Prueba - Sistema de Clientes

En esta prueba validaremos nuestros conocimientos vistos en el módulo.

Lee todo el documento antes de comenzar el desarrollo individual, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Descripción

Durante el último tiempo han ocurrido cambios significativos en nuestras vidas a causa de la tecnología, los pequeños locales han debido adaptarse a esta nueva necesidad y comenzar a mudar sus negocios hacia lo electrónico. Es por ello, que la pastelería “Bon Bon Jovi” le ha contactado a usted para crear un software, donde pueda tener un control de sus clientes, evitando así que vayan a su local rival “Dominic Completto”. Para esto, debemos considerar las siguientes funcionalidades:

Menú

Para poder navegar entre las distintas funciones de la aplicación, deberá existir un menú inicial. El menú permitirá listar, agregar, editar, cargar, exportar datos de clientes y salir de la sección.

```
1. Listar Clientes
2. Agregar Cliente
3. Editar Cliente
4. Cargar Datos
5. Exportar Datos
6. Salir
Ingrese una opción:
```

Agregar Cliente

Se debe permitir agregar clientes nuevos e incorporarlos en el listado. Por defecto, deberá venir la opción ”Activo” en la categoría Cliente, cuando este sea nuevo.

```
------------Crear Cliente------------

Ingresa RUN del Cliente:


Ingresa Nombre del Cliente:


Ingresa Apellido del Cliente:


Ingresa años como Cliente:

---------------------------------------
```

Listar Cliente

Se deben listar a todos los clientes existentes del negocio con su respectiva categoría. En el ejemplo, se muestra la lista para 2 clientes.

```
------------Datos del Cliente-------------

RUN del Cliente: 12.123.412-2
Nombre del Cliente: Nicolas
Apellido del Cliente: Cake
Años como Cliente: 7 años
Categoría del Cliente: Activo

-----------------------------------------
------------Datos del Cliente-------------

RUN del Cliente: 25.673.022-2
Nombre del Cliente: Taylor
Apellido del Cliente: Shift S.
Años como Cliente: 1 día
Categoría del Cliente: Activo

-----------------------------------------
```

Editar Cliente

Se podrá editar un cliente en caso de tener algún error en sus datos. Para esto, se deberá ingresar la opción de lo que se busca hacer.

```
-------------Editar Cliente-------------
Seleccione qué desea hacer:
1.-Cambiar el estado del Cliente
2.-Editar los datos ingresados del Cliente

Ingrese opcion:
----------------------------------------

Ingrese RUN del Cliente a editar:
```

En este caso, se buscará cambiar el estado del cliente.

```
Ingrese RUN del Cliente a editar:

-----Actualizando estado del Cliente----
El estado actual es: Activo
1.-Si desea cambiar el estado del Cliente a Inactivo
2.-Si desea mantener el estado del cliente Activo

Ingrese opcion:
----------------------------------------
```

Mientras que en este caso, se quiere cambiar los datos ingresados en un cliente, mediante la búsqueda por el RUN

```
Ingrese RUN del Cliente a editar:

----Actualizando datos del Cliente-----

1.-El RUN del Cliente es: 12.123.412-2
2.-El Nombre del Cliente es: Nicolas
3.-El Apellido del Cliente es:Cake
4.-Los años como Cliente son: 7 años

Ingrese opcion a editar de los datos del cliente:
----------------------------------------
```

Finalmente y en base a la elección del dato a editar, la pantalla arrojará como ejemplo lo siguiente.

```
Ingrese opcion a editar de los datos del cliente:
1
----------------------------------------
1.-Ingrese nuevo RUN del Cliente:
24.157.163-K
----------------------------------------
Datos cambiados con éxito
```

Cargar Datos

Mediante la selección del menú se debe cargar información de los clientes previos del negocio desde un archivo llamado "**DBClientes.csv"**.

```
---------Cargar Datos en Linux o Mac-----------

Ingresa la ruta en donde se encuentra el archivo DBClientes.csv:

home/usuario/Desktop

-----------------------------------------------

Datos cargados correctamente en la lista
```

```
---------Cargar Datos en Windows---------------

Ingresa la ruta en donde se encuentra el archivo DBClientes.csv:

C:\\usuario\equipo\Desktop

-----------------------------------------------

Datos cargados correctamente en la lista
```

El archivo "DBClientes.csv" contiene este formato.

```
17.162.856-5,Iron,Manjar,1 año,Activo
10.513.821-6,Thor,Tita ,4 años,Activo
17.151.677-8,Capitán,Marraqueta,9 años,Inactivo
```

