Orientación a Objetos I

Sesión Conceptual 02

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercicios
- Quiz

- Cierre

Inicio

{desafío}
latam_

Objetivo

- Comprender el paradigma de la Programación Orientada a Objetos (POO).
- Comprender la estructura e instancia de una clase y sus atributos para aplicar la sobrecarga de métodos.

Desarrollo

{desafío}
latam_

/* Comunidad de objetos */

Sobrecarga de métodos

```
public void aumentarVelocidad(int velocidad){
     velocidadActual = velocidadActual + velocidad;
}


public void aumentarVelocidad(){
     velocidadActual = velocidadActual + 10;
}


public void aumentarVelocidad(boolean maximoCiudad, boolean maximoCarretera){
     if(maximoCiudad) {
          velocidadActual = velocidadActual + 50;
     }
     if(maximoCarretera) {
          velocidadActual = velocidadActual + 100;
     }
}
```

Sobrecarga de métodos

```
5  public class MainClass {
6
7      public static void main(String[] args) {
8          Auto instancia = new Auto();
9          System.out.println("Auto creado");
0          instancia.encenderMotor();
1          instancia.aumentarVelocidad
2      }
3  }
4
```

- **aumentarVelocidad**() : void - Auto
- **aumentarVelocidad**(Integer velocidad) : void - Auto
- **aumentarVelocidad**(Boolean maximoCiudad, Boolear

Método toString()

Proviene desde la clase Object

Permite convertir a String el estado de una instancia.

Método toString()

```
Auto instancia = new Auto();
System.out.println("Auto creado");
instancia.encenderMotor();
System.out.println("Sistema de sonido creado");
SistemaSonido sistemaSonido = new SistemaSonido();
sistemaSonido.setVolumenMaximo(100);
instancia.setSistemaSonido(sistemaSonido);
System.out.println(instancia.toString());
```

Auto creado
Sistema de sonido creado
Modelo.Auto@15db9742

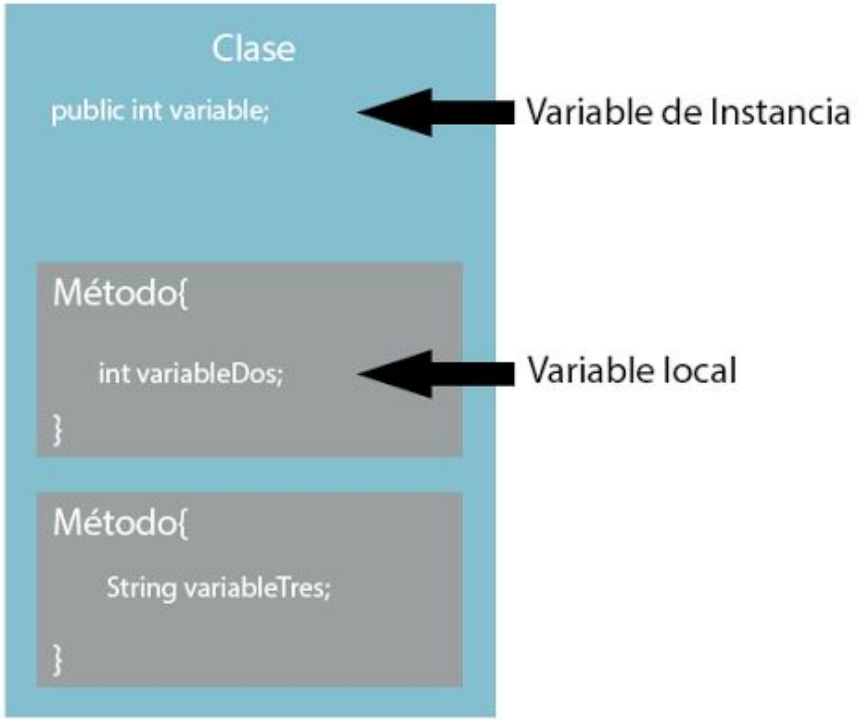
Auto creado
Sistema de sonido creado
Auto [marca=null, modelo=null, color=null, velocidadActual=0, motorEncendido=true, sistemaSonido=Modelo.SistemaSonido@15db9742]

Método toString()

