Curso 2019 2020

Practica de Programación Orientada a Objetos

Curso 2019-2020

Contacto:
Teléfono: 634 867 637
Correo: orico9@alumno.uned.es

OSCAR RICO MARTINEZ
ORICO9@ALUMNO.UNED.ES
Teléfono: 634 867 637

CONTENIDO:

ANALISIS DE LA APLICACIÓN

El funcionamiento del programa es el siguiente:

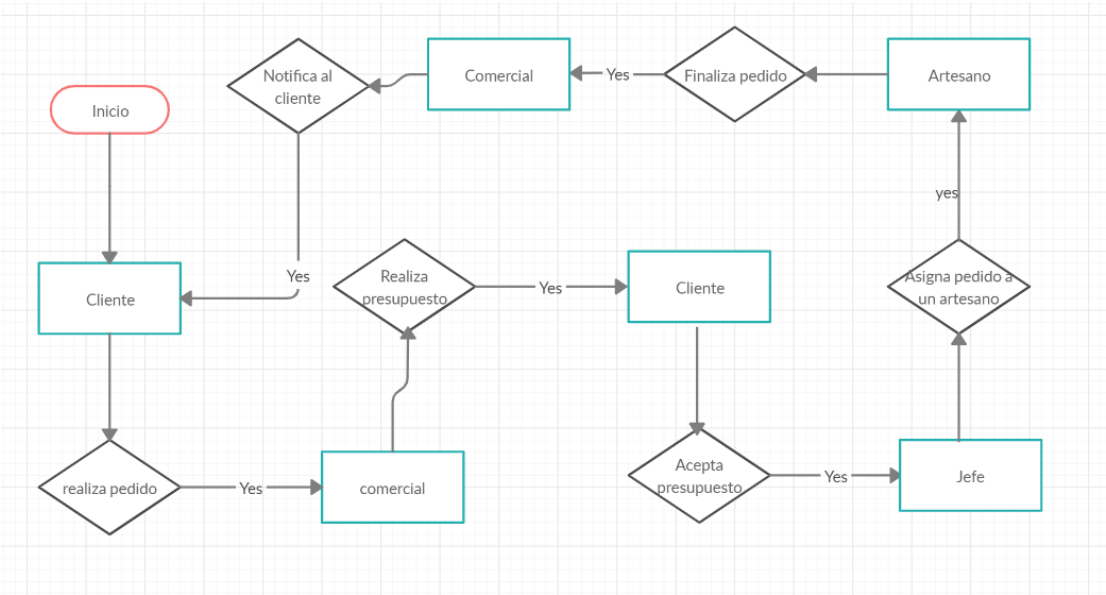
En el área *cliente*. Un cliente registra una cuenta con su nombre y adquiere automáticamente un *ID* (identificador de cliente), accede a la cuenta con el nombre del cliente y dentro de la cuenta puede realizar pedidos de diferentes muebles, consultar pedidos ya realizados, confirmar pedidos cuando el comercial asigne un precio, y recibir mensajes del comercial cuando un pedido esté listo para recoger.

En el área empleado. He decidido, para facilitar la corrección de la práctica, que el jefe y el comercial no necesitan ningún parámetro para acceder a sus gestiones, pero para acceder a las gestiones es necesario que tanto jefe como comercial estén en plantilla, en cambio, los artesanos necesitan el id para acceder a gestionar los pedidos.

