UNED ETS de Ingeniería Informática

1-6-2018

Sistema Integrado de Gestión de una Tienda de Electrodomésticos

Programación Orientada a Objetos

Consuelo Pinto Toro Dni: Y2778738F
CENTRO ASOCIADO: UNED CERVERA

Contenido

Introducción

En el presente documento se pondrán en juego los conocimientos de java estudiados durante el segundo cuatrimestre: implementando conceptos básicos de programación orientada a objetos, el objetivo principal de este documento es ofrecer una posible solución a la práctica de la asignatura “TIENDA DE ELECTRODOMESTICOS” del presente curso académico 2017-2018.

Para su correcto funcionamiento he intentado aplicar las funciones principales de este problema como la abstracción, sobre escritura, herencia y encapsulamiento, también utilice la interfaz recomendada por el equipo docente que fue BlueJ.

Se pretenderá realizar un programa de gestión automatizada de una tienda de electrodomésticos, es decir la gestión de ella misma, durante el desarrollo del programa tuve que tener una serie de consideraciones a la hora de construir el programita.

En nuestra tienda tendremos que manejar las siguientes entidades principales que son:

- **Clientes:** cada uno de los clientes puede comprar un producto, crear una ficha, cliente tener opciones de financiación, servicio de postventa, y servicio técnico y este a su vez recibirá la herencia de la clase **persona.**

- **Empleado:** en la sección de empleados se podrá crear un nuevo empleado visualizar los pagos por hora extra, sueldo, base horas extras tipo IRPF, si está casado i/o tiene hijos y a su vez recibirá la herencia de la clase persona donde se gestionará también todo lo que nos pide el enunciado de la práctica:

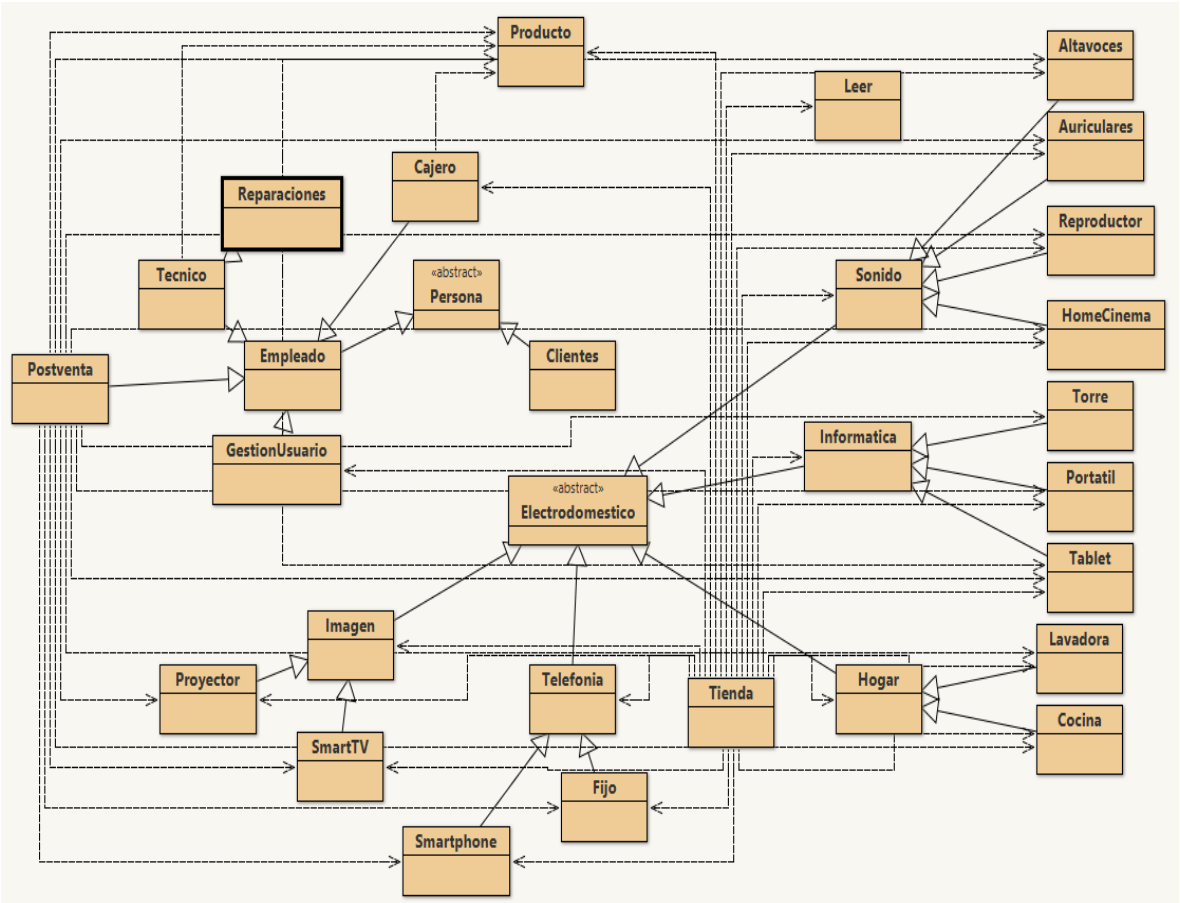
    1. Venta de un electrodoméstico (cajero).
    2. Financiación (financiación)
    3. Reparación de Electrodomésticos (técnico).
    4. Devolución de electrodoméstico (postventa).
    5. Gestión comercial (comercial).
    6. Gestión de usuarios.
    7. Gestión de clientes.

