Informe Nro1: Programación Avanzada

“Combustibles McQueen”

Nombre y Apellido:

Javiera Barreda

Dylan Vila

Correo:

Javiera.barreda@alumnos.ucn.cl

dylan.vila@alumnos.ucn.cl

Rut:

20.657.151-9

24.784.368-K

Nombre del Profesor:

Tomás Reimann (tomas.reimann@ce.ucn.cl)

Nombre de Ayudantes:

Rodrigo Aguilera (rodrigo.aguilera01@alumnos.ucn..cl)

Diego Cortés (diego.cortes07@alumnos.ucn.cl)

Introducción:

"El presente informe describe el desarrollo de un sistema de carga de combustible para una estación de servicio. El sistema permite a los clientes cargar combustible en diferentes tipos de vehículos, registrarse como miembros para recibir descuentos y consultar estadísticas de ventas. Se implementó en **Java**, utilizando la librería **ucn.StdIn y ucn.StdOut** para la entrada y salida de datos."

Descripción del Código:

Variables y estructura:

- Utilizamos un total de 8 variables generales para el desarrollo del código, las primeras 4 se nos brindan dentro de las instrucciones del taller, como son la cantidad de bombas, el valor del litro combustible y el valor del descuento para los miembros. Las variables restantes son en general para abarcar el total de cada punto, como es la variable "totalPagar", que se refiere al valor total que cada usuario debe pagar independiente si se le realiza un descuento o no. La variable "totalLitrosVendidos" y "totalDineroRecaudado" se asimila a lo mimo, la cantidad total de litros vendidos y el dinero total recaudado. Por ultimo las variables "totalClientes" y "totalMiembros", sirven para llevar la cuenta y el manejo de datos de los usuarios, pero haciendo la distinción de clientes y miembros.

```
int cantidadBombas = 4;
double precioLitro = 1293;
double descuento = 0.15;
double totalPagar = 0;
double totalLitrosVendidos = 0;
double totalDineroRecaudado = 0;
int totalClientes = 0;
int totalMiembros = 0;
```

- También utilizamos listas (arreglos) dentro del código, que sirven en su mayoría para guardar la información de los usuarios, ya sea el rut, nombre, ahorros y si es miembro. Las dos últimas listas, "usoBombas" es utilizada para manejar cual es la bomba que se está utilizando, la cual servirá para más adelante obtener la información de la bomba más usada y si la bomba aún tiene combustible o no. Por último, la lista "bombas" es información que nos brindaron dentro del taller, que nos indica que cada bomba debe tener 500 litros de combustible, de esta forma, en el código se le podrá ir haciendo el descuento de los litros utilizados a cada bomba.
    - Estos arreglos se usarán para el menú y los usuarios puedan ingresar sus datos, ser o no ser miembros, mostrar su ahorro, etc.

```
//listas para uso de clientes
String[] ruts = new String[1000];
String[] nombres = new String[1000];
double[] ahorros = new double[1000];
boolean[] esMiembro = new boolean[1000];
int[] usoBombas = new int[cantidadBombas];
double[] bombas = {500, 500, 500, 500};
```

- En este apartado tenemos los contadores generales que utilizamos dentro del código, ya sea para obtener la información de cuál es la bomba más usada, como llevar la cuenta de cuantos litros es vendido a cada tipo de vehículo, para así obtener cual es el vehículo más utilizado. Por último, tenemos un contador general de litros.

```
- //Contadores
  int bombaMasUsada = -99999;
  double litrosMoto = 0;
  double litrosAuto = 0;
  double litrosCamioneta = 0;
  double litrosCamion = 0;
  double litrosBus = 0;
  double litros = 0;
```

- Finalizamos las variables del Código (las principales por el momento) declarando el archivo de salida que será llenado con la información en la parte final del código.

```
- ArchivoSalida arch = new ArchivoSalida("Estadisticas.txt");
```

Funcionalidades principales:

- Haremos un “while” y continuamos imprimiendo un menú de opciones donde el usuario pueda interactúan con las funciones del programa.

```
- int opcion;
  while (true) {
      StdOut.println("============================");
      StdOut.println("=      BIENVENIDOS A      =");
      StdOut.println("=      COMBUSTIBLES       =");
      StdOut.println("=       MC QUEEN          =");
      StdOut.println("============================");
      StdOut.println("==     MENU PRINCIPAL     ==");
      StdOut.println("============================");
      StdOut.println("= [1].Cargar Combustible   =");
      StdOut.println("= [2].Ver Miembros         =");
      StdOut.println("= [3].Estadisticas         =");
      StdOut.println("= [4].Salir                =");
      StdOut.println("============================");
      StdOut.println("== SELECCIONE UNA OPCION: ==");
```

```
        StdOut.println("============================");
        opcion = StdIn.readInt();
```

