**Modelo programa tienda**

**Descripción general:**

Este software permite el manejo simple de una tienda, es decir, se manejan productos y compras (lo que vende la tienda) y descuentos en las compras por lo tanto no se tiene manejo de proveedores ni nada relacionado por el estilo siguiendo este oren de ideas tampoco se maneja un inventario, es decir los productos no tienen cantidades a la hora de comprar

**Manejo de archivos:**

Este software utiliza un manejo de archivos basado en locaciones, este crear una carpeta general en la cual dentro de ella se crea subcarpetas de cada clase a utilizar y dentro de ellas los archivos guardados correspondientes a cada creación

**Definición de clases:**

|  |
| --- |
| Clase Persona |

|  |  |  |
| --- | --- | --- |
| Clase Administrador | Clase Empleado | Clase Cliente |

**Persona:**

* **Código**
* **Nombre**
* **Apellido**
* **Id**
* **Celular**

clase abstracta con su respectivo constructor y sus métodos get y set

**administrador:**

* **Código**
* **Nombre**
* **Apellido**
* **Id**
* **celular**

clase que hereda de la clase abstracta persona, tiene su respectivo método constructor, como hereda de persona no necesita métodos get ni set, como esta no es una clase abstracta se debe guardar un registro de cada administrador creado para esto se tienen los métodos:

* **guardar datos:** guarda en un registro en la carpeta de administrador todos los datos del administrador creado
* **recuperar datos:** va a la ubicación del registro y extrae los datos para su manipulación
* **modificar datos:** recibe la ubicación del archivo y un string celular, va al registro y modifica el dato a entrar

**empleado:**

* **Código**
* **Nombre**
* **Apellido**
* **Id**
* **Celular**
* **Número de ventas**
* **Estado activo o inactivo**

clase que hereda de la clase abstracta persona, tiene su respectivo método constructor, tiene sus respectivos métodos get y set para los dos atributos los cuales son propios de esta clase (número de ventas, estado activo o inactivo), el atributo estado activo o inactivo es booleano

**métodos para guardar el registro:**

* **guardar datos:** guarda en un registro en la carpeta de empleado todos los datos del empleado creado
* **recuperar datos:** va a la ubicación del registro y extrae los datos para su manipulación
* **modificar datos:** recibe el archivo, un Long celular, int número de ventas y un boolean activo, va al registro y modifica los datos a entrar
* **modificar estado activo:** recibe el archivo y un boolean activo va al registro y modifica el dato a entrar.

**Clase cliente:**

* **Código**
* **Nombre**
* **Apellido**
* **Id**
* **Celular**
* **Puntos**

clase que hereda de la clase abstracta persona, tiene su respectivo método constructor, tiene sus respectivos métodos get y set para el atributo de esta clase el cual es puntos

métodos para guardar el registro:

* **guardar datos:** guarda en un registro en la carpeta de cliente todos los datos del cliente creado
* **recuperar datos:** va a la ubicación del registro y extrae los datos para su manipulación
* **modificar datos:** recibe el archivo, un long celular y un int puntos, va al registro y modifica los datos a entrar
* **modificar puntos:** recibe el archivo y un int puntos, va al registro y modifica el dato a entrar.

**producto:**

* **código**
* **nombre**
* **precio**

esta clase no hereda de la clase persona, tiene su respectivo constructor y sus respectivos métodos get y set

**métodos para guardar el registro:**

* **guardar datos:** guarda en un registro en la carpeta de cliente todos los datos del producto creado
* **recuperar datos:** va a la ubicación del registro y extrae los datos para su manipulación
* **modificar datos:** recibe el archivo y un String precio, va al registro y modifica los datos a entrar

**Criticas**

* No se tiene claridad en el modelo de clases, es decir, no hay una forma bien estructurada de modelo vista Controlador ya que se tiene el modelo, pero la vista y el controlador están juntos, es decir, no se tiene una separación entre estos aspectos
* El atributo celular y el atributo id son de tipo Long, deberían ser de tipo Sting ya que en ningún momento se opera con estos atributos, creando computaciones innecesarias al momento de convertirlos de un tipo Long a un tipo String
* En la vista al momento de modificar un producto no se informa que se está modificando y solo se piden los valores, debería mostrar cómo estaba el producto y luego como queda después de los cambios

|  |
| --- |
| **PERSONA** |
| * String codigo; * String id; * String nombre; * String apellidos; * String celular; |

