**Universidad de Buenos Aires - Facultad de Ciencias Económicas**
**Licenciatura en Sistemas de Información de las Organizaciones**
**TEORÍA DE LENGUAJES Y ALGORITMOS – PRIMER PARCIAL**
PRIMER CUATRIMESTRE DE 2019

**Fecha:** 09 de mayo de 2019

**Nombre y Apellido**…….…….....…… **Registro**…….....……

Un fabricante de heladeras busca desarrollar una aplicación sencilla para el técnico que visita a los clientes por un desperfecto. Se busca que pueda consultar los domicilios a visitar y registrar el resultado. Cuando el cliente reclama por el funcionamiento de su heladera, la novedad se carga en otro sistema, y a esta aplicación le llegará solamente un archivo con las visitas que debe realizar el técnico que la ejecuta.
A continuación se describen los datos que debe manejar la aplicación, y su funcionalidad:

- ✓ Hasta 100 visitas para reportar, almacenando para cada una:
  - Número de Visita (numérico entre 1 y 999.999, pueden no ser consecutivos)
  - Domicilio a Visitar (texto)
  - Nombre del Cliente a visitar (texto)
  - Descripción del Síntoma de mal funcionamiento (texto)
  - Descripción de la Causa y la Solución (texto; un solo campo para ambos datos)
  - Estado: (texto) indica si la visita se encuentra pendiente (P), o visitada (V)

  Para visitas vacías (no utilizadas), los campos numéricos son 0 y los de texto "".

Construir parte de la aplicación, de acuerdo a las siguientes especificaciones:

1. **(NO DESARROLLAR)** Al iniciarse, la aplicación cargará las visitas pendientes del archivo "visitas_pend.csv", mediante el método CargaInicial.
2. Luego **NO** presentará un menú, sino que llamará directamente al método **CompletarVisita** de la clase **ListaVisitas**.
   Allí la aplicación armará una lista con las visitas pendientes (como se ve en la próxima imagen). Si no hubiera ninguna mostrará el mensaje "No hay visitas pendientes". Si las hubiera mostrará la lista mientras solicita el número de visita a registrar:

```
Ingrese número de visita:
Número  Domicilio       Cliente         Problema
401     JUNIN 2399      SILVA, ANALIA   NO ENFRIA
405     VIAMONTE 2110   ACOSTA, ALBERTO PÉRDIDAS DE LIQUIDO
408     PARAGUAY 2542   SPLIPEK, PAULA  EXCESO DE RUIDO
```

3. Una vez que el técnico ingresa el número de visita deseada, la aplicación buscará la visita por su número.
4. Si no existen visitas con ese número, la aplicación informará la situación al usuario y salteará los pasos siguientes hasta el número 9, donde se consulta al usuario si quiere continuar en la aplicación.
5. Si existe una visita con ese número, pero la misma no está pendiente, la aplicación informará la situación al usuario y salteará los pasos siguientes hasta el número 9, donde se consulta al usuario si quiere continuar en la aplicación.
6. Satisfechas estas validaciones, la aplicación solicitará confirmación para proceder o no a completar la visita seleccionada ("S" o "N"):

```
Desea completar esta visita (S/N):
Número  Domicilio       Cliente         Problema
405     VIAMONTE 2110   ACOSTA, ALBERTO PÉRDIDAS DE LIQUIDO
```

7. Si responde afirmativamente, la aplicación solicitará la causa y solución, registrará este dato en la visita junto con el cambio de estado, y mostrará el mensaje "Completó satisfactoriamente la visita" y salteará los pasos siguientes hasta el número 9, donde se consulta al usuario si quiere continuar en la aplicación.
8. Si responde negativamente, la aplicación mostrará el mensaje "Eligió NO completar esta visita", y continuará con el paso siguiente.
9. **(NO DESARROLLAR)** Finalmente se consultará al usuario si desea continuar ("S" o "N"). Si desea continuar, la aplicación vuelve al punto 2. Si desea no continuar, la aplicación guardará las visitas pendientes las visitas pendientes del archivo "visitas_pend.csv", las visitas completadas en el archivo "visitas_visi.csv" y se cerrará.

---

***Se pide:*** **Desarrollar** el método **CompletarVisita** de la clase **ListaVisitas**, y todos los métodos que considere necesarios para los mismos.
***Aclaraciones:***

- ✓ Todos los ingresos de datos deben de ser validados (de acuerdo a los valores máximos y mínimos definidos en cada caso, y en ningún caso se admiten valores vacíos).
- ✓ Para todos los valores inválidos, se debe pedir al usuario que vuelva a ingresar el mismo, salvo en los casos donde se especifique otra acción (Ejemplo: "saltear los pasos hasta")
- ✓ La no utilización de las funcionalidades previamente desarrolladas será considerada como error.

## *CÓDIGO DESARROLLADO HASTA EL MOMENTO:*

**class Visita**

```
public const string EstPend = "P";
public const string EstVisi = "V";
public const long NumMin = 1;
public const long NumMax = 999999;
public long Numero { get; private set; }
public string Domicilio { get; set; }
public string NombreCliente { get; set; }
public string Sintoma { get; set; }
public string CausaSolucion { get; private set; }
public string Estado { get; private set; }
```

**class Visita (cont.)**

```
public void Reportar(string causaSoluc)
{
  CausaSolucion = causaSoluc;
  Estado = Visita.EstVisi;
}
public string Resumir()
{
  return Numero + "\t"
   + Domicilio + "\t"
   + NombreCliente + "\t"
   + Sintoma + "\n";
}
```

**class ListaVisitas**

```
  private Visita[] visitas;
  public ListaVisitas(int cantidad)
    //establece el tamaño del arreglo visitas
    //según la "cantidad" recibida, e inicializa
    //todas las posiciones como vacías
```

**class ListaVisitas (cont.)**

```
public void CargaInicial()
//carga las visitas pendientes del archivo
//"visitas_pend.csv" en el arreglo visitas

public void GuardadoFinal()
//guarda las visitas pendientes del arreglo visitas
//en el archivo "visitas_pend.csv", y las visitadas
//en el archivo "visitas_visi.csv"
```

```
class Program
static void Main(string[] args)
{
  const int cantidad = 100;

  string opcion = "";
  ListaVisitas listaVis = new ListaVisitas(cantidad);
  listaVis.CargaInicial();
  do
  {
    listaVis.CompletarVisita();
    opcion = ServValidac.PedirSoN("¿Desea continuar? S/N");
  } while (opcion == "S");
  listaVis.GuardadoFinal();
}
```

```
static class ServValidac
  public static string PedirStrNoVac(string mensaje)
    //muestra "mensaje" por consola, y pide ingreso
    //por consola hasta que el usuario ingrese un texto no vacío
    //para devolverlo

  public static string PedirSoN(string mensaje)
    //muestra "mensaje" por consola, y pide ingreso
       //por consola hasta que el usuario ingrese S o N
       //para devolverlo

  public static long PedirLong(string mensaje, long min, long max)
    //muestra "mensaje" por consola, y pide ingreso
    //por consola hasta que el usuario ingrese un numero entre
    //min y max para devolverlo
```