Al comenzar la primera opción es "**1. Cargar Combustible**":

- El usuario ingresara su rut: para eso usamos una condicional "if" y desarrollamos una variable para que el usuario escriba su rut, otra para el que sea o no sea miembro, a continuación, creamos un bucle "for" Recorre el arreglo "ruts[]", que contiene los "RUTs" registrados y poder buscar al cliente que este registrado.

```
- if (opcion == 1) {
        StdOut.println("Ingrese su RUT:");
        String rut = StdIn.readLine();

        // Buscar si el cliente está registrado
        int indiceCliente = -1;
        for (int i = 0; i < totalClientes; i++) {

            if (ruts[i].equals(rut)) {
                indiceCliente = i;
                break;
            }
        }
```

Si no está registrado:

- Ahora en una condicional "if" usaremos el "índiceCliente" para leer si el usuario es miembro o no y se le dará la opción de registrarse como miembro.
- Si el usuario responde **si,** ingresaremos "equalsIgnoreCase("si")" permite que la comparación sea insensible a mayúsculas y minúsculas (es decir, "SI", "si", "Si" o "sI" serán aceptados), si la respuesta es afirmativa, el programa solicita el nombre del usuario, el usuario será registrado en los arreglos. Finalmente, se muestra un mensaje indicando que el usuario ha sido registrado y que se le cobrará una tarifa de membresía.

- Si el usuario **rechaza registrarse**, el programa simplemente **suma 1 a "usuarios"** y no cambia nada más.
  Si la respuesta del usuario **no es "SI" ni "NO"**, se muestra un mensaje de **error** "continue"; hace que el programa vuelva a preguntar sin continuar con el código restante.

```
•   // Si no está registrado
    if (indiceCliente == -1) {
        StdOut.println("Usted no es miembro. ¿Desea registrarse? (SI/NO)");
        String registro = StdIn.readLine();
        if (registro.equalsIgnoreCase("si") || registro.equalsIgnoreCase("SI")) {
            StdOut.println("Ingrese su nombre:");
```

```
            String nombre = StdIn.readLine();
            ruts[totalClientes] = rut;
            nombres[totalClientes] = nombre;
            ahorros[totalClientes] = 0;
            esMiembro[totalClientes] = true;
            indiceCliente = totalClientes;
            totalMiembros++;
            StdOut.println("Se ha registrado exitosamente. Se le cobrará $2500 de
membresía.");
            totalClientes++;

        } else if (registro.equalsIgnoreCase("no") || registro.equalsIgnoreCase("NO"))
{
            ruts[totalClientes] = rut;
            nombres[totalClientes] = "Usuario";
            ahorros[totalClientes] = 0;
            esMiembro[totalClientes] = false;
            indiceCliente = totalClientes;
            totalClientes++;

        } else {
            StdOut.println("Opcion no valida. Intente de nuevo.");
            continue;

        }
```

Seleccione la bomba:

Para obtener la cantidad de combustible disponible en cada bomba creamos un bucle for que recorre todas las bombas de gasolina, donde “cantidadBombas” representa el número total de bombas.

También hacemos uso de un operador ternario “?”, donde si la cantidad de litros en la bomba es menor que 10, se considera no disponible.

De lo contrario, se muestra la cantidad de litros disponibles (bombas[i] + “litros”).

Por ejemplo:

- Salida:
  ESTADO DE LAS BOMBAS:
  Bomba1: 50 litros
  Bomba2: 8 litros → NO DISPONIBLE
  Bomba3: 30 litros
  Bomba4: 20 litros

```
StdOut.println("ESTADO DE LAS BOMBAS:");
for (int i = 0; i < cantidadBombas; i++) {
    String estado = (bombas[i] < 10) ? "NO DISPONIBLE" : bombas[i] + "litros";
    StdOut.println("Bomba" + (i + 1) + ":" + estado);

}
```

Para **seleccionar una bomba de combustible** se solicita al usuario que elija una bomba (del 1 al 4). Se ajusta la selección a índices de 0 a 3 restando 1 (bomba = bomba - 1).