El recorrido de un pedido en la fábrica es el siguiente:

Cuando el cliente realiza un pedido se asocia el ID del cliente al pedido, en el mismo instante y de manera automática el pedido crea una referencia, el comercial podrá, en ese momento, asignarle un precio al pedido teniendo en cuenta las características, por ejemplo; tipo de mueble, tamaño, cantidad, etc., una vez asignado el precio, el cliente podrá rechazar el pedido y será eliminado de la base de datos o podrá confirmar el pedido, en cuyo caso, el jefe de la empresa se encargara de asignar el pedido a un artesano, para ello, se muestra una lista de artesanos con el nombre y el id de cada uno, también se muestra la cantidad de trabajos activos que tiene asignado cada artesano, todo ello para hacer una asignación más coherente, cuando el jefe asigna un pedido a un artesano, el ID de dicho artesano se asocia con el estado del pedido y el pedido cambia al estado “en espera” el artesano cuenta con una lista de pedidos a realizar y para actualizar el estado de un pedido necesita introducir el número de *referencia* que tiene el pedido, existen tres estados, Fabricando, incompleto, completado, el artesano al finalizar el pedido actualiza el estado en “completado” y en ese momento el comercial puede ver que, hay pedidos completados y podrá enviar un mensaje al cliente para informarle de que puede pasar a recoger dicho pedido. El cliente recibirá el mensaje y se cerrará el ciclo lógico del proceso de fabricación.

Este sería el resumen de todo el procedimiento explicado anteriormente:



Interpretación del enunciado

Según el enunciado, la fabricación de todos los muebles se hace bajo demanda por el hecho de que el coste de almacenaje supondría un problema, en consecuencia, he interpretado que en cada pedido se deben definir unas características básicas como por ejemplo las medidas del mueble que el cliente desea y que por otro lado no puede haber un stock disponible sino más bien un catálogo de opciones generales como es el de los distintos tipos de mesas y sillas que la fábrica puede realizar.

Decisiones de diseño:

He optado por presentar un diseño lo más claro, ordenado y preciso posible, por ello, en el desarrollo del problema me he centrado en definir únicamente los atributos necesarios para la realización de la práctica, excluyendo campos como el número de teléfono, DNI y otros ejemplos que podría citar, pero que resulta innecesario, bajo mi criterio, para el desarrollo correcto del problema.

Después de estudiar y plantear diferentes estrategias para abordar el problema, he decidido implementar la siguiente estructura:

En la clase Sistema he declarado cuatro ArrayList que almacenan, Empleados, Clientes, Pedidos y Mensajes, con sus respectivos setter y getter. El criterio que he seguido para poner a los Empleados y Clientes en diferentes ArrayList ha sido, como ya he citado antes, la claridad y establecer un orden, además de una búsqueda más rápida y efectiva.

Respecto a los muebles y el pedido, he decidido que cada mueble se considera un pedido y que ese pedido puede constar de múltiples muebles siempre y cuando sean de las mismas características, es decir, las mismas medidas, el razonamiento lógico que he seguido es poder asignar los pedidos entre diferentes Artesanos, me explico, un cliente pide una silla y una mesa, son dos pedidos, el jefe puede asignar un pedido a cada artesano y la velocidad de fabricación aumenta, con lo cual, el cliente obtendrá su pedido con mayor velocidad, si el mueble es de la misma forma y medida lo puede realizar un único artesano aprovechando los patrones.

Con respecto a las asociaciones, he decidido asociar el ID del cliente al pedido para más tarde asociar el ID del artesano encargado de su fabricación. Para que el comercial pueda realizar un presupuesto debe cumplirse la condición de que el precio del pedido sea igual a cero, lo que significa que el pedido todavía no ha recibido un presupuesto, de igual manera, un cliente puede considerar un presupuesto siempre que se cumpla la condición de que el precio sea mayor que cero, teniendo en cuenta que el pedido contiene el ID del cliente y que no ha sido confirmado con anterioridad, el jefe podrá asignar un pedido cuando resulte ser cierta la condición de "confirmado", de manera parecida un artesano tiene acceso a un pedido asignado siempre y cuando el pedido contenga el ID del artesano que trata de realizarlo.

