TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CULIACÁN

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

PROGRAMACIÓN II

M.C. GLORIA EKATERINE PERALTA PEÑUÑURI

RESERVACIÓN DE VUELOS

PROYECTO VERSIÓN 1

BAJO PRECIADO CARLOS ALEXIS

GONZÁLEZ VELÁZQUEZ KAREN GUADALUPE

VIDRIALES TRUJILLO RENÉ CECIL

MARZO 2017 **CULIACÁN, SINALOA**

Reservación de vuelos

Este sistema permitirá la venta de boletos de una línea aérea. También es importante agregar la opción de pasajeros frecuentes que acumularán millas. Se manejará información para ciudades a las que vuela la aerolínea, vuelos que ofrece, datos para usuarios de club premier y boletos vendidos.

Opciones a incluir:

- **Alta de ciudades:** que serán destinos u orígenes para viajar. Los datos a incluir son: Clave de la ciudad, Nombre de la ciudad, Estado. Utiliza un arreglo de objetos para que guardes la información de las ciudades.

- **Alta de vuelos:** Aquí se permitirá agregar los datos de los vuelos: número de pasajeros, origen, y destino (tomado de las ciudades tanto en origen como destino), capacidad, boletos vendidos, costo, millas, días en que se realiza el vuelo (diaria, lunes, martes, etc..). Utiliza una clase vuelo para agregar objetos a un ArrayList.

- **Alta club premier:** se darán de alta los pasajeros que cuenten con club premier para que acumulen millas en cada vuelo que compren, los datos que se deben de considerar son: clave del club premier, Nombre, Domicilio, Millas acumuladas. Se debe de crear una clase, llamada ClubPremier para que se agreguen los objetos a una TablaHash.

- **Compra de boletos:** Se utilizará para vender boletos de los vuelos dados de alta, seleccionar el origen, destino y siempre y cuando se tenga disponibilidad. Hay que considerar Clave del Boleto, Nombre del pasajero, Edad del pasajero, Clave del vuelo (tomado del catálogo de vuelos), Clave del club premier (si es que se tiene). Al realizar una compra se debe de considerar actualizar la información de los datos de Boletos vendidos (para el vuelo seleccionado, en el ArrayList de vuelos) y también se deben de acumular las millas en caso de que el pasajero esté dado de alta en el club premier de la TablaHash de los clientes que tienen Club Premier.

- **Consulta de vuelos disponibles:** Aquí aparecerán todos los vuelos para una ciudad que se proporcione, siempre y cuando el vuelo tenga disponibilidad.

TABLA DE COLECCIONES

Tipo de colección	NOMBRE USADO	CLASE BASE
Arreglo de objetos	Arreglo	Ciudad
Hashtable	hashClub	ClubPremier
ArrayList	arrayVuelos	Vuelo
ArrayList	boletosVendidos	Boleto

ACTIVIDADES REALIZADAS

Nos dividimos el trabajo y estuvimos trabajando en un repositorio creado en GitHub.:

- René se encargó de la clase base de Ciudad, así como de los métodos correspondientes a las altas, validaciones al agregar, etc.
- Carlos creó el menú, la clase base del ClubPremier y los métodos para darlos de alta, el método para comprar boletos e implementó RandomKey para generar aleatoriamente la clave de los boletos vendidos. También otros métodos de validaciones y la librería de ConsoleTables.
- Karen creó la clase base de Vuelo y de Boleto. Los métodos para dar de alta y las validaciones al momento de registrar un vuelo. Implementó la librería de ReadHelper para ayudar al leer los datos, entre otras validaciones correspondientes con la funcionalidad del programa.