{desafío} latam_

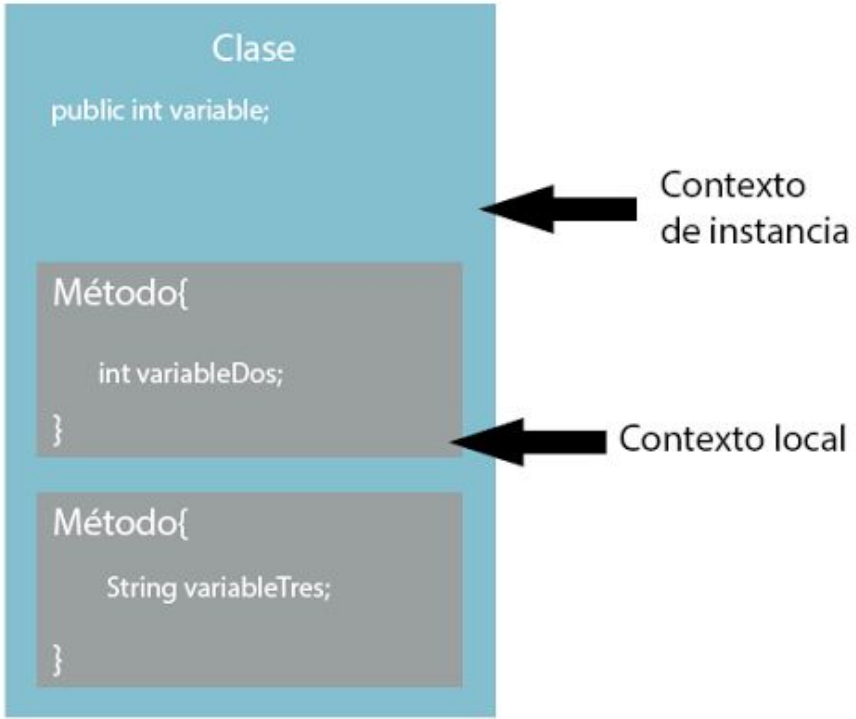
Variables de instancia y variables locales

1. Las variables de instancia:
   1.1. Se declaran fuera de los métodos
2. Las variables locales:
   2.1. Se declaran dentro de los métodos



Variables de instancia y variables locales

1. Las variables de instancia:
   1.1. Están disponibles en el contexto de toda la instancia
2. Las variables locales:
   2.1. Están disponibles en contextos locales o de método



Variables de clase

1. Se pueden llamar sin instanciar la clase que las contiene:

```
Auto.pruebaEstatica
```

```
public class Auto extends Vehiculo{

     public static String pruebaEstatica;
     ...
}
```

Métodos de clase

1. Funcionan igual que las variables de clase, ya que no son llamadas desde una instancia, sino desde una clase

```
double resultado = Utilidades.dividir(1, 5);
```

```
public class Utilidades{

     public static double dividir(double a, double b){
          if(b == 0)
               return 0;
          return (a/b);
     }

}
```

