main(String[] args) {
        CapturaPedido("1. Un primer plato y una bebida,2. Un primer plato + un segundo plato y una bebida,"
                + "3. Un primer plato + un segundo plato + una bebida y un postre,Salir");
    }
    public static void CapturaPedido(String menu) {
        String sel="";
        do {
            sel=Desplegable1(menu);
            switch (sel) {
            case "1. Un primer plato y una bebida":
                Pedido.CalcularPedido(Desplegable("Crema de espincas,Ensalada de verduras,Crema de brocoli,Caldo tlalpeño,Sopa mixteca"),
                        Desplegable("Coca cola,Pepsi,Naranjada,Limonada,Agua de sabor"));break;
            case "2. Un primer plato + un segundo plato y una bebida":
                Pedido.CalcularPedido(Desplegable("Crema de espincas,Ensalada de verduras,Crema de brocoli,Caldo tlalpeño,Sopa mixteca"),
                        Desplegable("Filete de pescado,Milanesa de pollo,Bistec a la mexicana,Pollo en escabeche,Carne asada,Lomo relleno,Pollo a la plancha"),
                        Desplegable("Coca cola,Pepsi,Naranjada,Limonada,Agua de sabor"));break;
            case "3. Un primer plato + un segundo plato + una bebida y un postre":
                Pedido.CalcularPedido(Desplegable("Crema de espincas,Ensalada de verduras,Crema de brocoli,Caldo tlalpeño,Sopa mixteca"),
                        Desplegable("Filete de pescado,Milanesa de pollo,Bistec a la mexicana,Pollo en escabeche,Carne asada,Lomo relleno,Pollo a la plancha"),
                        Desplegable("Coca cola,Pepsi,Naranjada,Limonada,Agua de sabor"),
                        Desplegable("Pastel helado,Helado,Fresas con crema,Platanos fritos,Flan casero,Gelatina"));break;
            case "Salir":break;
            }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
    public static String Desplegable(String menu) {
        String valores[]=menu.split(",");
        String res=(String)JOptionPane.showInputDialog(null,"Escoge una opcion","M E N U",
                JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
        return res;
    }
    public static String Desplegable1(String menu) {
        String valores[]=menu.split(",");
        String res=(String)JOptionPane.showInputDialog(null,"Selecciona el tipo de platillos a pedir o salir","M E N U",
                JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
        return res;
    }
}

```

La clase TestPedido es una clase de prueba que tiene un método principal main que llama al método CapturaPedido.

El método CapturaPedido muestra un menú desplegable en una ventana emergente usando la clase JOptionPane de Swing, con cuatro opciones: "Un primer plato y una bebida", "Un primer plato + un segundo plato y una bebida", "Un primer plato + un segundo plato + una bebida y un postre" y "Salir". Dependiendo de la opción seleccionada, se llama al método CalcularPedido de la clase Pedido, que toma diferentes argumentos en función de la opción seleccionada.

El método Desplegable y Desplegable1 son métodos auxiliares que se encargan de mostrar el menú desplegable y devolver la opción seleccionada.

TestPelicula:

```java
package Test;

import EntradaSalida.Tools;

public class TestPelicula {

    public static void main(String[] args) {
        CapturaPeliculas();
    }
    public static void CapturaPeliculas() {
        Pelicula peli1=new Pelicula("Gandhi", "Richard Attenborough", "Drama", (short)191, (short)1982, (byte)8);
        Pelicula peli2=new Pelicula("Thor", "Kenneth Branagh", "Acción", (short)115, (short)2011, (byte)7);
        Tools.imprimePantalla(peli1.toString()+"\n¿Es Epica? "+peli1.esPeliculaEpica2()+"\n Valoracion: "+peli1.CalcularValoracion2()+
                "\n\n"+peli1+"\n"+peli2.toString()+"\n¿Es Epica? "+peli2.esPeliculaEpica2()+"\n Valoracion: "+peli2.CalcularValoracion2()+
                "\n\n ¿Son Similares? "+peli1.esSimilar2(peli2)+"\n\n"+peli1.imprimirCartel2()+"\n\n"+peli2.imprimirCartel2());
    }
}

```

La clase TestPelicula es una clase que tiene un método main que crea dos objetos de la clase Pelicula e imprime información sobre ellos.

El método CapturaPeliculas crea dos objetos Pelicula utilizando el constructor de la clase y asigna valores a sus atributos, que incluyen el título, el director, el género, la duración en minutos, el año de lanzamiento y la valoración. Luego, se imprime información sobre las películas, incluyendo su estado épico y valoración. También se compara si las películas son similares.

TestSalon:

```java
package Test;

import EntradaSalida.Tools;

public class TestSalon {
    public static void CapturaObjetos() {
        Rectangulo salon=new Rectangulo(7,(float)6.5);
        Rectangulo alf1=new Rectangulo((float)3.8,(float)4.6);
        Rectangulo alf2=new Rectangulo((float)4.5,(float)2.3);
        Tools.imprimePantalla(Rectangulo.AreaCubierta(salon, alf1, alf2));
    }
    public static void main(String[] args) {
        CapturaObjetos();
    }
}

```

La clase TestSalon tiene dos métodos: CapturaObjetos y main.

El método CapturaObjetos instancia tres objetos de la clase Rectangulo con diferentes dimensiones, y luego llama al método AreaCubierta de la misma clase, pasándole los tres objetos como argumentos. El método AreaCubierta devuelve una cadena de caracteres que representa el área total cubierta por los tres rectángulos en el salón, y esta cadena se imprime en la pantalla utilizando el método imprimePantalla de la clase Tools.

El método main simplemente llama al método CapturaObjetos para iniciar el proceso. En resumen, este programa calcula el área total cubierta por tres rectángulos en un salón.

Conclusion:

La programación orientada a objetos en Java ofrece muchas herramientas útiles para diseñar programas de manera eficiente. Los métodos son una parte fundamental de esta programación, ya que permiten definir comportamientos y acciones específicas que los objetos pueden realizar.

En este texto, se ha abordado la definición y estructura de los métodos, el valor de retorno, la declaración de métodos de clase e instancia, el ámbito y tiempo de vida de las variables, la sobrecarga de métodos y la utilización de constructores y destructores. Todos estos temas son importantes para el diseño de programas en Java, y su comprensión permite mejorar la capacidad para desarrollar aplicaciones más complejas y funcionales.

En definitiva, el conocimiento sobre POO en Java es fundamental para cualquier desarrollador, y los métodos son una herramienta clave para el éxito en esta área. Con una comprensión completa de los conceptos aquí abordados, se estará en una posición privilegiada para diseñar programas más eficientes y funcionales.

Bibliografía:

KevinDCCsHeOs. (2023, April 16). *KevinDCCsHeOs/Tema3POO*. GitHub. https://github.com/KevinDCCsHeOs/Tema3POO

Guia de reporte de practicas. (2023). Padlet. https://padlet.com/mtzcastillo2023/tema-3-metodos-7ykd9bx1lgv1xqwr/wish/2538860880

TEMA 3 METODOS. (2023). Padlet. https://padlet.com/mtzcastillo2023/tema-3-metodos-7ykd9bx1lgv1xqwr

en, P. (2023). Programación ATS [YouTube Video]. In *YouTube*. https://www.youtube.com/@ProgramacionATS

de, C. (2023). pildorasinformaticas [YouTube Video]. In *YouTube*. https://www.youtube.com/@pildorasinformaticas