|  |
| --- |
| **CLIENTE** |
| * private int puntos; |
| * public void setPuntos(int puntos) * public int getPuntos() * public void guardarDatos(File f) * public void consultarDatos(File f) * public void modificarDatos(File f, String celular, int puntos) * public void modificarPuntos(File f, int puntos) |

|  |
| --- |
| **EMPLEADO** |
| * private int numeroVentas; * private Boolean activo; |
| * public void setNumeroVentas(int numeroVentas) * public int getNumeroVentas() * public void setActivo(Boolean activo) * public Boolean getActivo() * public void guardarDatos(File f) * public void consultarDatos(File f) * public void modificarDatos(File f, String celular, int numeroVentas, Boolean activo) * public void modificarActivo(File f, Boolean activo) |

|  |
| --- |
| **BONO** |
| * private String codigo; * private String descuento; |
| * public void setCodigo(String codigo) * public String getCodigo() * public void setDescuento(String descuento) * public String getDescuento() * public void guardarDatos(File f) * public void consultarDatos(File f) * public void modificarDatos(File f, String descuento) |

|  |
| --- |
| **ADMINISTRADOR** |
|  |
| * public void guardarDatos(File f) * public void consultarDatos(File f) * public void modificarDatos(File f, String celular) |

|  |
| --- |
| **PRODUCTO** |
| * private String codigo; * private String nombre; * private String precio; |
| * public void setCodigo(String codigo) * public String getCodigo() * public void setNombre(String nombre) * public String getNombre() * public void setPrecio(String precio) * public String getPrecio() * public void guardarDatos(File f) * public void consultarDatos(File f) * public void modificarDatos(File f, String precio) |

|  |
| --- |
| **MANEJOPRODUCTO** |
|  |
| * public void AgregarProducto(String codigo, String nombre, String precio) * public void EditarProducto ( String codigo, String precio) * public boolean ConsultarProductoExistente (String codigo) * public void ConsultarProducto (String codigo) * public String DevolverTotal (String codigo) |

|  |
| --- |
| **MANEJOBONO** |
|  |
| * public void CrearBono (String codigo, String descuento) * public boolean ConsultarBonoExistente (String codigo) * public void UtilizarBono (String codigo, int var2) * public void ListarBonos () |

|  |
| --- |
| **MANEJOEMPLEADO** |
|  |
| * public void AgregarEmpleado (String codigo, String identificacion, String nombre,String apellidos, String celular, int numeroVentas, Boolean activo) * public void ConsultarEmpleado (String identificacion ) * public void Estadoempleado (String identificacion) * public void ListarEmpleados () |

|  |
| --- |
| **MANEJOCLIENTE** |
|  |
| * public void AgregarCliente (String codigo ,String id, String nombre, String apellidos ,String celular,int puntos ) * public void UtilizarPuntos (String codigo, int var) * public void AgregarCompra (String codigo, int total) |

**Cambios al programa basado en las críticas:**

* Principalmente comencé cambiando el modelo, separando el controlador de la vista ya que se tenían controladores en los botones de la interfaz así que implemente nuevas clases con nuevos métodos para los manejos del modelo separando el modelo de la vista del controlador, reflejado en la gráfica posterior. conservando la implementación de la herencia como ya se utilizaba antes
* Cambie todas las variables de tipo Long por String por la razón de lo anterior mencionado en las criticas
* Al momento de modificar un producto se informa como estaba el producto y luego se informa como quedo según las modificaciones
* En la interfaz compra cambie algunos jtext por joption e implemente un nuevo botón de buscar al momento de ver el total de un producto para consultar si el producto que se va a comprar existe
* Al momento de finalizar la compra se verifica si el cliente existe, esto para saber si se sumaran los puntos o no y se informa si existe si se sumaron puntos o de lo contrario solo se notifica compra realizada
* Al momento de utilizar los puntos o un bono se informa cuanto será el total con el descuento incluido

|  |
| --- |
| **PERSONA** |

|  |
| --- |
| **ADMINISTRADOR** |

|  |
| --- |
| **CLIENTE** |

|  |
| --- |
| **EMPLEADO** |

|  |
| --- |
| **PRODUCTO** |

|  |
| --- |
| **BONO** |

|  |
| --- |
| **MANEJO PRODUCTO** |

|  |
| --- |
| **MANEJO**  **CLIENTE** |

|  |
| --- |
| **MANEJO EMPLEADO** |

|  |
| --- |
| **MANEJO**  **BONO** |

|  |
| --- |
| **INTERFAZ** |