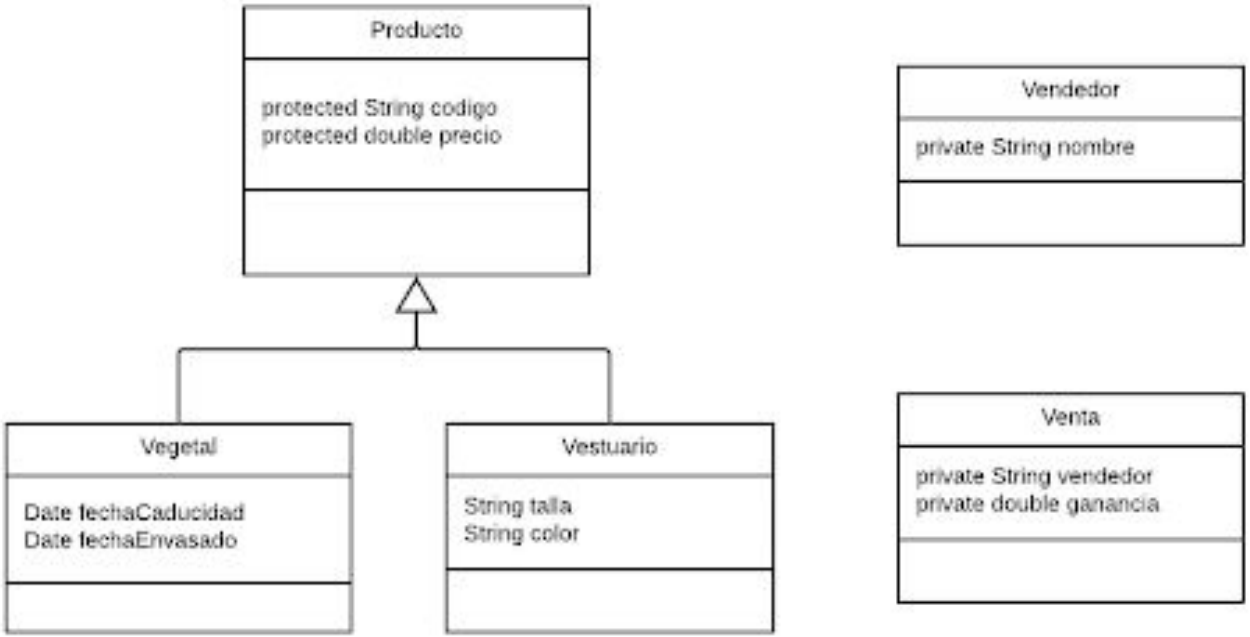
Vamos a practicar un poco más

Aplicación Ventas

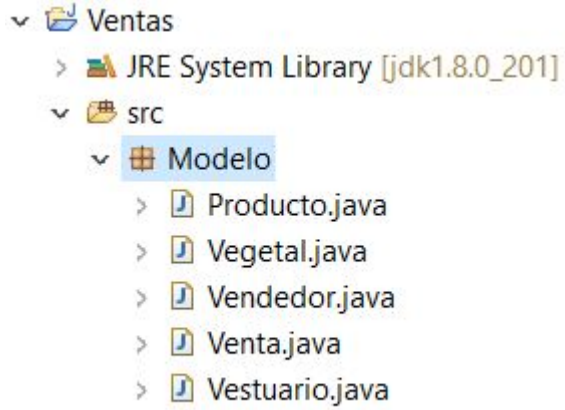
Planificación

- Nos puede ahorrar un dolor de cabeza si llevamos un alto porcentaje del desarrollo terminado y surge un imprevisto con alguna funcionalidad que no fue tomada en cuenta mientras se avanzaba.
- Te ayuda a resolver dudas del proyecto.
- Te ayuda a tener claro lo que harás y a tener las mejores ideas respecto de cómo construir el código para cada parte de tu proyecto.

Planificación



Creación del modelo




```
public class Vegetal extends Producto{
}



public class Vestuario extends Producto{

 }
```

{desafío} latam_

Los atributos de Producto

```java
/**
 * Clase padre de todos los productos del sistema
 */
public class Producto {

    protected double precio;
    protected String nombre;
    protected String codigo;

    /**
     * @return devuelve el precio
     */
    protected double getPrecio() {
        return precio;
    }
    /**
     * @return devuelve el nombre
     */
    protected String getNombre() {
        return nombre;
    }
    /**
```

```java
/**
 * @return devuelve el precio
 */
protected double getPrecio() {
    return precio
}
/**
 * @return devuel
 */
protected String
    return nombre
}
/**
```

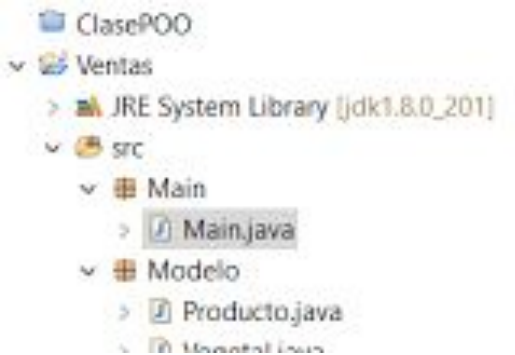
double Modelo.Producto.getPrecio()

Returns:

devuelve el precio

Press 'F2' for focus

El método main de la aplicación Ventas




```
1  package Main;
2
3  public class Main {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7
8      }
9
10 }
```

```
Scanner sc = new Scanner(System.in);

String valorEntrante = sc.nextLine();
```