Si el usuario ingresa un número fuera del rango "(bomba < 0 || bomba >=cantidadBombas)", o si la bomba tiene menos de **10 litros**, se muestra un mensaje de error., y después "continue;" **reinicia el proceso** para que el usuario seleccione otra bomba.

```
-   StdOut.println("SELECCIONE UNA BOMBA (1-2-3-4)");
    int bomba = StdIn.readInt() - 1;

    if (bomba < 0 || bomba >= cantidadBombas) {
        StdOut.println("Número de bomba inválido. Intente de nuevo.");
        continue;
    }
-
```

Para **ingresar el tipo de vehículo** ingresaremos el tipo de vehículo (uno de los vehículos ya mencionados en el taller) para estadísticas. Dependiendo del tipo de vehículo, se incrementa su contador correspondiente (largoMoto, largoAuto, etc.). Si el usuario ingresa un valor no válido, se muestra un mensaje de error y el proceso vuelve a iniciar con "continue;".

```
-   StdOut.println("Ingrese tipo de vehículo (moto,auto, camioneta, camión/bus)");
    String tipoVehiculo = StdIn.readLine();

    if (tipoVehiculo.equals("moto")) {
        largoMoto++;
        litrosMoto += litros;
    } else if (tipoVehiculo.equals("auto")) {
        largoAuto++;
        litrosAuto += litros;
    } else if (tipoVehiculo.equals("camioneta")) {
        largoCamioneta++;
        litrosCamioneta += litros;
    } else if (tipoVehiculo.equals("bus") || tipoVehiculo.equals("camion")) {
        largoCamion++;
        largoBus++;
        litrosBus += litros;
```

```
        litrosCamion += litros;
    } else {
        StdOut.println("Tipo de vehículo no valido. Intentelo nuevamente.");
        continue;
    }
```

Ahora se **solicita la cantidad de litros de combustible**, se actualiza el contador “totalLitrosVendidos”, que lleva la cuenta de todos los litros vendidos y "totalDineroRecaudado” lleva la cuenta del total a pagar. Si la cantidad ingresada **supera** lo que tiene la bomba, se muestra un error y se pide otra bomba.

- Se calcula el **precio total** (precioTotal).
- Si el usuario es **miembro**, recibe un **descuento** (descuentoTotal).
- Se obtiene el **total a pagar** restando el descuento.
- Se **descuenta** el combustible de la bomba (bombas[bomba] -= litros).
- Se **suma** la cantidad vendida a totalLitrosVendidos.
- Se **actualiza** el total de dinero recaudado.
- Se incrementa el contador de **uso de la bomba**
- Si el usuario tiene membresía, se **acumula su ahorro** con un condicional if (ahorros[indiceCliente])

```
-   StdOut.println("Ingrese cantidad de litros a cargar:");
    litros = StdIn.readDouble();
    totalLitrosVendidos += litros;
    totalDineroRecaudado += totalPagar;

    if (litros > bombas[bomba]) {
        StdOut.println("No hay suficiente combustible en la bomba. Intente con otra
    bomba.");
        continue;
    }
    double precioTotal = litros * precioLitro;
    double descuentoTotal = esMiembro[indiceCliente] ? precioTotal * descuento :
    0;
    totalPagar = precioTotal - descuentoTotal;
    bombas[bomba] -= litros;
    totalLitrosVendidos += litros;
    totalDineroRecaudado += totalPagar;
    usoBombas[bomba]++;

    if (esMiembro[indiceCliente]) {
        ahorros[indiceCliente] += descuentoTotal;

    }
```

Ahora desplegamos una **boleta** en la cual vamos a entregar los datos/resultados de la solicitud del usuario, se dará informe del cliente (nombre, rut, litros cargados, descuento, etc).

Se obtiene el nombre del cliente desde el array nombres usando indiceCliente, que almacena la posición del cliente en el registro.

Se muestra el nombre y el RUT del cliente en la boleta.

Indica cuántos litros de combustible fueron cargados y el tipo de vehículo.

Muestra el **precio por litro**, que es una constante definida en el programa.

Se muestra el **precio total** de la compra antes de aplicar descuentos.

Si el cliente es miembro, se le aplica un **descuento** del 15% (descuentoTotal).

