UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

Control de lectura 2

Autor

Alexandra Lalaleo

Materia:

Programación Orientada a Objetos

NRC:

1323

TUTOR:

Ing. Luis Enrique Jaramillo Montaño

Sangolquí – Ecuador

Tema de la Actividad:

Creación de Objetos y UML

Tipo de Actividad:

Diseño y Modelado

Descripción de la Actividad:

Diseñe 5 objetos diferentes con su correspondiente diagrama UML, asegurándose de mostrar las relaciones entre ellos.

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

PROGRAMACIÓN ORIENTADA A OBJETOS

ACTIVIDAD DE APRENDIZAJE 2.

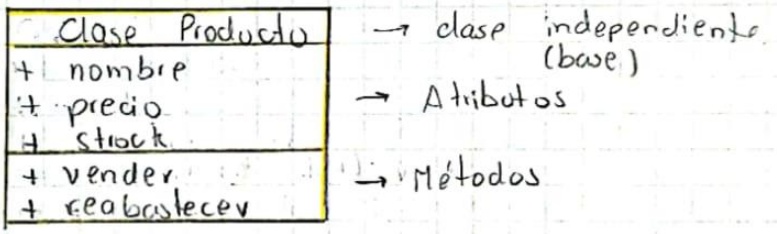
NOMBRE: Alexandra Lalaleo
NRC: 1323
TEMA: Creación de objetos y UML.

Diseñe 5 objetos diferentes con su correspondiente diagrama UML, asegurándose de mostrar las relaciones entre ellos.

SISTEMA DE TIENDA DE ROPA

1. Producto

Diagrama UML



CÓDIGO.

```
public class Producto {
    public String nombre;
    public double precio;
    public int stock;

    public Producto (String nombre, double precio, int stock) {
    this.nombre = nombre;
    this.precio = precio;
    this.stock = stock;
    }

    public void vender (int cantidad) {
    if (stock >= cantidad) {
       stock- = cantidad;
     } else {
```

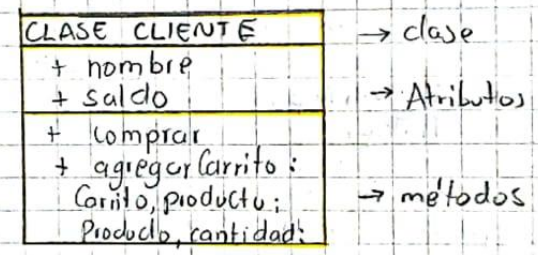
```
            System.out.println("Stock insuficiente.");
        }
    }

    public void reabastecer(int cantidad){
        stock += cantidad;
    }
}
```

Resumen:

Representa los productos disponibles en una tienda, con atributos como nombre, precio y stock. Tiene métodos como vender y reabastecer.

2. CLIENTE

Diagrama UML



CÓDIGO

```
public class Cliente {
    public String nombre;
    public double saldo;

    public Clientes(String nombre, double saldo) {
        this.nombre = nombre;
        this.saldo = saldo;
    }
    public void comprar(Carrito carrito){
        double total = carrito.calcularTotal();
        if (saldo) >= total){
            saldo -= total;
            System.out.println("Compra realizada con éxito.");
            }else{
            System.out.println("Saldo insuficiente.");
        }
    }
```

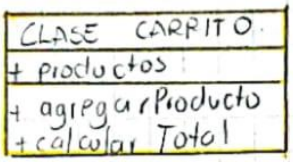
```
    public void agregarCarrito(Carrito carrito, Producto producto, int cantidad)
{ carrito.agregarProducto(producto, cantidad);
  }
}
```

Resumen:

Representa un cliente que compra productos. Tiene atributos como nombre y saldo. Tiene métodos como comprar y agregar carrito

3. CARRITO

Diagrama UML.