{desafío} latam_

El método main de la aplicación Ventas

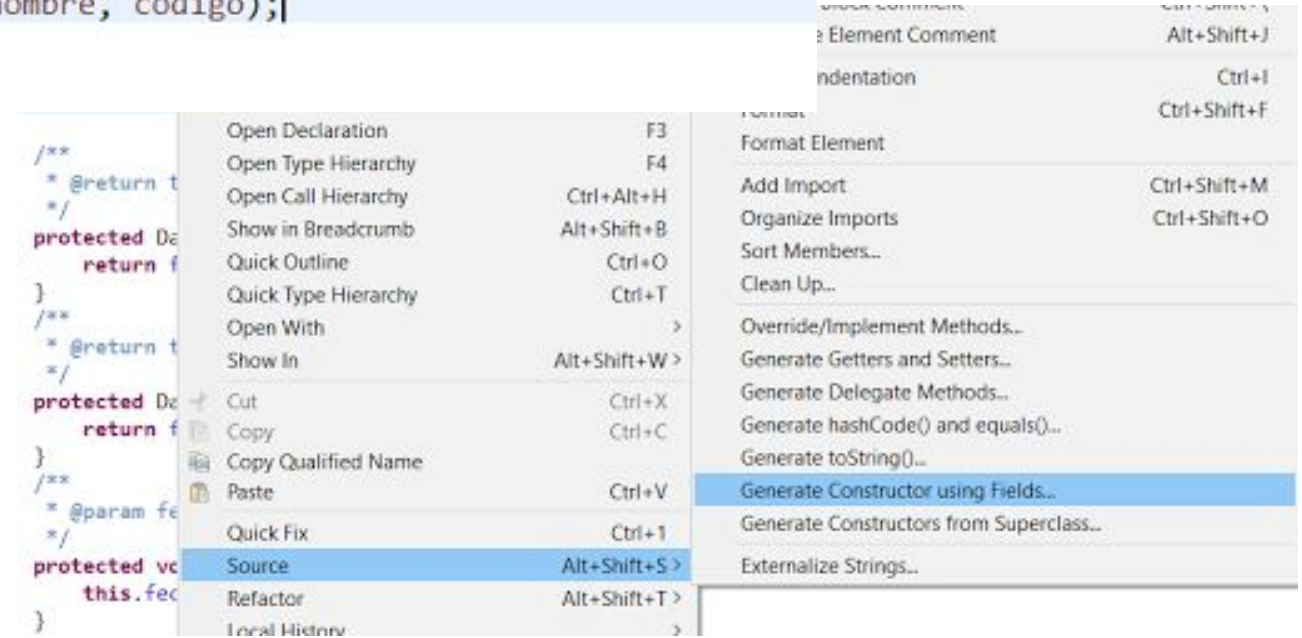
```
System.out.println("Ingrese su nombre de vendedor:");
String nombreVendedor = sc.nextLine();
Vendedor vendedorActual = new Vendedor(nombreVendedor);
System.out.println("Bienvenido "+vendedorActual.getNombre());
```

Constructor con parámetros de vendedor

```
public Vendedor(String nombre) {
     this.nombre = nombre;
}
```

Constructores sobrecargados

```
public class Vegetal extends Producto{

    public Vegetal(double precio, String nombre, String codigo) {
        super(precio, nombre, codigo);
    }
}
```



Los productos

```
ArrayList<Producto> productos = new ArrayList<>();


productos.add(new Vegetal(1000, "Papa", "vgl1", new Date(), new Date()));
productos.add(new Vegetal(800, "Tomate", "vgl2", new Date(), new Date()));
productos.add(new Vestuario(10000, "Camisa", "vst1", "M", "Blanco"));
productos.add(new Vestuario(15000, "Pantalon", "vst2", "M", "Cafe"));
```

