UNIVERSIDAD DE SONORA

Unidad Regional Centro

División de Ingeniería

Ingeniería en Sistemas de Información

Desarrollo de Sistemas 3

Maestro: Navarro Hernández Rene Francisco

Grupo 1 – Hora 11 Am – 12 Pm- Edif. 5G – Laboratório 205

Nombres integrantes del equipo:

- **Salcido Gutiérrez, Daniel Antonio**
- **Osornio Montaño Víctor Manuel**
- **Vallejo Leyva Marcos**

Archivos de acceso aleatorio

Vallejo Leyva Marcos, **Salcido Gutiérrez, Daniel Antonio, Osornio Montaño Víctor Manuel**

Puntos de Salcido Gutiérrez, Daniel Antonio:

Aquí están los fragmentos del código que corresponden a los puntos 5 mencionados en la tarea, mas aparte un método para ver todos los vendedores:

- 5. Implementar una consulta por zona a la que está asignado el vendedor. Se deben mostrar todos los datos de todos los vendedores que corresponden a esa zona.

```
private static void consultarPorZona(Scanner scanner) {
        System.out.print("Ingrese la zona a consultar: ");
        String zonaConsulta = scanner.nextLine();

        List<Vendedor> vendedoresEnZona =
vendedoresPorZona.getOrDefault(zonaConsulta, new ArrayList<>());

        if (!vendedoresEnZona.isEmpty()) {
            System.out.println("Vendedores en la zona '" +
zonaConsulta + "':");
            vendedoresEnZona.forEach(v ->
System.out.println(v.toString()));
        } else {
            System.out.println("No hay vendedores en la zona
especificada.");
        }
    }

    private static void verTodosLosVendedores() {
        if (!vendedores.isEmpty()) {
            System.out.println("Todos los vendedores en el archivo
.dat:");
            vendedores.forEach(v ->
System.out.println(v.toString()));
        } else {
            System.out.println("No hay vendedores registrados.");
        }
    }
}
```

Como se realizó:

- **Implementar una consulta por zona a la que está asignado el vendedor:**
  - El método **consultarPorZona(Scanner scanner)** permite al usuario ingresar una zona.
  - Se busca en el mapa **vendedoresPorZona** todos los vendedores que pertenecen a la zona especificada y se muestran todos los datos de estos vendedores.
  - Si no se encuentra ningún vendedor en la zona, se muestra un mensaje indicando que no hay vendedores registrados en esa zona.
- **Método para ver todos los vendedores:**
  - El método **verTodosLosVendedores()** muestra en pantalla todos los datos de todos los vendedores almacenados en la lista de **vendedores**.

- Si no hay vendedores registrados, se muestra un mensaje indicando que no hay vendedores registrados.

Puntos de Osornio Montaño Víctor Manuel:

Aquí están los fragmentos del código que corresponden a los puntos 2 y 4 mencionados en la tarea:

- 2. Incluir la opción para agregar un nuevo vendedor al archivo. Se deben solicitar al usuario cada uno de los campos del nuevo vendedor y guardar los datos en el archivo.

```
private static void agregarVendedor(Scanner scanner) {
    System.out.print("Ingrese el código del vendedor: ");
    int codigo = scanner.nextInt();
    scanner.nextLine(); // Consumir la nueva línea

    if (vendedores.stream().anyMatch(v -> v.getCodigo() == codigo)) {
        System.out.println("El código de vendedor ya existe. Por favor,
elija otro código.");
        return;
    }

    System.out.print("Ingrese el nombre del vendedor: ");
    String nombre = scanner.nextLine();
    System.out.print("Ingrese la fecha de nacimiento del vendedor
(dd/mm/yy): ");
    String fechaNacimiento = scanner.nextLine();
    System.out.print("Ingrese la zona del vendedor: ");
    String zona = scanner.nextLine();
    System.out.print("Ingrese el total de ventas mensuales del vendedor:
");
    double ventasMensuales = scanner.nextDouble();

    Vendedor nuevoVendedor = new Vendedor(codigo, nombre,
fechaNacimiento, zona, ventasMensuales);
    vendedores.add(nuevoVendedor);
    vendedoresPorZona.computeIfAbsent(zona, k -> new
ArrayList<>()).add(nuevoVendedor);
    System.out.println("Vendedor agregado exitosamente.");
}
```

➢ **Incluir la opción para agregar un nuevo vendedor al archivo:**

- El método **agregarVendedor(Scanner scanner)** se encarga de solicitar al usuario todos los datos necesarios para crear un nuevo vendedor, incluyendo código, nombre, fecha de nacimiento, zona y ventas mensuales.
- Luego, se crea un nuevo objeto **Vendedor** con los datos proporcionados y se agrega tanto a la lista de **vendedores** como al mapa **vendedoresPorZona**, que agrupa vendedores por zona.
- Los datos del nuevo vendedor se escriben en el archivo de salida (**nombreArchivoSalida**) para que se mantengan persistidos.