- **Electrodoméstico:** en la sección de electrodomésticos se podrá visualizar la marca, modelo, precio unidad, color, consumo, peso, donde también se encontrarán las subclases de Hogar, Informática, Sonido, Telefonía, Imagen.

    1. **Informática:** Torre, Portátil, Tablet
    2. **Hogar:** Lavadora, Cocina.
    3. **Telefonía:** Smartphone, Fijo

4. **Imagen:** Proyector, SmartTV
5. **Sonido**: Altavoces, Auriculares, Reproductor, Home Cinema.

I. Nivel 1: Presentación del diagrama de clases.



Como vemos en la imagen anterior se ve el diagrama de la tienda conformado con veinte y nueve clases y subclases en las cuales solo enseñare los tipos de variables y algo de fragmentos de código de las clases y sus atributos principales.

La función del **Main** donde se pueden encontrar el menú **comprar** y **vender** productos si están **disponibles** en la tienda, se puede comprar, **crear un nuevo cliente, añadir** más productos en la compra actual y consultar el **importe total** de la cuenta.

También hare una breve descripción de Clase **Persona** tal como nos dice el enunciado de la práctica nos dice que creemos con sus respectivos atributos que serán:

- **Nombre.**
- **Apellidos.**
- **Dni.**
- **Sexo.**
- **Edad.**
- **Dirección**
- **Teléfono**

Todos estos atributos los heredaran las clases **Empleado** y **Cliente,** en la clase empleado lo inicialicé con los siguientes atributos:

- **Pago por hora extra.**
- **Sueldo Base.**
- **Cuantas horas extras que realizo.**
- **IRPF.**
- **Casado.**
- **Número de hijos.**

II. Nivel 2: Gestión básica de electrodomésticos y clientes.

La gestión de la **clase electrodomésticos** esta de modo abstracto donde heredaran los atributos **Hogar, Informática, Sonido, Imagen, Telefonía,** a continuación, las variables tal y como nos dice el enunciado de la práctica.

- **Marca**
- **Modelo**
- **Precio de la unidad**
- **Color**
- **Consumo**
- **Peso**

Dentro de la clase electrodomésticos están las variables o atributos mencionadas anteriormente con sus respectivos getters y setters donde verificaremos mediante condicionales la comprobación de la **marca**, calculamos el **precio**, del electrodoméstico lo dejo predefinido, en color blanco y mediante un switch enumero los casos y les designo las letras que comúnmente están en los electrodomésticos habituales, también hago la comprobación de color con condicionales y el precio final del electrodoméstico.