El menú del usuario

private

static

Venta

crearVenta

ArrayList<Producto> productos

```java
public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese su nombre de vendedor:");
        String nombreVendedor = sc.nextLine();
        Vendedor vendedorActual = new Vendedor(nombreVendedor);
        System.out.println("Bienvenido "+vendedorActual.getNombre());

        ArrayList<Producto> productos = new ArrayList<Producto>();

        productos.add(new Vegetal(1000, "Papa", "vgl1", new Date(), new Date()));
        productos.add(new Vegetal(800, "Tomate", "vgl2", new Date(), new Date()));
        productos.add(new Vestuario(10000, "Camisa", "vst1", "M", "Blanco"));
        productos.add(new Vestuario(15000, "Pantalon", "vst2", "M", "Cafe"));
    }

    private static Venta crearVenta(ArrayList<Producto> productos) {

    }
```

```java
private static Venta crearVenta(ArrayList<Producto> productos) {
	// Un scanner para ingresar codigos de producto
	Scanner sc = new Scanner(System.in);
	//Un booleano para mantener el ciclo while con vida.
	boolean faltanProductos = true;
	//Una variable para almacenar los codigos que el vendedor ingrese
	String codigoProducto;
	// Un ciclo while que pregunte codigos mientras la variable faltanProductos = true
	while(faltanProductos == true) {
		// Un mensaje para avisarle al usuario que ya puede ingresar codigos
		System.out.println("Ingrese un codigo y enter o solo presione enter para salir de la venta:");
		// Preguntamos por codigos de productos para agregar a la venta
		codigoProducto = sc.nextLine();
		if(codigoProducto.isEmpty()) {
			//Verificamos con isEmpty() si el usuario ingreso algo o no
			// Si el usuario no ingresa nada (el vendedor) vamos a dejar de agregar productos a la venta
			// Cambiando la variable del while a false, para que termine el ciclo.
			faltanProductos = false;
		}else {
			// Aqui agregaremos más codigo
		}

	}
	//Finalmente vamos a devolver ventaActual cuando termine de ejecutarse el metodo
	//recuerda que esta variable es una instancia de Venta
	return ventaActual;
}
```

```
boolean productoExiste = false;
for(Producto producto : productos) {
     /*
     Comparamos el valor ingresado con todos los codigos de la lista de productos
     Si alguno coincide, se suma el precio a la ganancia de la ventaActual
     en el if(productoExiste) que esta al final
     */
     /*
     Los string se comparan con el metodo string1.equals(string2) y no con ==
     Este metodo hace una comparacion mas "correcta" de los string
     y es heredado desde la superclase Object
     */
     if(producto.getCodigo().equals(codigoProducto)) {
          productoExiste = true;
          ventaActual.setGanancia(ventaActual.getGanancia() + producto.getPrecio());
          //Se utiliza break; para detener el for()
          break;
     }
}
if(productoExiste) {
     //Enviamos el mensaje cuando encontremos el producto y sumamos
     System.out.println("Producto agregado");
     System.out.println("Total de la venta: $"+ventaActual.getGanancia());
}
```

```
ArrayList<Venta> ventas = new ArrayList<Venta>();
double gananciaSesion = 0;

boolean cerrarSesion = false;
while(cerrarSesion == false) {
     // Mensajes del menu
     System.out.println("Su total de ventas es: "+gananciaSesion);
     System.out.println("Presione 1 y enter para crear una venta, presione solamente enter para
salir");
     String opcionElegida = sc.nextLine();
     // Leemos la opcion que desea el usuario y si es 1, creamos una venta
     if(opcionElegida.equals("1")) {
          Venta ventaFinal = crearVenta(productos);
          ventaFinal.setVendedor(vendedorActual.getNombre());
          ventas.add(ventaFinal);
          gananciaSesion = gananciaSesion + ventaFinal.getGanancia();
     }else if(opcionElegida.isEmpty()) {
          // Si la opcion esta en blanco, vamos a salir del while
          cerrarSesion = true;
}
```

El método main de la aplicación Ventas

```
System.out.println("Cerrando sesion...");
System.out.println("-  -  -");
System.out.println("Ganancias de la sesion:"+gananciaSesion);
System.out.println("Ventas de la sesion:");
for(Venta venta : ventas) {
     System.out.println(venta.toString());
}
System.out.println("-  -  -");
System.out.println("Hasta pronto "+vendedorActual.getNombre());
```