Exportar datos

Mediante la selección del menú se debe exportar información de los clientes y sus respectivas categorías. El archivo debe ser "**clientes.txt**" o "**clientes.csv**" dependiendo del formato elegido.

```
---------Exportar Datos-----------
Seleccione el formato a exportar:
1.-Formato csv
2.-Formato txt

Ingrese una opción para exportar:
----------------------------------
```

Dependiendo del formato y del equipo, las rutas quedarán así:

```
---------Exportar Datos en Linux o Mac-----------
Ingresa la ruta en donde desea exportar el archivo clientes.csv:
home/usuario/Desktop
-------------------------------------------------
Datos de clientes exportados correctamente en formato csv.
```

```
---------Exportar Datos en Windows---------------
Ingresa la ruta en donde desea exportar el archivo clientes.txt:
C:\\usuario\equipo\Desktop
-------------------------------------------------
Datos de clientes exportados correctamente en formato txt.
```

Requerimientos

1. Crear proyecto en Java.
   (1 Punto)

   1.1. Crear un proyecto nuevo a través de Eclipse como proyecto Java.
   `Nuevo Proyecto → Proyecto Java → Ingresar nombre proyecto → Finalizar`

   1.2. Crear 6 paquetes sobre los cuales trabajaremos:
   - Package main
   - Package modelo
   - Package servicio
   - Package test
   - Package utilidades
   - Package vista

   1.3. Crear el enum CategoríaEnum en el package Modelo, la cual contendrá 2 posibles valores: Activo e Inactivo.

   1.4. Crear la clase Cliente en package modelo, con los siguientes requisitos:
   - String runCliente
   - String nombreCliente
   - String apellidoCliente
   - String aniosCliente (se puede cambiar a int si lo desea)
   - CategoriaEnum nombreCategoria
   - Generar el constructor con parámetros
   - Generar los getter y setter correspondientes
   - Generar el toString para los parámetros

   1.5. Crear la clase Main con método public static void en el package main, la cual usará una instancia para iniciar el menú.

2. Utilizar ciclos y métodos.
(2 Puntos)

2.1. Crear clase Menu en el package vistas, que debe contener los siguientes atributos:
- clienteServicio, instancia de ClienteServicio.
- archivoServicio, instancia de ArchivoServicio.
- exportadorCsv, instancia de ExportarCsv.
- exportarTxt, instancia de ExportarTxt.
- Definir un String fileName = “Clientes” (para exportar el archivo)
- Definir un String fileName1 = “DBClientes.csv” (para importar el archivo)
- scanner, instancia de Scanner para recibir valores a través del teclado.
- iniciarMenu, muestra el menu principal y recibe la entrada del teclado a través del scanner. Contiene la lógica para denotar los demás métodos en base a la entrada del teclado.

2.2. La clase Menu debe contener los siguientes métodos para la construcción y selección del menu:
- listarCliente.
- agregarCliente.
- editarCliente.
- importarDatos.
- exportarDatos.
- terminarPrograma.

2.3. Se deben sobreescribir los métodos nombrados previamente en el punto 7, dentro de la misma clase Menú de la siguiente manera:
- listarClientes, muestra lista de clientes agregados, ya sea por importación o agregando a mano
- agregarCliente, solicita ingreso de datos y llena objeto de tipo Cliente.
- editarCliente, permite la edición de algún cliente en caso de requerirlo o cambiar el estado del cliente.
- cargarDatos, ejecuta la carga de datos del archivo “DBClientes.csv”.
- exportarDatos, llama a método para exportar clientes en formato “.txt” o “.csv”.
- terminarPrograma, el cual finaliza la ejecución del sistema.

Hint: El único método que no se debe sobreescribir es iniciarMenu, ya que contiene su implementación al inicio de la clase para formar el menú.

3. Utilizar arreglos y archivos
(2 Puntos)

3.1. Crear la clase ClienteServicio en el package servicio con los siguientes requisitos:
- Crear un atributo llamado List<Cliente> listaClientes
- Generar un constructor de ClienteServicio que tenga esta listaClientes como una nueva ArrayList.
- Generar el método público sin valor de retornolistarClientes e implementar el ciclo más idóneo para recorrer cada uno de los clientes.
- Generar un public void del método agregarCliente y pasarle los parámetros de la clase Cliente. Utiliza este método para guardar clientes en una instancia de cliente.
- Generar un public void del método editarCliente y pasarle los parámetros de la clase Cliente.
- Crear un getter de listaCliente y que pueda retornar una listaClientes.

