Informe de C/C++

OPTIMIZAR EL REGISTRO DEL INVENTARIO DE UNA TIENDA DE CELULARES

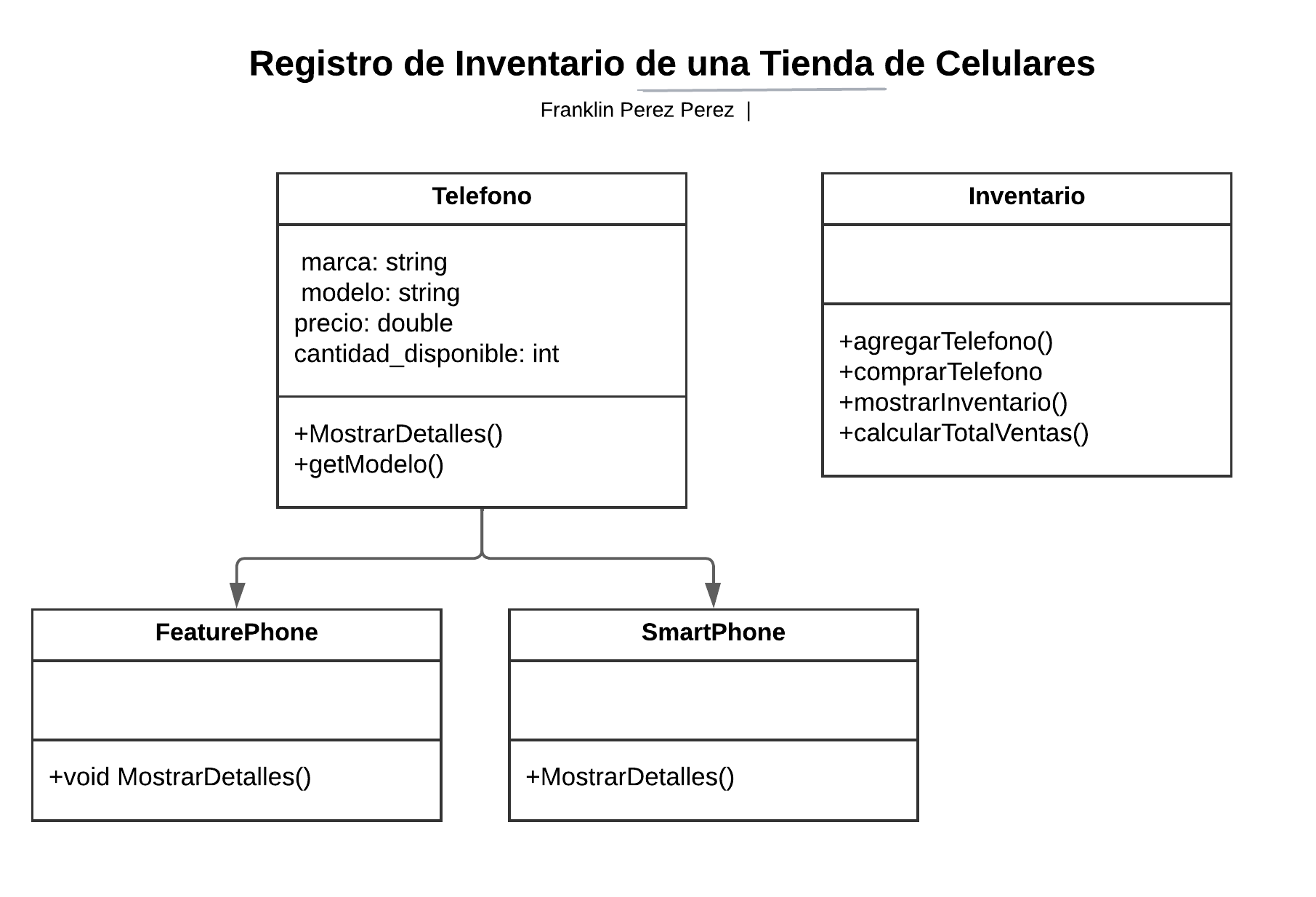
FRANKLIN PÉREZ PÉREZ

Introducción

En este informe se detalla el diseño y la implementación de un sistema de gestión de inventario para una tienda de celulares en línea. El sistema permite a los usuarios realizar diversas acciones, como agregar celulares al inventario, buscar celulares por marca o modelo, comprar celulares disponibles y calcular el total de ventas.