- En la siguiente imagen se encuentra un fragmento de código como se en la imagen se ve la comprobación de las marcas que dispondrá la tienda.

```
public void comprobarMarca(String marca) {
        if  (  marca == "Apple"   || marca == "Asus"
            || marca == "Sony"    || marca == "HP"
            || marca == "Huawei"  || marca == "Epson"
            || marca == "Balay"   || marca == "Bosch"
            || marca == "LG"      || marca == "Sharp"
            || marca == "Jata"    || marca == "Romba"
            || marca == "Irobot"  || marca == "Solac"
            || marca == "Philips" || marca == "Rowenta"
            || marca == "Siemens" || marca == "Daewo"
            || marca == "Toshiva" || marca == "Canon"
            || marca == "JVC"     || marca == "Fagor"
            || marca == "Teka"    || marca == "Lynx"
            || marca == "Nodor"   || marca == "Panasonic"
            || marca == "Braun"   ||  marca == "Krups") {
            this.marca = marca;
        } else {
            this.marca = "";
        }
    }
```

- Calculamos el precio con condicionales para que nos devuelva el precio unitario del producto este método lo utilice teniendo en cuenta que los productos ya sea por peso o bajo consumo puede variar en el precio de cada unidad.

```
//Con un condicional calculo el precio
private double calcularPrecio() {
    double pesoPorPrecio = 0;

    this.comprobarConsumoEnergetico(this.consumo);

    if(this.peso >= 0 && this.peso <= 19) {
        pesoPorPrecio = 19;
    } else if(this.peso >= 20 && this.peso <= 49) {
        pesoPorPrecio = 50;
    } else if(this.peso >= 50 && this.peso <= 79) {
        pesoPorPrecio = 80;
    }   else {
        pesoPorPrecio = 100;
    }


    this.precioUnit+= pesoPorPrecio;

    return this.precioUnit;
}
```

- Resaltando las características más relevantes de la clase electrodomésticos comprobaremos el consumo energía mencionado anteriormente.

```
public void comprobarConsumoEnergetico(char letra) {
    boolean exist = false;
    //compruebo el consumo energetico  y las pongo con letras
    switch(letra) {
        case 'A':
            exist = true;
            this.precioUnit = 100.00;
        break;
        case 'B':
            exist = true;
            this.precioUnit = 80.00;
        break;
        case 'C':
            exist = true;
            this.precioUnit = 60.00;
        break;
        case 'D':
            exist= true;
            this.precioUnit = 50.00;
        break;
        case 'E':
            exist = true;
            this.precioUnit = 30.00;
        break;
        case 'F':
            exist = true;
            this.precioUnit = 10.00;
        break;
    }
     if(exist) {
        this.consumo = letra;
    }
}
```

- Para concluir la comprobación del color del electrodoméstico también mencionado en el enunciado de la práctica.

```
public void comprobarColor(String color) {
    if(color == "Negro" || color == "Azul"
        || color == "Gris" || color == "Rojo") {
        this.color = color;
    } else {
        this.color = "Blanco";
    }
}

public double precioFinal() {
    return calcularPrecio();
   }
}
```

III. Nivel 3: Compras.

En nuestra clase clientes tendremos tres identificadores para la financiación del cliente no hará falta crear más ya que tenemos la herencia de la clase **Persona** a continuación las siguientes:

- **Ultima nomina**
- **Solicita financiación**
- **Concedida financiación**
- Más la herencia de la clase persona que lo mencionamos anteriormente que es **Nombre, Apellidos, Dni, Sexo, Edad, Dirección, Teléfono.**

Así como también se podrá pagar en **caja,** se encontrará en la **clase Cajero** donde nos encontraremos con las variables**.**

- **Producto productos**
- **Cajero**
- **Cliente**

A continuación, un fragmento de código donde mediante un for coja la lista de los productos y nos lo muestre por pantalla:

```
public void mostrarProductos(Producto[] productos) {
        for (int i = 0; i < productos.length; i++) {
                System.out.print(productos[i] + "\n-------\n");
        }
}
```

En la imagen mostraremos el fragmento para que no enseñe el nombre de los artículos de la tienda.

```
public void mostrarNombreProductos(Producto[] productos) {
        for (int i = 0; i < productos.length; i++) {
                System.out.println(i + 1 + " " + productos[i].getNombre() + "\n");
        }
        System.out.println("\n-----------\n");
}
```