DIAGRAMA DE CLASES

Diagrama completo:

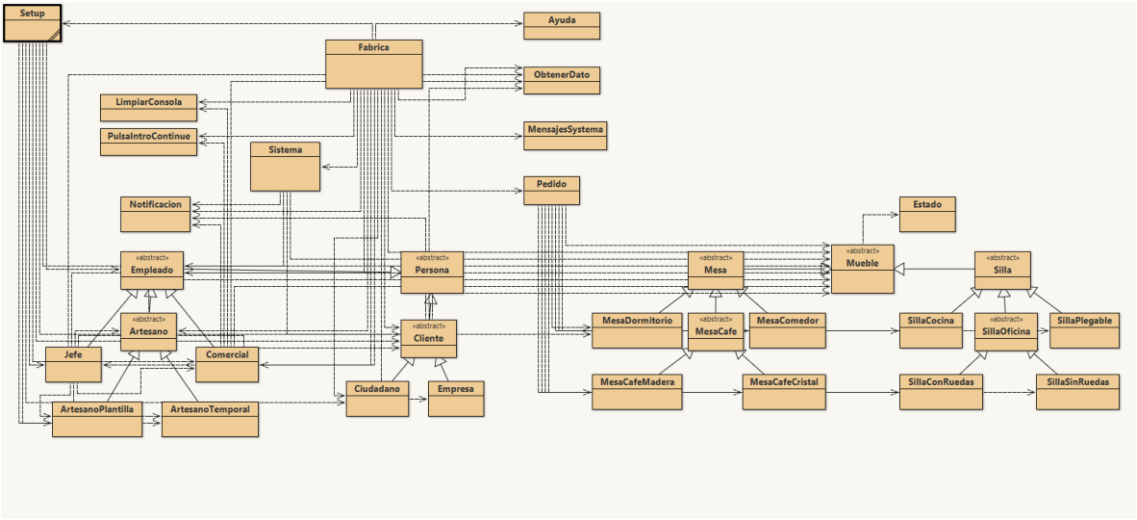


Diagrama Mueble:

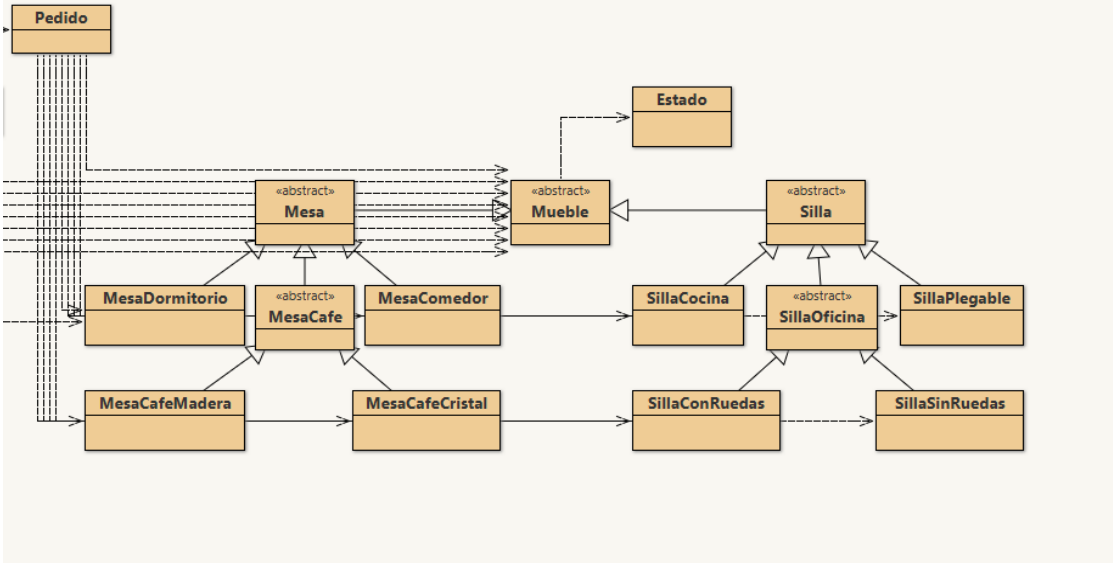
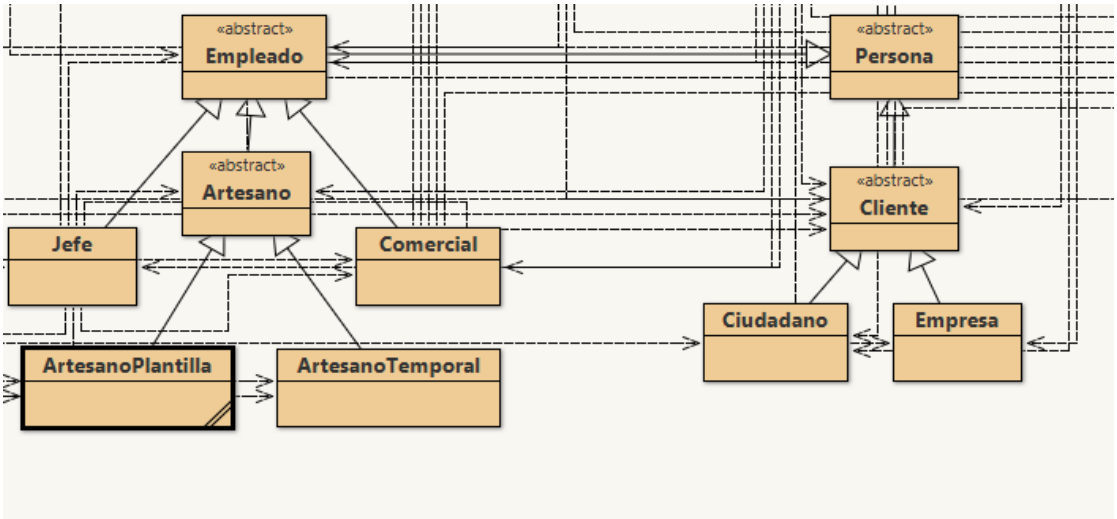