CÓDIGO

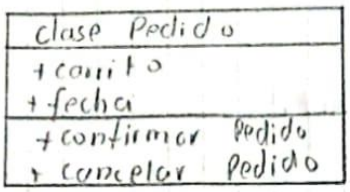
```
import java.util.HashMap;
import java.util.Map;

public class Carrito {
  public Map<Producto, Integer> productos;

  public Carrito (){
    productos= new HashMap<>();
  }

  public void agregarProducto (Producto , in cantidad) {
    productos.put (producto, productos.getOrDefault (producto, 0) +
    cantidad);
    }

    public double calcularTotal(){
    double total=0
    for (Map.Entry<Producto, Integer> entry: productos.entrySet()){
        total+= entrygetKey().precio * entry.getValue();
        }return total;
    }
  }
}
```

Resumen

Representa un carrito de compras. Contiene una lista de productos y tiene métodos como agregar Producto y calcular Total.

4. Pedido

Diagrama UML.



CÓDIGO.

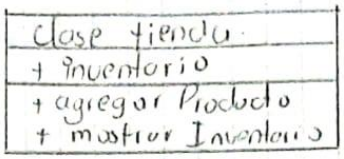
```
import java.time.LocalDate;

public class Pedido {
public Carrito Carrito;
public LocalDate fecha;

Public Pedido (Carrito carrito) {
this. Carrito = carrito;
this. fecha = LocalDate.now();
}
public void confirmar Pedido() {
    System.out.println ("Pedido confirmado el:" + fecha);
    }

public void cancelarPedido(){
    System.out.println ("Pedido cancelado.");
}
}
```

Resumen:

Representa un pedido confirmado de un cliente. Contiene el carrito y la fecha del pedido. Métodos como confirmar Pedido y cancelar Pedido.

5. Tienda.

Diagrama UML.



código.

```
import java.util.ArrayList;
import java.util.List;

public class Tienda {
  public List<Producto> inventario;

  public Tienda (){
    inventario = new ArrayList<>();
  } public void agregarProducto (Producto producto) {
    inventario.add (producto);
  }

  public void mostrarInventario (){
  for (Producto producto: inventario){
    System.out.println ("Producto: " + producto.nombre +
    ", Precio: " + producto.precio + ", Stock: " + producto.stock);
      }
  }
}
```

Resumen.

Administra productos y clientes. Métodos como agregar Producto y mostrar producto.

Informe sobre la Tarea de Modelado de Objetos

Introducción

Se presenta el desarrollo de un sistema de programación orientado a objetos que modela una tienda virtual. Este sistema incluye cinco clases principales: Producto, Cliente, Carrito, Pedido y Tienda. A continuación, se describen los atributos, métodos, diagramas UML y relaciones entre las clases, junto con el código para su implementación.

Descripción de los Objetos

Producto

Representa los productos disponibles en una tienda, con atributos como nombre, precio y stock. Tiene métodos como vender y reabastecer.

Cliente

Representa un cliente que compra productos. Tiene atributos como nombre y saldo. Tiene métodos como comprar y agregar Carrito.

Carrito

Representa un carrito de compras. Contiene una lista de productos y tiene métodos como agregar Producto y calcular Total.

Pedido

Representa un pedido confirmado de un cliente. Contiene el carrito y la fecha del pedido. Métodos como confirmar Pedido y cancelar Pedido.

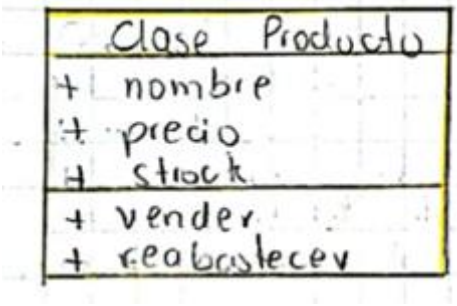
Tienda

Administra productos y clientes. Métodos como agregar Producto y mostrar Inventario.

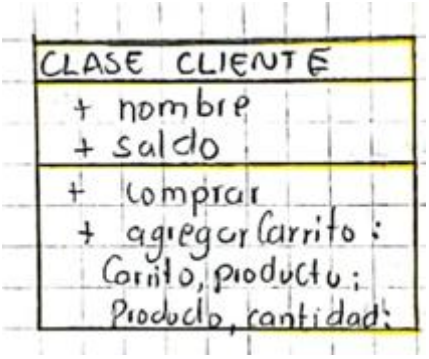
Diagramas UML

Se adjunta los diagramas UML de cada clase individual, mostrando sus atributos y métodos.