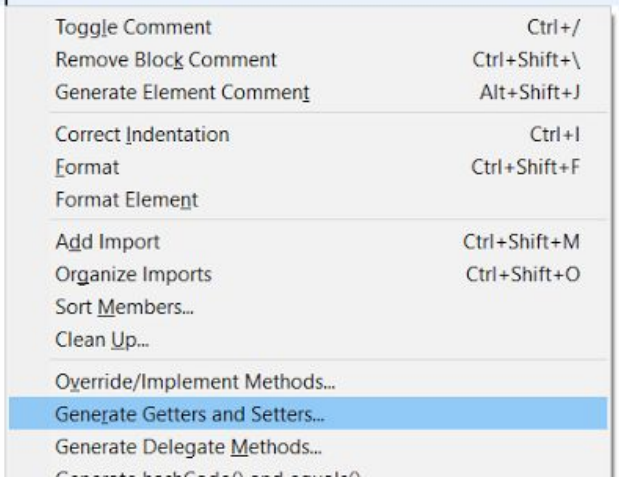
Composición y Asociación

Composición

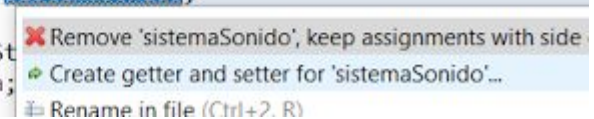
Una clase dentro de otra

Composición

```java
package Modelo;

import java.util.List;

public class SistemaSonido {

    private Integer volumenMaximo;
    private Integer volumenActual;
    private List<String> funciones;
    private String funcionActual;

}
```




```java
public class Auto extends Vehiculo {

    private String marca;
    private String modelo;
    private String color;
    private Integer velocidadActual;
    private Boolean motorEncendido;
    private SistemaSonido sistemaSonido;

    public void setMarca(St
        this.marca = marca;
    }
```



{desafío} latam_

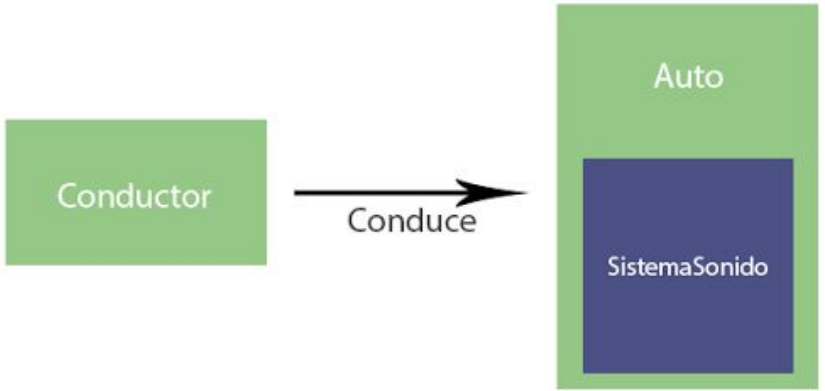
Composición

Se puede probar la composición en el método main:

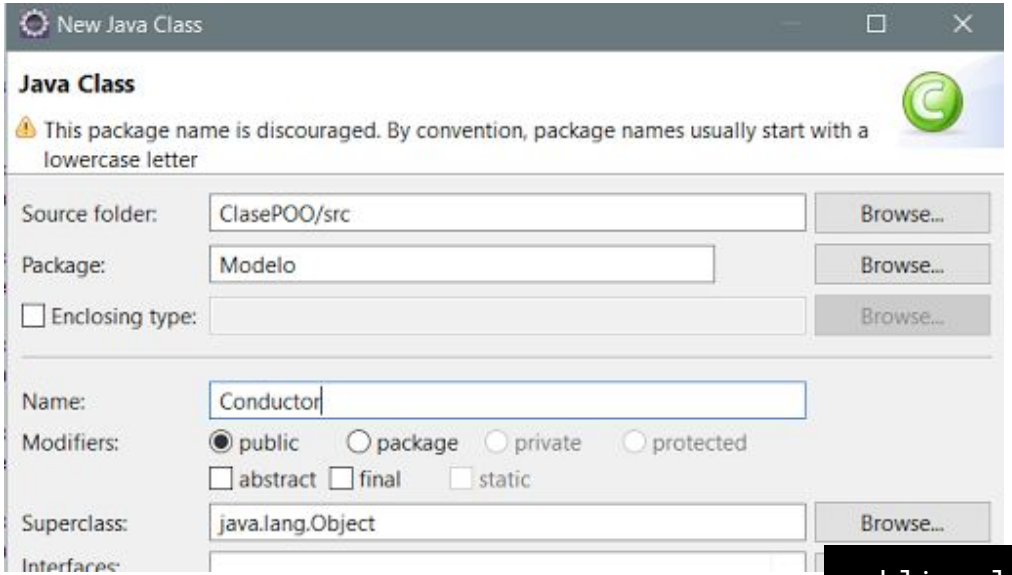
```
import Modelo.SistemaSonido;
public class MainClass {
     public static void main(String[] args) {
          Auto instancia = new Auto();
          System.out.println("Auto creado");
          instancia.encenderMotor();
          System.out.println("Sistema de sonido creado");
          SistemaSonido sistemaSonido = new SistemaSonido();
          sistemaSonido.setVolumenMaximo(100);
          instancia.setSistemaSonido(sistemaSonido);
          System.out.println(instancia.toString());
     }
}
```

Asociación

La diferencia entre composición y asociación es conceptual



Creando la clase Conductor




```
public class Conductor {
     private String nombre;
     private Auto autoConducido;
     //Aqui abajo agregaremos los getter, setter y
la sobrecarga de .toString()
```