**Objetivo:**

El objetivo principal de este proyecto es desarrollar un sistema eficiente y robusto que facilite la gestión del inventario de una tienda de celulares en línea, permitiendo a los usuarios acceder a una amplia variedad de celulares, realizar compras y mantener un registro actualizado de las ventas.



***Funcionalidades del Sistema***

1-Agregar Celulares al Inventario

void Inventario::agregarTelefono(Telefono\* telefono) {  
 listaInventario.push\_back(telefono);  
}

El método `agregarTelefono()` en la clase `Inventario` permite añadir dinámicamente objetos `Telefono` al inventario de la tienda. Toma un puntero a un objeto `Telefono` como argumento y lo agrega al final del vector `telefonos` dentro del objeto `Inventario`. Es importante destacar que este método no crea una copia del teléfono, sino que almacena un puntero al teléfono pasado como argumento. Por lo tanto, es responsabilidad del código que llama a este método garantizar que el objeto `Telefono` referenciado por el puntero siga existiendo mientras esté en uso en el inventario.

2-Mostrar Inventario Completo

void Inventario::mostrarInventario() {  
 cout<<"Inventario"<<'\n';  
 if (listaInventario.empty())cout<<"No tenemos telefonos"<<'\n';  
 int cont = 1;  
 for (auto telefono: listaInventario) {  
 cout<<cont++<<"-";  
 telefono->MostrarDetalles();  
 }  
}

El método `mostrarInventario()` en la clase `Inventario` imprime los detalles de todos los teléfonos presentes en el inventario de la tienda. Itera a través de cada teléfono en el inventario y llama al método `mostrarDetalles()` de cada uno para mostrar su información específica.

3-Comprar Celulares

bool Inventario::comprarTelefono(string modelo, int cantidad) {  
 for (auto telefono : listaInventario) {  
 if (telefono->modelo == modelo && telefono->cantidad\_disponible >= cantidad) {  
 telefono->cantidad\_disponible -= cantidad;  
 return true;  
 }  
 }  
 return false;  
}

El método `comprarTelefono()` en la clase `Inventario` permite a los clientes comprar teléfonos del inventario de la tienda. Verifica la validez del ID proporcionado y la disponibilidad del teléfono seleccionado. Si la compra es exitosa, actualiza la cantidad disponible del teléfono en el inventario y muestra un mensaje de confirmación.

4-Calcular el Total de Ventas

int Inventario::calcularTotalVentas() {  
 int aux = 0;  
 for (int t: listaTlefonosVendidos) {  
 aux += t;  
 }  
 return aux;  
}

- El sistema es capaz de calcular el total de ventas realizadas desde que comenzaron a vender.

- Los informes de ventas incluirán datos relevantes, como el total de ingresos generados y el número de productos vendidos

**Características Adicionales;**

- Utilización de programación orientada a objetos (POO) para diseñar clases como `Celular`, `Inventario`, y `Usuario`.

- Implementación de herencia para definir subclases específicas de celulares, como `Smartphone`, `FeaturePhone`, etc.

- Uso de sobrecarga de operadores para facilitar la comparación de celulares por marca o modelo.

- Empleo de la Standard Template Library (STL) de C++ para estructuras de datos como vectores o mapas para almacenar los celulares y usuarios.

**Conclusiones**

El desarrollo de este sistema de gestión de inventario ha sido un proceso enriquecedor que ha permitido aplicar diversos conceptos de programación avanzada, como la programación orientada a objetos, la herencia, la sobrecarga de operadores y el uso de la STL en un contexto práctico. El sistema resultante proporciona una solución eficiente y escalable para la gestión del inventario de una tienda de celulares en línea.

## Futuras Mejoras

- Implementación de un sistema de autenticación de usuarios para garantizar la seguridad de las transacciones.

- Incorporación de funcionalidades adicionales, como la generación de informes de ventas y la gestión de inventario en tiempo real.

-Crear una interfaz de usuario para una experiencia más intuitiva y atractiva para los usuarios finales.