Finalmente, se muestra el **monto final a pagar** después del descuento con un mensaje de cortesía para cerrar la transacción...

```
-    StdOut.println("------BOLETA------");
        String nombreMiembro = nombres[indiceCliente];
        StdOut.println("Nombre: " + nombreMiembro);
        StdOut.println("Rut: " + rut);

        StdOut.println("Litros cargados: " + litros);
        StdOut.println("Tipo de vehículo: " + tipoVehiculo);
        StdOut.println("Precio por litro: $" + precioLitro);
        StdOut.println("Precio total: $" + precioTotal);
        StdOut.println("Descuento aplicado: $" + descuentoTotal);
        StdOut.println("Total pagar: $" + totalPagar);
        StdOut.println("Gracias por su compla. ¡Vuelva pronto!");
    }
```

Y ahora empezamos con la primera opción **“2. Ver Miembros”:**

Esta parte del código nos permite visualizar los usuarios registrados como miembros, y ordenarlos de mayor a menor según el ahorro.

Si no hay clientes (totalClientes == 0), se muestra un mensaje y se termina la ejecución de esta opción.

Para **ordenar por ahorro de mayor a menor** usaremos el ordenamiento burbuja contando el total de clientes y con el condicional “if” (ahorros[j] < ahorros[j+1]) significa que el cliente actual tiene menos ahorro que el siguiente, por lo que se intercambian

Se intercambiarán los valores en el arreglo de **ahorros**.

Se intercambian los **nombres** de los clientes en el mismo orden.

Se intercambian los **ruts** correspondientes.

Se intercambia el estado de membresía con el arreglo **miembro** (true o false), asegurando que los datos sigan siendo consistentes.

Muestra la lista de **miembros ordenados registrados por ahorro**, recorriendo todos los clientes y verificando si esMiembro[i]es un **true**.

Si un cliente es miembro, se imprime su **nombre y ahorro acumulado**.

```
-   //Registro: Ver miembros:
    } else if (opcion == 2) {
        if (totalClientes == 0) {
            StdOut.println("No hay miembros registrados");
        }  else {
            // Ordenar por ahorro de mayor a menor
            for (int i = 0; i < totalClientes - 1; i++) {
                for (int j = 0; j < totalClientes - i - 1; j++) {
                    if (ahorros[j] < ahorros[j + 1]) {
                        // Intercambiar ahorros
                        double contAhorro = ahorros[j];
                        ahorros[j] = ahorros[j + 1];
                        ahorros[j + 1] = contAhorro;

                        // Intercambiar nombres
                        String contNombre = nombres[j];
                        nombres[j] = nombres[j + 1];
                        nombres[j + 1] = contNombre;

                        // Intercambiar ruts
                        String contRut = ruts[j];
                        ruts[j] = ruts[j + 1];
                        ruts[j + 1] = contRut;

                        // Intercambiar estado de miembro
                        boolean contEsMiembro = esMiembro[j];
                        esMiembro[j] = esMiembro[j + 1];
                        esMiembro[j + 1] = contEsMiembro;
                    }
                }
            }

            StdOut.println("Miembros registrados ordenados por ahorro:");
            for (int i = 0; i < totalClientes; i++) {
                if(esMiembro[i]) { // solo se muestran miembros
                    StdOut.println("Nombre: " + nombres[i] + " - Ahorro:
```

```
-      $" + ahorros[i]);
                }
            }
        }
```

Y ahora empezamos con la tercera opción **“3. Estadisticas”:**

Para la **Opción 3** del código, se despliegan las estadísticas requeridas, de esta forma se ejecuta solo si la variable opción es igual a 3

- Para desplegar la información del porcentaje total de miembros utilizamos un operador ternario (condición ? Valor_si_verdadero : valor_si_falso), la condición utilizada es “totalClientes > 0 “de esta forma se evalúa que clientes sea mayor a 0.
- **Si es verdadero:**
    - Se calcula el porcentaje de miembros dividiendo el total de miembros con el total de clientes, el resultado es multiplicado por 100 para expresarlo en porcentaje.

- **Si es falso (totalClientes == 0):**
    - Se muestra 0 para evitar una división por cero.

```
} else if (opcion == 3) {
    StdOut.println("----ESTADÍSTICAS----");
    StdOut.println("Porcentaje de miembros:" + (totalClientes > 0 ? ((double)
totalMiembros / totalClientes) * 100 : 0) + "%");
```

Para lograr obtener el vehículo más usado, definimos una variable “vehiculoMasUsado” que inicialmente tiene el valor “moto”. Ya que se establece en la segunda línea que los litros de moto son “maxCarga” es decir, la mayor carga de combustible.