{desafío} latam_

Asociación

Se puede probar la asociación en el método main:

```
public static void main(String[] args) {
     Auto instancia = new Auto();
     instancia.setMarca("Opel");
     instancia.setModelo("Corsa");
     instancia.setColor("Blanco");
     Conductor conductor = new Conductor();
     conductor.setAutoConducido(instancia);
     conductor.setNombre("Juan");
     System.out.println("Auto y conductor creados");
     System.out.println(conductor.toString());
}
```

Quiz

{desafío}
latam_

/* Manejo de Excepciones */

Mejorando la aplicación Ventas

Menú para crear productos

Nuevo método en la clase Main

```
private static ArrayList<Producto> menuProductos() {
ArrayList<Producto> listaProducto = new ArrayList<>();
boolean salir = false;
while(salir == false) {
     System.out.println("- - - Menú de creación de productos - - -");
     System.out.println("1. Crear Producto Vegetal");
     System.out.println("2. Crear Producto de Vestuario");
     System.out.println("3. Salir");
     System.out.println("- - -");
     // Un scanner para leer la opción
     Scanner sc = new Scanner(System.in);
     int opcionElegida = Integer.parseInt(sc.nextLine());
}
return listaProducto;
}
```

Menú para crear productos

```
. . .
      int opcionElegida = Integer.parseInt(sc.nextLine());
      if(opcionElegida == 1) {
            System.out.println("Creando producto vegetal");
            listaProducto.add(new Vegetal((Math.random()*10000),"Vegetal","vgl"+listaProducto.size(),new Date(),
new Date()));
      }else if(opcionElegida == 2) {
            System.out.println("Creando producto vestuario");
            listaProducto.add(new Vestuario((Math.random()*10000),"Vestuario","vst"+listaProducto.size(),"XL",
"Rojo"));
      }else if(opcionElegida == 3) {
            System.out.println("Volviendo al menu principal");
            return listaProducto;
      }else {
            System.out.println("Opción desconocida");
      }
      }//< Final del while
      return listaProducto;
}
```

Menú para crear productos

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Ingrese su nombre de vendedor:");
    String nombreVendedor = sc.nextLine();
    Vendedor vendedorActual = new Vendedor(nombreVendedor);
    System.out.println("Bienvenido "+vendedorActual.getNomb

    ArrayList<Producto> productos = new ArrayList<Producto>

//      productos.add(new Vegetal(1000, "Papa", "vgl1", new Dat
//      productos.add(new Vegetal(800, "Tomate", "vgl2", new Da
//      productos.add(new Vestuario(10000, "Camisa", "vst1", "M
//      productos.add(new Vestuario(15000, "Pantalon", "vst2",
    menuProductos();

    ArrayList<Venta> ventas = new ArrayList<Venta>();
    double gananciaSesion = 0;


    boolean cerrarSesion = false;
    while(cerrarSesion == false) {
        // Mensajes del menu
        System.out.println("Su total de ventas es: "+ganan
        System.out.println("Presione 1 y enter para crear u
```

Probando la aplicación

```
Ingrese su nombre de vendedor:
Juan Perez
Bienvenido Juan Perez
- - - Menú de creación de productos - - -
1. Crear Producto Vegetal
2. Crear Producto de Vestuario
3. Salir
- - -
2
Creando producto vestuario
- - - Menú de creación de productos - - -
1. Crear Producto Vegetal
2. Crear Producto de Vestuario
3. Salir
- - -
```

Probando la aplicación

```
Exception in thread “main” java.lang.NumberFormatException: For input string: “”
at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
at java.lang.Integer.parseInt(Integer.java:592)
at java.lang.Integer.parseInt(Integer.java:615)
at Main.Main.menuProductos(Main.java:118)
at Main.Main.main(Main.java:20)
```

Excepciones

Excepciones

Errores en tiempo de ejecución que terminan el programa.

at Main.Main.menuProductos(Main.java:118)

(Main.java:118)

```
int opcionElegida = Integer.parseInt(sc.nextLine());
```

Tipos de excepción

Todas heredan de la clase Exception

Cuando algo falla, se instancia una subclase que nos da información de lo que falló

```
NumberFormatException

NullPointerException
```

Manejo de excepciones

Try - Catch

```
try {
     int total = 3 / 0;
}catch(Exception excepcion) {
     int total = 0;
}
```

Manejo de excepciones

Catch múltiple