Diagrama persona :



CLASES

A continuación, la descripción de cada clase y una explicación de la funcionalidad que realizan los métodos más importantes.

Clase persona: Clase abstracta *que define a una persona, he decidido abstraerme de los datos que no he considerado relevantes para desarrollar la práctica, con lo cual, la clase persona tiene únicamente el atributo de nombre.*

Métodos públicos:
public void setNombre(String nombre)
public void getNombre()
public String toString()

Clase Empleado: Clase abstracta que hereda de la clase Persona, he decidido que, aparte del nombre que hereda, el empleado tendrá un sueldo y un numero de id *que será asignado automáticamente cuando el empleado es dado de alta, para ello tiene una variable static que contabiliza el número de empleados que se van contratando y se asigna el valor que tenga dicha variable al id empleado cuando es creado, resumiendo, el id es el dato que se usa para realizar las búsquedas, asignaciones, etc., por otra parte el empleado ocupara un lugar de trabajo dentro de la empresa por ello se inicializa una variable de tipo String que será definida por las clases que hereden de la misma.*

Métodos públicos:
public void setSueldo(int sueldo)
public int getSueldo()
public void setPuesto(String puesto)
public String getPuesto()
public void setId(int id)
public int getId()
public String toString()

Clase Comercial: Clase que hereda de la clase Empleado, define el campo "puesto" como "Comercial" y adquiere nuevas funcionalidades que paso a detallar; Gestiona la comunicación con el cliente, realiza presupuesto de nuevos pedidos, gestiona el informe de pedidos completados y con ello notifica al cliente del estado del producto.

Métodos públicos:
public void presupuesto()
public void verPedidoCompletado()
public Notificacion envioMensaje(int id, String asunto, String mensaje)
public String toString()

Clase Jefe: Clase que hereda de la clase Empleado, define el atributo puesto en "Jefe" y no adquiere nuevos atributos, pero si funcionalidades que paso a detallar; Gestiona el funcionamiento de la fábrica, historial de trabajos que realiza cada artesano, acceso al estado de cada pedido, historial de cada mueble y con ello me refiero a la visualización de la cantidad de pedidos de un tipo de mueble, asignación de un pedido a un artesano, informe de empleados contratados, informe de clientes registrado con el historial total de pedidos realizados.

Métodos públicos:
public Jefe(String nombre, int sueldo)
public void contratarArtesanoFijo(String nombre, int sueldo, ArrayList<Empleado> arrayEmpleado)
public void contratarArtesanoTemporal(String nombre, int sueldo, ArrayList<Empleado> arrayEmpleado)
public void contratarComercial(String nombre, int sueldo, ArrayList<Empleado> arrayEmpleado)
public void contratarJefe(String nombre, int sueldo, ArrayList<Empleado> arrayEmpleado)
public void despedirEmpleado(int id, ArrayList<Empleado> arrayEmpleado)
public void listarPlantilla(ArrayList<Empleado> arrayEmpleado)
public void GestionPedidos(ArrayList<Empleado> arrayEmpleado, ArrayList<Mueble> arrayPedido)
public void estadoPedidos(ArrayList<Mueble> arrayPedido)
public void listarClientes(ArrayList<Cliente> arrayCliente)
public void historialArtesanos(ArrayList<Empleado> arrayEmpleados)
public String toString()

Clase Artesano: Clase que hereda de la clase Empleado y que define el campo "puesto" como "Artesano" además inicializa en null el campo "tipoContrato" que será definido por las clases que hereden de la clase Artesano, tambien se inicializan dos campos nuevos, uno contabiliza el número de pedidos que tiene asignados actualmente y el segundo contabiliza el total de pedidos gestionados desde que el artesano es contratado, esta clase contiene todas las funcionalidades que van a realizar los artesanos de la fábrica, a continuación paso a explicar dichas funcionalidades; un artesano gestiona los trabajos que le han sido asignados y actualiza los estados de cada pedido, con ello el artesano podrá listar los trabajos que tiene asignados y actualizar el estado en consecuencia, para realizar esta tarea el artesano introduce el número de referencia del pedido que desea actualizar y selecciona el nuevo estado, si el pedido es actualizado en el estado "completado" el artesano dejara de ver dicho pedido entre sus próximos trabajos y se descontara del registro de pedidos activos.

Métodos públicos:
public Artesano(String nombre, int sueldo)
public String historial()
public void setTipoContrato(String tipoContrato)
public String getTipoContrato()
public void setNumPedidos()
public int getNumPedidos()
public int getHistorialPedidos()
public void descontarPedido()
public void verTrabajo(ArrayList<Mueble> mueble)
public void actualizaEstadoFabricando(int referencia, ArrayList<Mueble >arrayPedido)
public void actualizaEstadoIncompleto(int referencia,ArrayList<Mueble> arrayPedido)
public void actualizaEstadoFinalizado(int referencia,ArrayList<Mueble> arrayPedido)
public String toString()

Clase ArtesanoPlantilla: Clase que hereda de la clase Artesano y que únicamente define el tipo de contrato, con lo cual, define el campo "tipoContrato" con "contrato fijo", Un artesano en plantilla realizara todas las funciones que hereda de la clase abstracta Artesano

Métodos públicos:
public String toString();

Clase ArtesanoTemporal: Clase que hereda de la clase Artesano y que únicamente define el tipo de contrato, con lo cual define el campo "tipoContrato" con "contrato fijo",

Métodos públicos:
public String toString();

Clase cliente: Clase que hereda de la clase persona, en esta clase se definen nuevos campos, hay un campo llamado idCliente y otro campo que es static llamado idClienteSiguiente que contabiliza el número de clientes registrados, cuando un cliente se registra adquiere como id el valor que tenga idClienteSiguiente y después idClienteSiguiente se incrementa para el siguiente usuario que necesite registrarse, con ello se obtiene un identificador único para cada cliente, también se ha declarado otro campo llamado totalPedidos que contabiliza el total de pedidos realizados por un cliente, esta clase tiene diversas funcionalidades que paso a explicar; Realizar pedidos de los diferentes tipos de muebles asociando el idCliente al pedido, listar los pedidos realizados, rechazar o aceptar el presupuesto que previamente el comercial a asignado, acceder a los mensajes que el comercial le mande.