4. Aplicar Polimorfismo y Herencia según el Paradigma Orientado a Objetos (POO)
(2 Puntos)

4.1. Crear una clase abstracta de nombre Exportador en package Servicio, que contenga un método abstracto para exportar, cuyos parámetros serán String fileName y List<Cliente> listaClientes.

4.2. Crear una clase ExportadorCsv en el package servicio, que contenga un método exportar, cuyos parámetros serán String fileName y una List<Cliente> listaClientes. Se deben realizar las implementaciones correspondientes al interior del método usando PrintWriter y Filewriter para la exportación de archivos.

4.3. Crear una clase ExportadorTxt en el package servicio, que contenga un método exportar, cuyos parámetros serán String fileName y una List<Cliente> listaClientes. Se deben realizar las implementaciones correspondientes al interior del método usando PrintWriter y Filewriter para exportación de archivos.

Hint: Los pasos 8, 9 y 10 buscan tener una clase abstracta (Exportador), con un solo método exportar, cuyas clases ExportadorCsv y ExportadorTxt extienden de esta clase e implementar su método. Por lo tanto, los métodos declarados serán pertenecientes a cada Exportador bajo el concepto de herencia. Al utilizarlo, se instancia alguno de los exportadores en la clase menu ocupando polimorfismo.

5. Aplicar instancias al Paradigma Orientado a Objetos (POO)
(2 Puntos)

5.1. Crear la clase ArchivoServicio en el package servicio que extiende a la clase Exportador. Esta contiene los siguientes requisitos:

- Crear el método cargarDatos que recibe por parámetro un String fileName, el cual indica el nombre del archivo a cargar. Se deben realizar las implementaciones correspondientes al interior del método usando FileReader y BufferedReader (para lectura de archivos).
- Crear el método exportar que será una herencia proveniente de la clase Exportador, cuyos parámetros serán los mismos que se van a implementar en el paso 8.

5.2. Crear una clase Utilidad en package utilidades, que contenga métodos reutilizables para el menú como limpiar pantalla, mostrar mensajes, etc.

6. Aplicar pruebas unitarias
(1 Punto)

6.1. Añadir dependencias para pruebas Unitarias, ver ejemplo en apartado Anexos al final del documento.

6.2. Escribir pruebas unitarias para ClienteServicio.

- Método agrearClienteTest para verificar el funcionamiento de agregarCliente (se debe agregar un cliente para que el test corra de manera correcta)
- Método agrearClienteNullTest para verificar el funcionamiento de agregarCliente en caso que vengan casos nulos (se debe agregar un cliente nulo para que el test corra de manera correcta).

 ¡Mucho éxito!

Consideraciones y recomendaciones

- Una vez terminado el desafío, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el .zip en el LMS.

- Se puede optar por los tipos de datos con los que se sienta más cómodo.
- Crear los métodos que se estimen convenientes, además de los expresados en los requerimientos.
- Crear la clase Main que contenga método main para iniciar el programa. Debe iniciar con el menú.
- Se deben utilizar iteraciones de la librería Streams.
- Utilizar biblioteca JUnit 4 para realizar testing (implementación detallada en anexos).

Para poder aplicar y correr los test solicitados, debemos agregar las bibliotecas correspondientes de "JUnit". Para ello debemos seguir los siguientes pasos:

1. Nos posicionamos sobre el proyecto creado entre los package (Que en el ejemplo se llama "PruebaIntento1") > Hacemos click derecho, buscamos "propiedades y le damos click nuevamente. Debería desplegar la siguiente pestaña.

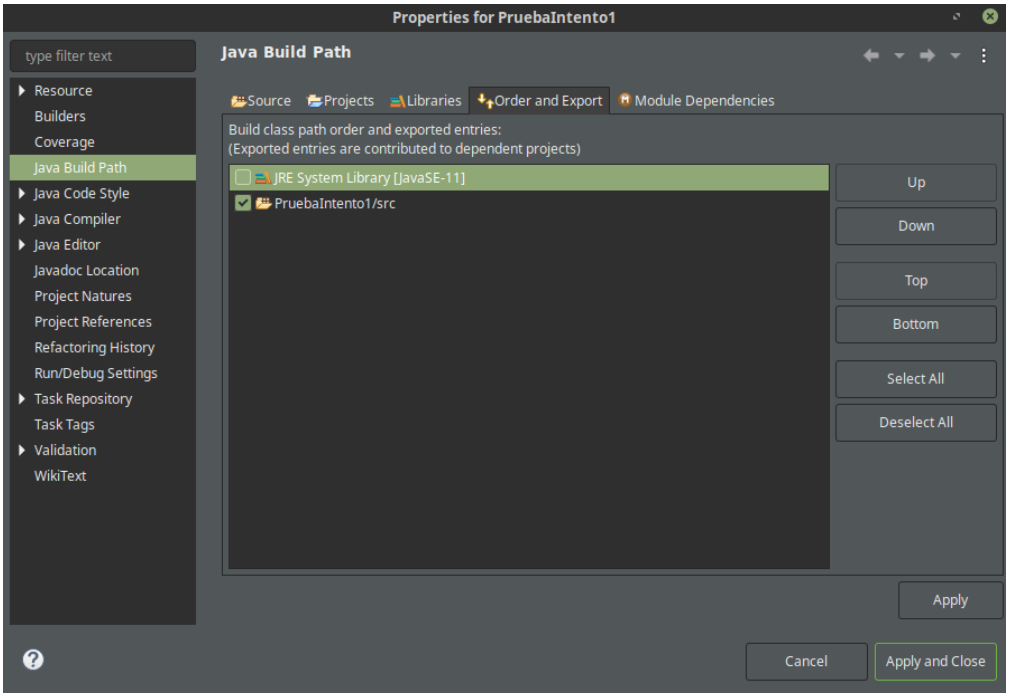

Imagen 1: Pestaña de apertura propiedades
Fuente: Desafío Latam

2. Hacemos click sobre “Java Build Path”, buscamos la pestaña “Libraries” (bibliotecas) como se muestra en la imagen y luego colocamos “Add Library” manteniendo la opción “Module Path” marcada.

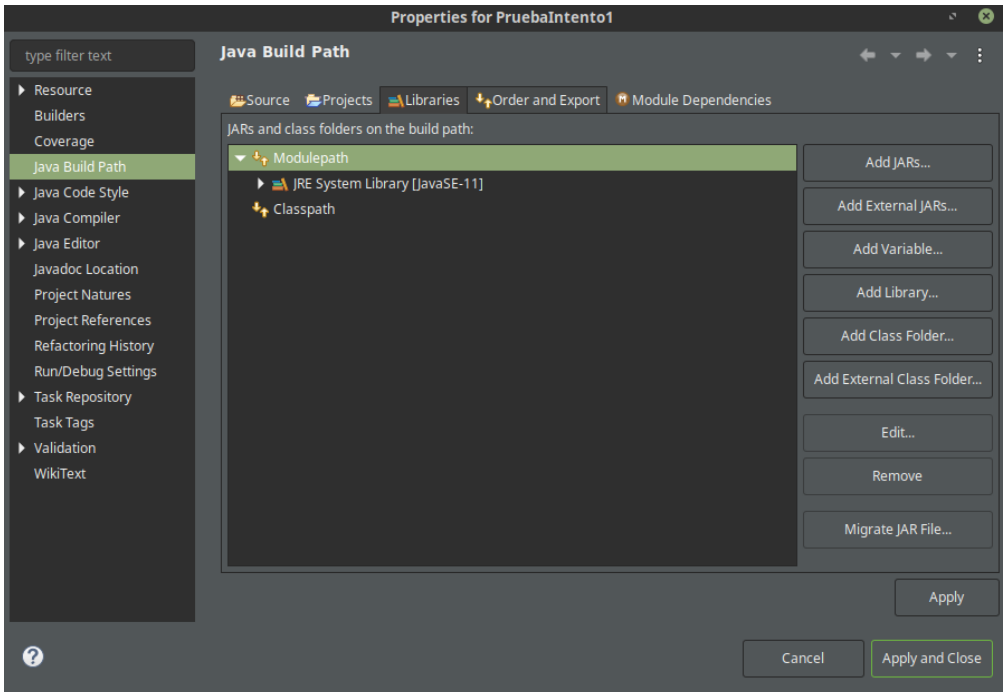

Imagen 2: Selección de ModulePath pestaña propiedades
Fuente: Desafío Latam

3. Cuando hacemos click sobre “Add library” se despliega la siguiente pestaña.

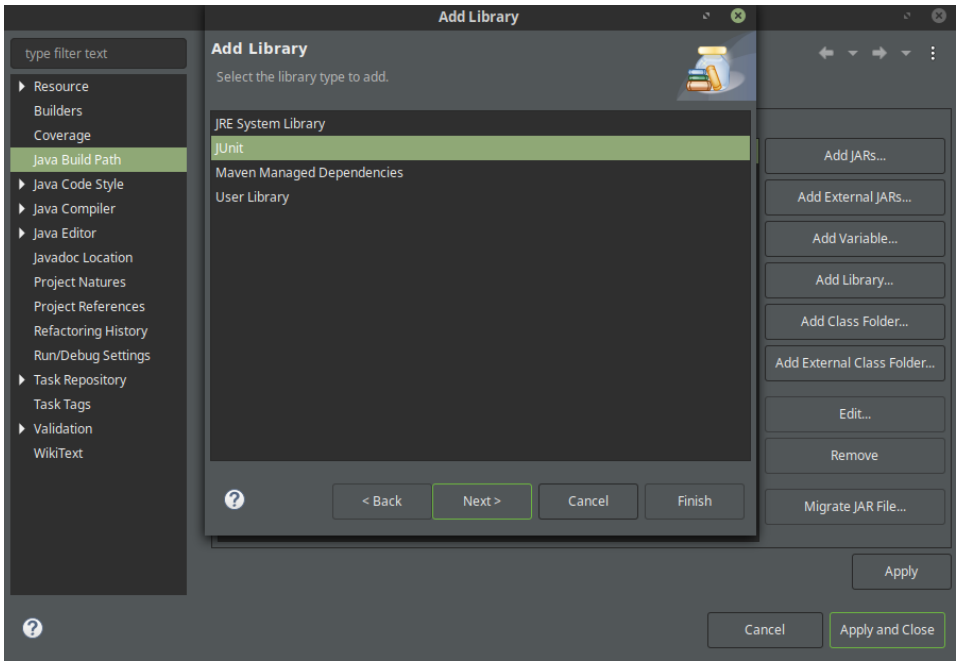

Imagen 3: Selección JUnit en la pestaña Add Library
Fuente: Desafío Latam

4. Damos click a next, seleccionamos la biblioteca “JUnit 4” y le damos a finish.

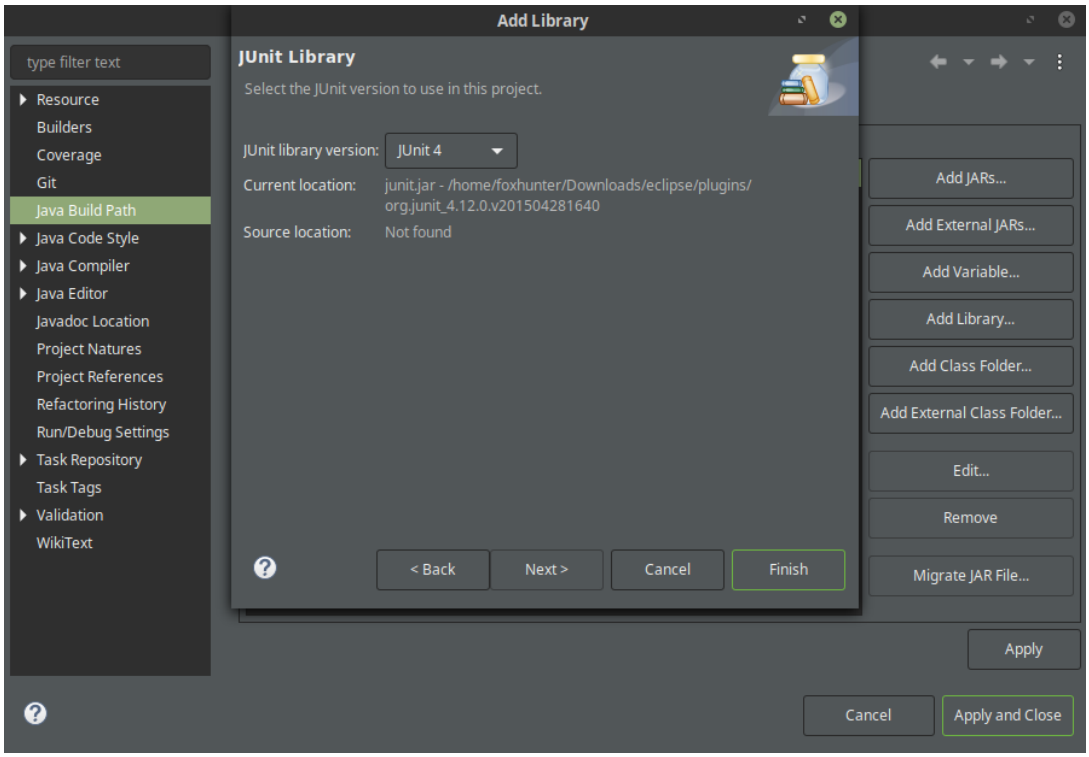

Imagen 4: Selección JUnit 4 en la pestaña Add Library
Fuente: Desafío Latam

5. Luego le damos click en “Apply and Close”.