```
try {
     Scanner sc = new Scanner(System.in);
     String variable = sc.nextLine();
     if(variable.isEmpty()){
          variable = null;
     }
     int total = 3 / Integer.parseInt(variable);
}catch(NullPointerException ex1) {
     System.out.println("No se puede dividir por un valor nulo.");
     int total = 0;
}catch(NumberFormatException ex2) {
     System.out.println("El valor de variable no es un número.");
     int total = 0;
}catch(Exception ex3){
     System.out.println("Error inesperado: "+ex3.getMessage());
     int total = 0;
}
```

Manejo de excepciones

finally

```
Connection cn = null;
try {
    cn = fuenteDeDatos.getConnection();
}catch(Exception e){
     System.err.out("Ha ocurrido un error");
} finally {
     cn.close();
}
```

Manejo de excepciones

throws, throw y Excepciones personalizadas

Para entender esto, vamos a ver un ejemplo de la creación de una excepcion llamada `DivisionPorCero` y la pondremos dentro de un método `dividir` que reciba dos numeros a dividir y a través de un `if`, valide que ninguno de los dos sea 0; Si alguno de los dos es 0, se va a utilizar un `throw` para arrojar la excepción `DivisionPorCero` que creamos:

```
public static class DivisionPorCero extends Exception {
      DivisionPorCero(){
            super("No es posible dividir por cero");
      }
}

public static void main(String[]  args){
      int  resultado;
      try{
            resultado=dividir(5,0);
            opcionElegida = Integer.parseInt(sc.nextLine());
      }catch(DivisionPorCero ex){
            System.err.println(ex);
      }finally{
            resultado=0;
      }
      System.out.println("Resultado:"+resultado);
}

public static int dividir(int a, int b)throws DivisionPorCero{
      if(b==0){
            throw new DivisionPorCero();
      }else{
            return  a  /  b;
      }
}
```

Resultado:

test.test$DivisionPorCero: No es posible dividir por cero

Manejo de excepciones en aplicación Ventas

Agregando un try catch se resuelve el problema

```
int opcionElegida = 0;
try {
     opcionElegida = Integer.parseInt(sc.nextLine());
}catch(Exception otraExcepcion) {
     otraExcepcion.printStackTrace();
     opcionElegida = 0;
}
```

```
public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Ingrese su nombre de vendedor:");
      String nombreVendedor = sc.nextLine();
      Vendedor vendedorActual = new Vendedor(nombreVendedor);
      System.out.println("Bienvenido "+vendedorActual.getNombre());

      ArrayList<Producto> productos = new ArrayList<Producto>();

      productos = menuProductos();

      System.out.println("- - - Productos Disponibles - - -");
      for(Producto producto : productos) {
            System.out.println(productos.indexOf(producto)+"-"+producto.getCodigo());
            System.out.println(" - Precio: $"+producto.getPrecio());
      }
      System.out.println("- - -");

      ArrayList<Venta> ventas = new ArrayList<Venta>();
      double gananciaSesion = 0;
```

```
boolean cerrarSesion = false;
      while(cerrarSesion == false) {
            // Mensajes del menu
            System.out.println("Su total de ventas es: "+gananciaSesion);
            System.out.println("Presione 1 y enter para crear una venta, presione solamente enter para salir");
            String opcionElegida = sc.nextLine();
            // Leemos la opcion que desea el usuario y si es 1, creamos una venta
            if(opcionElegida.equals("1")) {
                  Venta ventaFinal = crearVenta(productos);
                  ventaFinal.setVendedor(vendedorActual.getNombre());
                  ventas.add(ventaFinal);
                  gananciaSesion = gananciaSesion + ventaFinal.getGanancia();
            }else if(opcionElegida.isEmpty()) {
                  // Si la opcion esta en blanco, vamos a salir del while
                  cerrarSesion = true;
            }
      }

      System.out.println("Cerrando sesion...");
      System.out.println("-  -  -");
      System.out.println("Ganancias de la sesion:"+gananciaSesion);
      System.out.println("Ventas de la sesion:");
      for(Venta venta : ventas) {
            System.out.println(venta.toString());
      }
      System.out.println("-  -  -");
      System.out.println("Hasta pronto "+vendedorActual.getNombre());
}
```

Aplicación lista

Quiz

{desafío}
latam_

Cierre

{desafío}
latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales