APLICACIÓN MUSICAL EXPERIENCE

La aplicación Musical Experience es una aplicación Java en modo consola que nos permitirá gestionar una tienda. En este caso se ha elegido una tienda de música en la que se venderá en diversos formatos: CD, vinilo y casete. Podremos asignar nuestra venta a un determinado cliente, con lo que se implementara un sistema para eliminar/añadir/listar clientes, eliminar/añadir/listar música y vender música a determinado cliente; dicha venta será almacenada en un fichero de texto y también se podrá listar. El resto de entidades (clientes y música) serán almacenados en memoria , así que cuando termine la ejecución de la aplicación, desaparecerán.

Como toda aplicación de consola, se manejará usando parámetros y su uso se describe a continuación:

- Una vez iniciada la aplicación esta nos mostrará el siguiente mensaje: “Esperando instrucción”
- Las instrucciones posibles son:
    - **crear “cliente”|”música”|”venta”**
      En este caso estamos creando una entidad.El comando “crear” pedirá al usuario que rellenen los campos de cada entidad y al final mostrará un mensaje estilo “Cliente creado correctamente con código X”, dónde el código es un número incremental independiente para cada entidad. Cuando creemos una venta al tener que asignarla a un cliente nos pedirá el código de dicho cliente y para insertar música en esa venta también se hará mediante código (sólo se podrá asignar una música a una venta); que suceda esto se notificará al usuario de la siguiente manera: “Música X asignada a la venta actual”. Como hemos dicho anteriormente, solo las ventas se guardarán en un fichero.
    - **listar “cliente”|”música”|”venta”**
      Listará los elementos actuales almacenados en memoria.
    - **eliminar “cliente”|”música”|”venta [código]”**
      Eliminará el elemento que se pase por código. El código se podrá pasar como parámetro de la instrucción o no;si no se pasa, se deberá preguntar al usuario por él.
    - **Cerrar**
      Cuando la aplicación reciba esta instrucción finalizará su ejecución

En revisiones posteriores del proyecto ser realizaron las siguientes mejoras:

MEJORAS REALIZADAS:

1.-Hemos creado una clase Empleado heredada de Persona.

2.-Funciones para la Validación de datos e instrucciones de entrada .

3.-Creación de la clase Venta heredada de la clase Música que crea ficheros de texto con formato de recibo. En cada venta un mismo cliente puede incluir varios productos, por lo que para un mismo cliente tendremos un solo recibo por su compra y no varios recibos por producto.

4.-Generación de código de producto y código de empleado y cliente con información del mismo e incremental. Por ejemplo con el código del producto podremos saber inmediatamente el formato del producto y el número de producto dentro de dicho formato. Así mismo al cliente/empleado podremos identificarlo con su dni y sabremos qué número de cliente/empleado tiene. Si hubiésemos utilizado un simple contador no tendríamos esta información.

5.-Mejora de la función para listar clientes, productos y ventas ahora no sólo podremos listar todos los clientes, productos y ventas de la tienda sino que podremos buscar los datos de un cliente o un producto en concreto a través de su código o de su nombre de archivo cuando se trate de una venta.

6.-Creación de la posibilidad de listar todos los empleados o uno concreto a través del código de pantalla para mostrar los datos por consola.

7.-Implementación de las funciones para borrar todos los clientes, empleados, música y ventas de la tienda.

8.-Implementación de la función eliminar un cliente a través de su código.

9.-Implementación de la función recursiva DeseaRepetir() para no tener que introducir de forma repetida la instrucción por consola cuando queremos repetirla.

10.-Implementación de la función para mostrar las posibles instrucciones que el usuario puede utilizar en la aplicación.

11.-Implementación de la función para obtener el empleado que más ha facturado de la tienda.

GLOSARIO DE FUNCIONES:

`>>public static int CrearCodigoPersona(int a , int b)`: Esta función sirve para generar el código del empleado y del cliente de forma automática. Dicho código será la concatenación de dos enteros: el número del dni + el número de empleado o cliente. La numeración de empleados y clientes se hace de forma independiente.

- Se utiliza cuando creamos un cliente o un empleados→funciones CrearCliente() y CrearEmpleado()

`>>public static String CrearCodigoMusica(int a , String sB)`: Esta función sirve para generar el código del producto de forma automática. Dicho código es la concatenación de un entero y un String (que dependerá del formato del mismo):

el número de producto + "000" | "001" | "002", donde:

- "000"→Se utiliza para formato vinilo
- "001"→Se utiliza para formato casette
- "002"→Se utiliza para formato cd

Dicho código es la concatenación de un String y un entero y devuelve un String, esto es necesario porque si pasaramos el String "000" a entero, la conversión seria "0". Además con la creación del primer producto de un tipo de formato, por ejemplo cd, nuestro código sería "0002", al pasarlo a String el código de formato "002"→ "2", después lo concatenaríamos con el número del producto que es "0"→ "02" y al volverlo a pasar a entero nos quedaría "2" lo que realmente puede llevar a conflicto y no nos da ninguna información del producto.

- Se utiliza cuando creamos un producto→ función CrearMusica()

`>>public static int ValidarEntero()`: Esta función sirve para controlar la entrada de datos de usuario. Se utiliza para detectar cuando el usuario no ha introducido un entero y se ejecuta hasta que éste lo introduce.

- Se utiliza en la creación de clientes y empleadospara validar el número de dni, el código postal y el número de teléfono→ función CrearCliente() y CrearEmpleado()
- Se utiliza en la creación de productos para validar el año de publicación, la duración y el número de copias→función CrearMusica()

`>>public static float ValidarDecimal()`: Esta función sirve para controlar la entrada de datos de usuario. Se utiliza para detectar cuando el usuario no ha introducido un decimal y se ejecuta hasta que éste lo introduce.

- Se utiliza en la creación de empleados para validar el sueldo→función CrearEmpleado()
- Se utiliza en la creación de productos para validar el precio→función CrearMusica()

`>>public static String ValidarLetra()`: Esta función sirve para controlar la entrada de datos de usuario. Se utiliza para detectar cuando el usuario no ha introducido un carácter o letra y se ejecuta hasta que éste lo introduce.

- Se utiliza en la creación de empleados y clientes para validar la letra del dni→función CrearCliente() y CrearEmpleado()

`>>public static String ValidarFecha()`: Esta función se utiliza para validar la fecha de nacimiento de un cliente o un empleado. Gestiona que sólo podamos introducir fechas válidas:

- Analiza que no se introduzcan días o meses que no existen.
- También analiza si el año es bisiesto o no para poder habilitar el día 29 de febrero.
- Analiza que la persona que introduce su fecha de nacimiento no pueda tener más de 114 años.
- Valida la fecha en un formato determinado dd/mm/aaaa.
- Gestiona que no se introduzca ningún carácter especial por equivocación.

En caso de que la fecha introducida no sea válida la función se ejecuta hasta que el usuario introduce una correcta.

- Se utiliza en la creación de clientes y empleados para validar su fecha de nacimiento→funciones CrearCliente() y CrearEmpleado()

`>>public static boolean DetectarAnyoBisiesto(int anyo)`: Esta función se utiliza para detector si el año que se pasa como argumento es o no un año bisiesto para poder habilitar el día 29 de Febrero. Si es bisiesto devuelve un valor true y si no un false.

- Se utiliza en la validación de la fecha de nacimiento de un cliente o un empleado→ función ValidarFecha()

`>>public static String ValidarFormatoProducto()`: Esta función sirve para validar que el formato que introduce el usuario es uno de los permitidos: vinilo, casette o cd. En caso de que no se introduzca uno de estos,la función se ejecutará hasta que el usuario introduzca un formato válido.

- Se utiliza en la creación de productos para validar el formato→función CrearMusica()

`>>public static String DeseaRepetir()`: Esta función permite al usuario realizar de forma recursiva una instrucción para no tener que volver a introducirla. Por ejemplo tras crear un cliente, nos preguntara si deseamos crear otro y el usuario podrá decidir si quiere crearlo o si desea salir al menú de instrucciones principal.

- Se utiliza en cada una de las instrucciones implementadas en el programa

`>>public static String getExtension(String filename)`: Esta función se utiliza para obtener la extensión de un archivo. La función busca la posición en la que se encuentra el carácter '.' y extrae la subcadena del nombre del archivo desde esa posición hasta el final de la cadena. Esta subcadena se correspondería con el formato del archivo y es lo que devuelve la función.

- Se utiliza para poder crear la lista de archivos de tipo texto de un directorio→función ObtenerListaVentas()

`>>public static String ValidarCorreo()`: Esta función sirve para validar que el correo introducido por el usuario realmente se corresponda con el formato de una dirección de correo electrónico. Para ello se comprueba que la cadena introducida contenga dos caracteres especiales "@ " y ".". Si el usuario no introduce correctamente el correo la función se ejecutará hasta que el correo introducido sea válido.

- Se utiliza cuando creamos un cliente o un empleado →funciones CrearCliente() y CrearEmpleado()

`>>public static ArrayList<String> ObtenerListaVentas()`: Esta función se utiliza para listar todos los archivos de texto que se crean cuando realizamos una venta a modo de recibo y que se encuentran en el directorio por defecto de la aplicación. La función se asegura que la extensión de archivo sea txt y que vayan precedidos por la palabra "venta".

- Se utiliza cuando queremos mostrar todas las ventas realizadas o una en concreto→función ListarFicherosVentas()
- Cuando queremos validar que una venta existe→función ValidarVenta()
- Cuando queremos eliminar todos los archivos de texto asociados a las ventas→ EliminarTodasVentas()

`>>public static int NumeroUltimaVenta()`: Esta función se utiliza para obtener el número del ultimo archivo de texto asociado a la última venta. Los archivos a modo de recibo tendrán como nombre: ventax.txt, esta función lo que hace es que extrae el valor "x" que va dentras de la palabra "venta" y antes de formato ".txt" que indicará el número de venta que es.

- Se utiliza cuando creamos una nueva venta para saber que índice tenemos→función RellenarDatosVenta()

`>>public static void CrearCliente(Tienda tienda)`: Esta función se utiliza para crear un cliente con sus datos y añadirlo a la lista de clientes de la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void CrearEmpleado(Tienda tienda)`: Esta función se utiliza para crear un empleado con sus datos y añadirlo a la lista de empleados de la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void CrearMusica(Tienda tienda)`: Esta función se utiliza para crear un producto con sus datos y añadirlo al catálogo de la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void CrearVenta(Tienda tienda)`: Esta función se utiliza para crear una venta (que podrá estar formada por varios productos del catálogo) que generará un archivo de texto. En dicho archivo de texto que tendrá formato de recibo tendremos:

- Número de venta de la tienda.
- Fecha y Hora en la que se realizó la venta.
- Nombre y código del empleado que atendió la venta.
- Nombre y código del cliente que realizó la compra.
- Productos que constituyen la venta con sus características y precios.
- Importe total de la venta a abonar por el cliente.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static Empleado ValidarEmpleadoVenta(Tienda tienda)`: Se utiliza para garantizar que el empleado asociado al código que introduce el usuario está registrado en la lista de empleados de la tienda. Esta función devuelve el empleado asociado al código o un null en caso contrario.

- Se utiliza cuando creamos una venta para garantizar que el empleado que atiende la venta está registrado.→función CrearVenta()

`>>public static Cliente ValidarClienteVenta(Tienda tienda)`: Se utiliza para garantizar que el cliente asociado al código que introduce el usuario está registrado en la lista de clientes de la tienda. Esta función devuelve el cliente asociado al código o un null en caso contrario.

- Se utiliza cuando creamos una venta para garantizar que el cliente que realiza la compra está registrado.→función CrearVenta()

`>>public static Musica ValidarMusicaVenta(Tienda tienda)`: Se utiliza para garantizar que el producto asociado al código que introduce el usuario está registrado en el catálogo de la tienda. Esta función devuelve el producto asociado al código o un null en caso contrario.

- Se utiliza cuando creamos una venta para garantizar que el producto que es objeto de la venta está registrado, en cuyo caso decrementará el número de copias de ese producto en la lista de música de la tienda y lo devolverá. Si se diera el caso de que

fuera la última copia del producto, se eliminaría el producto del catálogo→función CrearVenta()

`>>public static boolean ValidarVenta(String nombrearchivo)`: Se utiliza para garantizar que el fichero de texto asociado a la venta que introduce el usuario está en el directorio por defecto de la aplicación. Esta función devuelve un true si existe dicho fichero o un false en caso contrario.

- Se utiliza cuando vamos a mostrar un determinado fichero de texto por consola para garantizar que existe→función ListarFicherosVentas()
- Se utiliza cuando vamos a eliminar un determinado fichero de texto para garantizar que existe→función EliminarVenta()

`>>public static Venta RellenarDatosVenta(Empleado empleado, Cliente cliente, Musica musica)`: Esta función se utiliza para rellenar todos los datos referentes a la compra. Devuelve el objeto venta con todas sus características.

- Se utiliza cuando creamos una venta→función CrearVenta()

`>>public static void RegistrarVenta(Venta venta, RWFichero fichero, boolean cabecera, boolean fin)`: Esta función se encarga de escribir en un fichero de texto el recibo correspondiente a la venta.