Métodos públicos:
public int getIdCliente();
public void setTotalPedidos();
public int getTotalPedidos();
public void confirmarPresupuesto();
public void listarPresupuestos();
public void mensajesBuzon();
public abstract String getTipoCuenta();
public String toString();

Clase Ciudadano: Clase que hereda de la clase Cliente y que únicamente define el tipo de cuenta, con lo cual define el campo tipoCuenta en "ciudadano".

Métodos públicos:
public String getTipoCuenta();
public String toString();

Clase Empresa: Clase que hereda de la clase Cliente y que únicamente define el tipo de cuenta, con lo cual el campo tipoCuenta es definido como "empresa".

Métodos públicos:
public String getTipoCuenta();
public String toString();

Clase Mueble: Clase abstracta que contiene las características generales de un mueble, el propósito de la clase mueble es definir de manera generalizada el comportamiento base que ha de tener un mueble como pedido de compra de un cliente.

Métodos públicos:
public void setPrecio(int precio)
public int getPrecio()
public void setCantidad(int cantidad)
public int getCantidad()
public int getRef()
public int referencia()
public void setIdCliente(int idCliente)
public int getIdCliente()
public String toString()

Clase Mesa: Clase abstracta que hereda de la clase Mueble y que define las medidas que puede tener una mesa.

Métodos públicos:
public Mesa(int idCliente, int alturaCm, int largoCm, int anchoCm, int cantidad)
public int getAltura()
public int getLargo()
public int getAncho()
public String getMesa()
public String toString()

Clase MesaCafe: Clase abstracta que hereda de la clase Mesa y que incorpora un nuevo campo para dotar a la mesa de cristal y que será inicializado en las subclases.

Métodos públicos:
public MesaCafe(int idCliente, int alturaCm, int largoCm, int anchoCm, int cantidad)
public void setOptionCristal(String opcionCristal)
public String getOptionCristal()
public String getMesa()
public String toString()

Clase MesaCafeMadera: Clase que hereda de la clase MesaCafe y que tiene las caracteristicas de una mesa de café sin cristal.

Métodos públicos:
public MesaCafeMadera(int idCliente, int alturaCm, int largoCm, int anchoCm, int cantidad)
public String toString()

Clase MesaCafeCristal: Clase que hereda de la clase MesaCafe y que contiene las caracteristicas de una mesa de café con cristal.

Métodos públicos:
public MesaCafeCristal(int idCliente, int alturaCm, int largoCm, int anchoCm, int cantidad)
public String toString()

Clase MesaDormitorio: Clase que hereda de la clase Mesa y que tiene las características de una mesa para el dormitorio con cajones.

Métodos públicos:
public MesaDormitorio(int idCliente, int alturaCm, int largoCm, int anchoCm, int numeroCajones, int cantidad)
public void setCajones(byte numeroCajones)
public int getCajones()
public String toString()

Clase MesaComedor: Clase que hereda de la clase Mesa y que se define como mesa de comedor.

Métodos públicos:
public MesaComedor(int idCliente, int alturaCm, int largoCm, int anchoCm, int cantidad)
public String toString()

Clase Silla: Clase abstracta que hereda de la clase Mueble y que define las características de una silla

Métodos públicos:
public Silla(int idCliente, int cantidad, boolean reposaBrazos)
public void setTipoSilla(String tipoSilla)
public String getTipoSilla()
public boolean getReposaBrazos()
public String toString()

Clase SillaOficina: Clase abstracta que hereda de la clase abstracta Silla y que se define como una silla de oficina.