- 4. Incluir una opción para modificar cualquier dato que el usuario requiera, menos el número de empleado.

```
private static void modificarVendedor(Scanner scanner) {
    System.out.print("Ingrese el código del vendedor a modificar:
");
    int codigo = scanner.nextInt();
    scanner.nextLine(); // Consumir la nueva línea

    Optional<Vendedor> vendedorEncontrado = vendedores.stream()
            .filter(v -> v.getCodigo() == codigo)
            .findFirst();

    if (vendedorEncontrado.isPresent()) {
        Vendedor vendedor = vendedorEncontrado.get();
        System.out.print("Ingrese el nuevo nombre del vendedor: ");
        String nombre = scanner.nextLine();
        vendedor.setNombre(nombre);
        System.out.print("Ingrese la nueva fecha de nacimiento del
vendedor (dd/mm/yy): ");
        String fechaNacimiento = scanner.nextLine();
        vendedor.setFechaNacimiento(fechaNacimiento);
        System.out.print("Ingrese la nueva zona del vendedor: ");
        String zona = scanner.nextLine();
        vendedor.setZona(zona);
        System.out.print("Ingrese el nuevo total de ventas
mensuales del vendedor: ");
        double ventasMensuales = scanner.nextDouble();
        vendedor.setVentasMensuales(ventasMensuales);

        System.out.println("Vendedor modificado exitosamente.");
    } else {
        System.out.println("Vendedor no encontrado.");
    }
}
```

- **Incluir una opción para modificar cualquier dato que el usuario requiera, menos el número de empleado:**

  - El método **modificarVendedor(Scanner scanner)** permite al usuario ingresar el código del vendedor que desea modificar.
  - Si se encuentra el vendedor en la lista de **vendedores**, se solicitan los nuevos datos al usuario (nombre, fecha de nacimiento, zona y ventas mensuales).
  - Luego, se actualizan los datos del vendedor seleccionado, y los cambios se reflejan tanto en la lista de **vendedores** como en el archivo de salida.

Puntos de Vallejo Leyva Marcos:

Vallejo Leyva Marcos, **Salcido Gutiérrez, Daniel Antonio, Osornio Montaño Víctor Manuel**

Aquí están los fragmentos del código que corresponden a los puntos 1 y 3 mencionados en la tarea:

- 1. Agregar un campo adicional para almacenar las ventas mensuales para cada vendedor:

```
// Agregar un campo adicional para almacenar las ventas mensuales para
cada vendedor
System.out.print("Ingrese el total de ventas mensuales del vendedor:
");
        double ventasMensuales = scanner.nextDouble();

        Vendedor nuevoVendedor = new Vendedor(codigo, nombre,
fechaNacimiento, zona, ventasMensuales);
        vendedores.add(nuevoVendedor);
        System.out.println("Vendedor agregado exitosamente.");
```

Para este punto se agregó un nuevo campo llamado "ventasMensuales" al objeto Vendedor para almacenar las ventas mensuales de cada vendedor. Este campo se solicita al usuario cuando se agrega un nuevo vendedor mediante la opción "1. Agregar Vendedor" en el menú principal. Luego, se crea una instancia de Vendedor con todos los datos ingresados por el usuario, incluidas las ventas mensuales, y se agrega a la lista de vendedores.

- 3. Incluir una opción para borrar los datos de un vendedor:

```
// Incluir una opción para borrar los datos de un vendedor
private static void borrarVendedor(Scanner scanner) {
    System.out.print("Ingrese el código del vendedor a borrar: ");
    int codigo = scanner.nextInt();

    boolean encontrado = false;
    for (Vendedor vendedor : vendedores) {
        if (vendedor.getCodigo() == codigo) {
            // Confirmación antes de borrar
            System.out.print("¿Está seguro de que desea borrar a
este vendedor? (S/N): ");
            scanner.nextLine(); // Consumir la nueva línea antes de
leer la respuesta
            String confirmacion = scanner.nextLine();
            if (confirmacion.equalsIgnoreCase("S")) {
                vendedores.remove(vendedor);
                System.out.println("Vendedor borrado
exitosamente.");
            } else {
                System.out.println("Borrado cancelado.");
            }
            encontrado = true;
            break;
        }
    }

    if (!encontrado) {
        System.out.println("Vendedor no encontrado.");
    }
}
```

Para el tercer punto se implementó una opción en el menú principal llamada "2. Borrar Vendedor" que permite al usuario ingresar el código del vendedor que desea borrar. Después de ingresar el código, se verifica si el vendedor existe en la lista de vendedores. Si el vendedor existe, se solicita una confirmación al usuario antes de realizar el borrado. Si el usuario confirma ("S"), se elimina el vendedor de la lista. Si el usuario cancela ("N"), se informa que el borrado ha sido cancelado. Si el vendedor no se encuentra, se muestra un mensaje indicando que el vendedor no fue encontrado.