Gestión de compra por unidades, verificación si el producto está disponible y si no hay stock además el precio de la unidad.

```
public double comprarProducto(Producto[] productos, int num,
                int cantidadUnidades) {
        if (productos[num - 1].isDisponible()) {
                if (productos[num - 1].getCantStock() >= cantidadUnidades) {
                        System.out.println("La compra se ha realizado con éxito!!\n");
                        productos[num - 1].setCantStock(productos[num - 1]
                                        .getCantStock() - cantidadUnidades);
                        return cajero+= cantidadUnidades
                                        * productos[num - 1].getPrecioUnit();
                } else {
                        System.out.println("No hay cantidad suficiente de producto");
                }
        } else {
                System.out.println("No hay cantidad suficiente de producto");
        }
        return cajero;
}
```

También la clase cajero está vinculada con la clase productos donde esta los atributos o variables como:

- **Identificador**
- **Precio Unidad**
- **Disponibilidad de stocks**
- **Disponibilidad del producto**
- **Dimensión de la Array**

```
public String toString() {
        return "ID: " + this.getID() + "\n" + "Precio unidad: "
                        + this.getPrecioUnit() + " €\n" + "En Stock: "
                        + this.getCantStock() + "\n";
}
```

IV. Nivel 4: Reparaciones.

Así como nuestro cliente podrá contar con los seis tipos de servicio de que es la venta de un electrodoméstico, la financiación encontrado en la clase **cliente** el servicio **Técnico** y servicio de **Postventa** que lo comentaremos dentro de poco, cabe decir que servicio técnico contara con los siguientes atributos:

- **Garantía**
- **Garantía Plus**
- **Fecha de comienzo**
- **Fecha de finalización**
- **Numero de reparación**
- **Tipo de producto**
- **Solicitud de reparación**
- **Caja**
- Este a su vez les dará una la herencia a la clase **Reparaciones** donde nos encontraremos el **Producto** y así generará un nuevo electrodoméstico en **reparación** con sus fechas respectivas

  Aquí un fragmento de código de los constructores.

```
public Reparaciones(Producto productoRep, Date start, Date finish, int number) {
    super(sueldoBase, horasExtras, tipoIRPF, casado, numeroHijos, nombre);
    this.productoRep = productoRep;
    this.start = start;
    this.finish = finish;
    this.number = number;
    this.Producto = Producto;

}
```

  En principio tendría que generar un nuevo **producto en reparación**.

```
    public Reparaciones(Producto producto,int number){
    this.producto = productoRep;
    this.start = new productoRep();
    this.finish = new productoRep ();
    this.number = number;
}
```

V. Nivel 5: Devoluciones.

En el servicio devoluciones nos lo encontraremos en la clase de **Postventa** nos encontraremos con la herencia de clase empleado donde se gestionarán las devoluciones y los cambios de algún tipo de producto, también teniendo en cuenta las fecha devolución del producto

- **Producto**
- **Devolución**
- **Fecha de reembolso**
- **Fecha de cambio de producto**
- **Identificador de la devolución**

Dentro de la misma clase **Postventa** crearemos una nueva **devolución** y les designaremos el formato de fecha de la devolución y mediante una instanciación en principio era la creación de una comprobación de todos los productos que se encuentran disponibles en la tienda con sus respectivos constructores getters y setters tuve bastantes problemas con desarrollar el código y no obtuve los resultados óptimos con la idea previa que tenía a continuación un fragmento de código.

En el fragmento de código nos encontraremos la devolución del producto asi como también gestionar el cambio de producto.

```
this.producto = producto;
this.start = new devolucion();
this.devolucion = devolucion;
this.start = start;
this.cambioproducto = cambioproducto;
this.number = number;
this.ID = ID;
```

Intentando darle un formato a la fecha de **devolución** del producto.

```
public String toString() {
    String format = "dd/MM/yy";
    SimpleDateFormat formater = new SimpleDateFormat(format);
    StringBuilder str = new StringBuilder();

    str.append(number);
    str.append(" ");
    StringBuilder number = str.append(producto.getNumber());
```

VII. Biografía

Toda la información necesaria para realizar la práctica, y como ha sido la documentación (API) sobre los diferentes métodos y del tipo de clases de java utilizadas, ha sido obtenida de la página oficial de Oracle en la plataforma de java Standart en la edición 7 haciendo soporte de java, en esta página se encuentra información más detallada cada una de las clases.

https://docs.oracle.com/javase/7/docs/api/