Métodos públicos:
public SillaOficina(int idCliente, int cantidad, boolean reposaBrazos)
public abstract String ruedas();

Clase SillaConRuedas: Clase que hereda de la clase SillaOficina y que le da el atributo de tener ruedas.

Métodos públicos:
public SillaConRuedas(int idCliente,int cantidad, boolean reposaBrazos)
public String ruedas()
public String toString()

Clase SillaSinRuedas: Clase que hereda de la clase SillaOficina y que define una silla de oficina sin ruedas.

Métodos públicos:
public SillaSinRuedas(int idCliente, int cantidad, boolean reposaBrazos)
public String ruedas()
public String toString()

Clase SillaCocina: Clase que hereda de la clase Silla y que define una silla de cocina.

Métodos públicos:
public SillaCocina(int idCliente, int cantidad, boolean reposaBrazos)

Clase SillaPlegable: Clase que hereda de la clase Silla y que define una silla plegable.

Métodos públicos:
public SillaPlegable(int idCliente, int cantidad, boolean reposaBrazos)
public String getTipoSilla()
public String toString()

Clase Estado: La clase Estado se instancia dentro de la clase mueble y le da un estado de fabricación al mueble además de asignar al Artesano dicho mueble.

Métodos públicos:
public Estado()
public void setAsignado(boolean asignado)
public boolean getAsignado()
public void setAprobado(boolean aprobado)
public boolean getAprobado()
public int getIdArtesano()
public void setIdArtesano(int idArtesano)
public void setIdReg(int idArtesano)
public int getIdReg()
public void setFabricando()
public boolean getFabricando()
public void setIncompleto()
public boolean getIncompleto()
public void setFinalizado()
public boolean getFinalizado()
public String toString()

Clase Pedido: La clase pedido esta relacionada con todas las subclases de la clase Mueble y sirve para instanciar las clases de tipo mueble y contabilizar el número de muebles pedidos.

Métodos públicos:
public void Pedido()
public void sumaLosPedidosIniciales()
public Mueble addMesaDormitorio(int idCliente, int alturaCm, int largoCm, int anchoCm, int numeroCajones, int cantidad)
public Mueble addMesaComedor(int idCliente,int alturaCm, int largoCm, int anchoCm,int cantidad)
public Mueble addMesaCafeMadera(int idCliente,int alturaCm, int largoCm, int anchoCm,int cantidad)
public Mueble addMesaCafeCristal(int idCliente,int alturaCm, int largoCm, int anchoCm,int cantidad)
public Mueble addSillaCocina(int idCliente,int cantidad, boolean reposaBrazos)
public Mueble addSillaPlegable(int idCliente,int cantidad, boolean reposaBrazos)
public Mueble addSillaConRuedas(int idCliente,int cantidad, boolean reposaBrazos)
public Mueble addSillaSinRuedas(int idCliente,int cantidad, boolean reposaBrazos)
public String historialPedidos()

Clase ObtenerDato: Esta clase realiza la función de pedir los datos que el programa necesita en cada momento.

Métodos públicos:
ObtenerDato()
public String nombre()
public ArrayList<String> arrayComprueba()
public byte option()
public void setnom(String nom)
public String nombreRegistro()
public void borrarCliente(String nombre)
public int getId()
public int sueldo()
public int precio()
public int altura()
public int largo()
public int ancho()
public int cantidad()
public int cajones()
public boolean confirmacion()
public int pideId()
public int refePedido()
public String pideString()
public boolean reposabrazos()

Clase Ayuda: La clase ayuda es creada para facilitar la comprensión del programa y ofrece un soporte que da información dependiendo del lugar del programa que se solicite dicha ayuda.