De esta forma hacia abajo, si el auto ha consumido más litros que la moto, se actualiza la mayor carga con los litros de auto y el vehículo más usado se cambia a “auto”. Este patrón se repite con camioneta, camion y bus.

Por último, se despliega por consola el vehículo que más cargo.

```
String vehiculoMasUsado = "moto";
double maxCarga = litrosMoto;
if (litrosAuto > maxCarga) {
    maxCarga = litrosAuto;
    vehiculoMasUsado = "auto";
}
```

```
if (litrosCamioneta > maxCarga) {
    maxCarga = litrosCamioneta;
    vehiculoMasUsado = "camioneta";
}
if (litrosCamion > maxCarga) {
    maxCarga = litrosCamion;
    vehiculoMasUsado = "camion";
}

if (litrosBus > maxCarga) {
    maxCarga = litrosBus;
    vehiculoMasUsado = "bus";
}
StdOut.println("Vehículo que más cargó:" + vehiculoMasUsado);
```

Para obtener la bomba más usada, comenzamos inicializando la variable de "bombaMasUsada" con valor 0, posteriormente creamos un bucle for para encontrar la bomba más usada.

Se recorre el arreglo "usoBombas", el cual almacena el número de veces que cada bomba ha sido utilizada.

El bucle for empieza en i=1 porque al comienzo dijimos que la bomba más usada inicialmente es la 0.

De esta forma si el código encuentra una bomba que tiene más usos que la actual (usoBombas[bombaMasUsada]), se actualiza la variable "bombaMasUsada" con el nuevo índice i.

Por último, se imprimen las restantes estadísticas como es la bomba más que logramos obteniendo gracias al bucle anterior, el total de litros vendidos que obtenemos en la opción 1 del código y el total de dinero recaudado que igualmente obtenemos en la opción 1.

```
bombaMasUsada = 0;
for (int i = 1; i < usoBombas.length; i++) {
    if (usoBombas[i] > usoBombas[bombaMasUsada]) {
        bombaMasUsada = i;
    }
}
StdOut.println("Bomba más usada: " + (bombaMasUsada + 1));
StdOut.println("Litros Vendidos:" + (totalLitrosVendidos));
StdOut.println("Dinero recaudado: " + (totalDineroRecaudado));
```

Y ahora empezamos con la primera opción **“4. Salir”:**

Esta parte del código maneja la opción de salida **cerrar el programa** donde se muestra un mensaje de despedida y se almacenan algunos datos (archivo de salida), el cuál debe incluir las estadísticas mencionadas anteriormente.

En el momento que el usuario ingresa la opción 4 el programa imprime un mensaje de despedida.

Se crea un objeto tipo registro (regSal), pasando por el parámetro 4.

Calculamos el porcentaje de los litros vendidos por cliente (totalLitrosVendidos / (totalClientes > 0 ? totalClientes: 1). La división evita errores si totalClientes > 0 (previene división por cero usando “?” totalClientes :1).

Luego, multiplica por 100 para expresarlo como un porcentaje y lo almacena en el registro.

Guardamos la bomba de gasolina mas usada.

Guardamos el total de litros vendidos.

Guardamos el total de dinero recaudado.

“(arch.writeRegistro(regSal))” escribe el registro en un archivo de las estadísticas de ventas.

Y “**(arch,close())”** Cierra el archivo, asegurando que todos los datos se guarden correctamente.

Break termina con el bucle while.

Si el usuario ingresa una **opción diferente a las permitidas**, se muestra un mensaje de error **"Opción no válida. Intente de nuevo.".**

```
-    else if (opcion == 4) {
                StdOut.println("Gracias por visitarnos. ¡Vuelva pronto!");

                Registro regSal = new Registro(4);
                regSal.agregarCampo(totalLitrosVendidos / (totalClientes > 0 ? totalClientes: 1) * 100
    + "%");
                regSal.agregarCampo(bombaMasUsada);
-               regSal.agregarCampo(totalLitrosVendidos);
                regSal.agregarCampo(totalDineroRecaudado);

                arch.writeRegistro(regSal);
                arch.close();

                break; // Termina el bucle while

            } else {
                StdOut.println("Opción no válida. Intente de nuevo.");
```

```
        }
      }
    }
  }
```

Cantidad de horas que cada integrante dedicó a la realización del taller.

	Diseñar	Codificar	Informe
Javiera Barreda A.	2 horas	6 horas	3 horas
Dylan Vila	2 horas	4 horas	4 horas