- Se utiliza cuando creamos una venta→función CrearVenta()

`>>public static void ListarPersona(Tienda tienda, String tipo, String opcion)`: Esta función se encarga de mostrar por consola los datos referentes a clientes y empleados. Nos permite mostrar o bien un cliente/empleado concreto introduciendo el código de cliente/empleado o todos los clientes/empleados registrados en la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void ListarMusica(Tienda tienda, String opcion)`: Esta función se encarga de mostrar por consola los datos referentes los productos registrados en el catálogo de la tienda. Nos permite mostrar o bien un producto concreto introduciendo el código del producto o todos los productos del catálogo.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void ListarFicherosVentas(Tienda tienda, String opcion)`: Esta función se encarga de mostrar por consola los ficheros de texto asociamos a cada venta a modo de recibo. Nos permite mostrar o bien archivo de texto concreto introduciendo el nombre del archivo o todos los archivos de texto de la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static int BuscarCliente(int codigocliente, Tienda tienda )`: Esta función se utiliza para buscar el cliente que coincida con el código introducido por el usuario dentro de la lista de clientes de la tienda. La función devolverá un -1 si no encontró al cliente o la posición del cliente dentro de la lista de clientes de la tienda.

- Se utiliza cuando queremos mostrar por pantalla un cliente concreto→función ListarPersona()
- Se utiliza cuando queremos eliminar un cliente concreto→función EliminarCliente()

`>>public static int BuscarEmpleado(int codigoempleado, Tienda tienda )`: Esta función se utiliza para buscar el empleado que coincida con el código introducido por el usuario dentro de la lista de empleados de la tienda. La función devolverá un -1 si no encontró al empleado o la posición del empleado dentro de la lista de empleados de la tienda.

- Se utiliza cuando queremos mostrar por pantalla un empleado concreto→función ListarPersona()
- Se utiliza cuando queremos eliminar un empleado concreto→función EliminarEmpleado ()

`>>public static int BuscarMusica(String codigomusica, Tienda tienda )`: Esta función se utiliza para buscar el producto que coincida con el código introducido por el usuario dentro del catálogo de la tienda. La función devolverá un -1 si no encontró el producto o la posición del producto dentro del catálogo de la tienda.

- Se utiliza cuando queremos mostrar por pantalla un producto concreto→funcion ListarMusica()
- Se utiliza cuando queremos eliminar un producto concreto→función EliminarMusica()

`>>public static void MostrarCliente(Cliente cliente)`: Esta función muestra por consola los datos referentes al objeto cliente que se le pasa como argumento.

- Se utiliza cuando listamos clientes→función ListarPersona()

`>>public static void MostrarMusica(Musica musica)`: Esta función muestra por consola los datos referentes al objeto musica que se le pasa como argumento.

- Se utiliza cuando listamos productos→función ListarMusica()

`>>public static void MostrarEmpleado(Empleado empleado)`: Esta función muestra por consola los datos referentes al objeto empleado que se le pasa como argumento.

- Se utiliza cuando listamos empleados→función ListarPersona()
- Se utiliza para mostrar los datos del empleado que más ventas facturó→función BuscarEmpleadoMasVentas()

`>>public static void MostrarFicheroVenta(RWFichero fichero)`: Esta función se utiliza para leer los archivos de texto generados al realizar una venta.

- Se utiliza cuando listamos los archivos de ventas→función ListarFicherosVentas()

`>>public static void MostrarInstrucciones()`: Esta función se utiliza para mostrar por consola las posibles instrucciones que el usuario puede introducir.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarCliente(Tienda tienda)`: La función se utiliza para eliminar un cliente concreto de la lista de clientes de la tienda a través de su código.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarEmpleado(Tienda tienda)`: La función se utiliza para eliminar un empleado concreto de la lista de empleados de la tienda a través de su código.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarMusica(Tienda tienda)`: La función se utiliza para eliminar un producto concreto del catálogo de la tienda a través de su código.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarVenta(Tienda tienda)`: La función se utiliza para borrar un archivo de texto asociado a una venta a través del nombre del archivo (sin el formato), que introduce el usuario. La función se asegura de que el archivo existe previamente y de no ser así se le comunica al usuario.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarArchivoTexto(String nombrearchivo)`: La función se utiliza para borrar un archivo de texto asociado a una venta a través del nombre del archivo (sin el formato) gestionando las posibles excepciones que puedan surgir.

- Se utiliza cuando vamos a borrar un archivo de texto concreto→ función EliminarVenta()

`>>public static void EliminarTodasVentas(Tienda tienda)`: La función se utiliza para borrar todos los archivos de texto generados al realizar las ventas del directorio por defecto de la aplicación.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarTodasPersonas(String opcion, Tienda tienda)` : La función se utiliza para borrar toda la lista de empleados o de clientes de la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void EliminarTodosProductos(Tienda tienda)` : La función se utiliza para borrar todo el catálogo de la tienda.

- Se utiliza en la función principal de la aplicación→ función main()

`>>public static void BuscarEmpleadoMasVentas(Tienda tienda)` : La función se utiliza para saber que empleado realizo facturó más dinero en ventas y mostrar sus datos por consola.

- Se utiliza en la función principal de la aplicación→ función main()