Métodos públicos:
public Ayuda()
public void ayudaEmpleadoComercial()
public void ayudaEmpleadoJefeContratos()
public void ayudaEmpleadoJefe()
public void ayudaEmpleado()
public void ayudaClienteCuenta()
public void ayudaCliente()
public void ayudaGeneral()
public void ayudaBasica()

Clase Sistema: Clase que contiene todos los ArrayList que necesita el programa, ArrayList para Empleados, Clientes , Muebles y Notificaciones.

Métodos públicos:
public Sistema()
public ArrayList<Mueble> listarPedido(int id)
public ArrayList<Empleado> arrayArtesano()
public void borraMensaje(Notificacion mensaje)
public ArrayList<Notificacion> notifica(int idCliente)
public boolean existeCliente(int id)
public boolean removeArrayCliente(String nombre,int id)
public void addArrayPedido(Mueble pedido)
public ArrayList<Mueble> arrayPedido()
public void addArrayCliente(Cliente cliente)
public ArrayList<Cliente> cliente()
public void limparRegistros()
public ArrayList notificacion()
public ArrayList<Empleado> empleado()
public void borrarAsignacionArtesano(int id)
public void addEmpleado(Empleado empleado)

Clase Setup: se encarda de cargar datos al programa, empleados, clientes y pedidos.

Métodos públicos:
public Setup()
public void retardo(String caracter, int delay)
public void addPedidos(ArrayList<Mueble> arrayPedido)
public void addEmpleados(ArrayList<Empleado> arrayEmpleado)
public void addClientes(ArrayList<Cliente> arrayCliente)
public void addClientesComprueba(ArrayList<String> comprueba)
public void organizandoPedidos(ArrayList<Mueble> pedido)

Métodos destacados

Clase Comercial:

public Notificacion envioMensaje(int id, String asunto, String mensaje)
Método que instancia la clase Notificación pasando como argumento al constructor el ID del cliente, el asunto del mensaje y el cuerpo del mensaje, y devuelve el objeto creado.

public void presupuesto(ArrayList<Mueble> arrayPedido)
Método que busca pedidos sin precio y permite establecer un precio al pedido

public void verPedidoCompletado(ArrayList<Mueble> arrayPedido)
Método que permite visualizar los pedidos que contienen el estado de "Completado" y muestra el ID del cliente que realizo el pedido.

Clase Jefe:

public void GestionPedidos(ArrayList<Empleado> arrayEmpleado, ArrayList<Mueble> arrayPedido)
Método que muestra los pedidos confirmados y una lista de artesanos para realizar la asignación.

Clase Cliente:

public void confirmaPresupuesto(ArrayList<Mueble> confirma)
Método que muestra los pedidos con presupuesto y permite aceptar o rechazar el pedido

public void mensajeBuzon(ArrayList<Notificacion> notifica)
Método que permite listar todos los mensajes recibidos hasta el momento.

Clase ObterDatos:

public String nombreRegistro()
Método que pide el nombre del cliente que quiere realizar una cuenta y comprueba que el nombre no este ya en uso dentro del sistema, si el nombre esta usado le pedirá que ingrese un nuevo nombre.

public void borrarCliente(String nombre)
Cuando un cliente cancela su cuenta se usa este método para dejar libre dicho nombre en el sistema.

Clase Sistema:

ArrayList<Mueble> listarPedido(int id)
Devuelve una lista de pedidos de un cliente en concreto, pasando por argumento el id del cliente.

public ArrayList<Empleado> arrayArtesano()
Devuelve una lista de empleados que son únicamente artesanos.

public ArrayList<Notificacion> notifica(int idCliente)
Se pasa como parámetro el id del cliente y devuelve todos los mensajes del cliente.

public void borrarAsignacionArtesano(int id)
Se le pasa como parámetro el id del artesano que es despedido y se ponen a disposición del jefe todos los trabajos que tenia el artesano para poder hacer una nueva asignacion.

public void limparRegistros()
Si un cliente realiza un registro incorrecto, usuario null, es eliminado del registro.