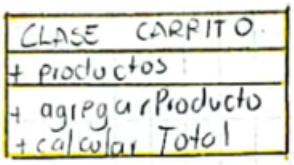
Clase Producto



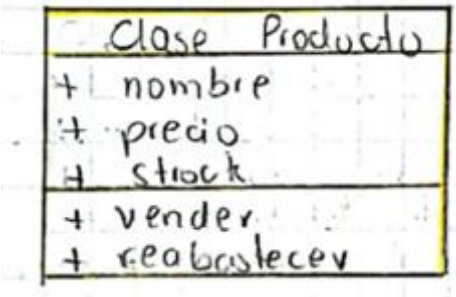
Clase Cliente



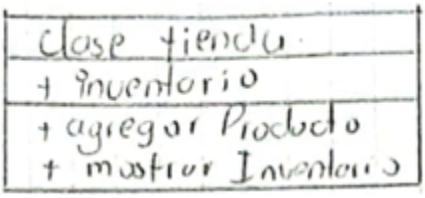
Clase Carrito



Clase Pedido



Clase Tienda



Relaciones UML

- **Asociación** entre Cliente y Carrito.
- **Composición** entre Pedido y Carrito.
- **Asociación** entre Tienda y Producto.

Código del sistema

```
// Clase Producto

public class Producto {

    public String nombre;

    public double precio;

    public int stock;



    public Producto(String nombre, double precio, int stock) {

        this.nombre = nombre;

        this.precio = precio;

        this.stock = stock;

    }



    public void vender(int cantidad) {

        if (stock >= cantidad) {

            stock -= cantidad;

        } else {
```

```
            System.out.println("Stock insuficiente.");
        }
    }

    public void reabastecer(int cantidad) {
        stock += cantidad;
    }
}

// Clase Cliente
public class Cliente {
    public String nombre;
    public double saldo;

    public Cliente(String nombre, double saldo) {
        this.nombre = nombre;
        this.saldo = saldo;
    }

    public void comprar(Carrito carrito) {
```

```
        double total = carrito.calcularTotal();
        if (saldo >= total) {
            saldo -= total;
            System.out.println("Compra realizada con éxito.");
        } else {
            System.out.println("Saldo insuficiente.");
        }
    }

    public void agregarCarrito(Carrito carrito, Producto producto, int cantidad) {
        carrito.agregarProducto(producto, cantidad);
    }
}

// Clase Carrito
import java.util.HashMap;
import java.util.Map;

public class Carrito {
    public Map<Producto, Integer> productos;
```

```
    public Carrito() {
        productos = new HashMap<>();
    }

    public void agregarProducto(Producto producto, int cantidad) {
        productos.put(producto, productos.getOrDefault(producto, 0) + cantidad);
    }

    public double calcularTotal() {
        double total = 0;
        for (Map.Entry<Producto, Integer> entry : productos.entrySet()) {
            total += entry.getKey().precio * entry.getValue();
        }
        return total;
    }
}

// Clase Pedido
import java.time.LocalDate;
```

```
public class Pedido {
    public Carrito carrito;
    public LocalDate fecha;

    public Pedido(Carrito carrito) {
        this.carrito = carrito;
        this.fecha = LocalDate.now();
    }

    public void confirmarPedido() {
        System.out.println("Pedido confirmado el: " + fecha);
    }

    public void cancelarPedido() {
        System.out.println("Pedido cancelado.");
    }
}

// Clase Tienda
```

```
import java.util.ArrayList;
import java.util.List;

public class Tienda {
    public List<Producto> inventario;

    public Tienda() {
        inventario = new ArrayList<>();
    }

    public void agregarProducto(Producto producto) {
        inventario.add(producto);
    }

    public void mostrarInventario() {
        for (Producto producto : inventario) {
            System.out.println("Producto: " + producto.nombre + ", Precio: " +
producto.precio + ", Stock: " + producto.stock);
        }
    }
```

Este sistema demuestra la relación entre los objetos mediante asociaciones y permite gestionar compras en una tienda de manera efectiva.