Program.cs

```
using System;
using System.Collections;
using System.Linq; // Linq sólo se usó para el método random
key
using ConsoleTables;
using ReadHelperLibrary;

namespace Reservación_Vuelos {
    class MainClass {

        private int Contador = 0;
        private Ciudad[] Arreglo = new Ciudad[100];
        private Hashtable hashClub = new Hashtable();
        private ArrayList arrayVuelos = new ArrayList();
        private ArrayList boletosVendidos = new ArrayList();
        private Vuelo newVuelo;
        private Boleto newBoleto;
        private Random random = new Random();
        private ClubPremier newCliente; //Tengo declarado el
objeto a la altura de la clase para
                                        // hacer pool de obje
tos y no estar creando nuevas instancias

        public static void Main (string[] args) {
            MainClass here = new MainClass();
            here.Menu();
        }

        private void Menu(){

            int opc;
            do {
                Console.Clear();
                Console.WriteLine("1.- Alta de ciudades.");
                Console.WriteLine("2.- Alta de vuelos.");
                Console.WriteLine("3.- Alta club premier.");
                Console.WriteLine("4.- Compra de boletos.");
                Console.WriteLine("5.- Consulta de vuelos dis
ponibles.");
                Console.WriteLine("6.- Club Premier.");
                Console.WriteLine("7.- Boletos vendidos.");
                Console.WriteLine("8 - Salir.");
                opc = ReadHelperLibrary.ReadHelper.ReadInt("=
> ");
```

```
                Console.Clear();
                switch (opc) {
                    case 1:
                        AltaCiudades();
                        break;
                    case 2:
                        AltaVuelo();
                        break;
                    case 3:
                        AltaClubPremier();
                        break;
                    case 4:
                        CompraBoletos();
                        break;
                    case 5:
                        Vuelos();
                        break;
                    case 6:
                        ClubPremier();
                        break;
                    case 7:
                        BoletosVendidos();
                        break;
                    case 8:
                        opc = Salir();
                        break;
                    default:
                        Console.WriteLine("Esa no fue una opc
ión válida. :c");
                        ConfirmarSalida();
                        Console.Clear();
                        break;
                }
            } while(opc != 8);
        }

        // ALTAS
        private void AltaClubPremier() {
            string nombre, domicilio;
            double millas=0.0;

            Console.WriteLine("==========================");
            Console.WriteLine("$$ ALTA DE CLUB PREMIER $$");
            Console.WriteLine("==========================");

            nombre = ReadHelperLibrary.ReadHelper.ReadString(
```

```
"Nombre del cliente: ");

            if (ValidarNombre(nombre))
                Console.WriteLine("\n>> Ese cliente ya está r
egistrado en Club Premier <<\n");
            else {
                domicilio = ReadHelperLibrary.ReadHelper.Read
String("Domicilio: ");
                newCliente = new ClubPremier(nombre, domicili
o, millas);
                hashClub.Add(hashClub.Count + 1, newCliente);
                Console.WriteLine("\n>> Cliente añadido al cl
ub exitosamente <<\n");
            }

            ConfirmarSalida();
        }

        private void AltaVuelo () {
            // Primero validar si hay ciudades disponibles
            if (Contador <2 ) {
                Console.WriteLine("\n>> Aún no se han agregad
o suficientes ciudades <<\n");
                ConfirmarSalida();
                return;
            }

            int numPasajeros;
            string origen, destino, dias;
            double costo, millas, capacidad;

            Console.WriteLine("=========================");
            Console.WriteLine("$$ ALTA DE VUELOS $$");
            Console.WriteLine("=========================");

            Ciudades();
            // Validar que existan en el arreglo de objetos
            int posicion;

            do {
                origen = ReadHelperLibrary.ReadHelper.ReadStr
ing("Clave origen: ");
                posicion = CityPosition(origen);

                if (posicion == -1)
                    Console.WriteLine("\nNo existe esa ciudad
```

```
. Ingrese otra. \n");

            } while (posicion == -1);

            do {
                destino = ReadHelperLibrary.ReadHelper.ReadSt
ring("Clave destino: ");
                posicion = CityPosition(destino);

                if (posicion == -1)
                    Console.WriteLine("\nNo existe esa ciudad
. Ingrese otra. \n");

                if (destino == origen) {
                    Console.WriteLine("\nLa ciudad de destino
 no puede ser igual a la de origen. Ingrese otra. \n");
                    posicion = -1;
                }

            } while (posicion == -1);

            numPasajeros = ReadHelperLibrary.ReadHelper.ReadI
nt("Número de pasajeros disponibles en ese vuelo: ");
            capacidad = ReadHelperLibrary.ReadHelper.ReadDoub
le("Capacidad del avión: ");
            costo = ReadHelperLibrary.ReadHelper.ReadDouble("
Costo: ");
            millas = ReadHelperLibrary.ReadHelper.ReadDouble(
"Millas: ");

            do {
                Console.WriteLine("\nDías en que se realiza e
l vuelo: ");
                Console.WriteLine("1.- Diaria\n2.- Lunes\n3.-
 Martes\n4.- Miércoles\n5.- Jueves\n6.- Viernes\n7.- Sábado"
+
                                  "\n8.- Domingo");
                int opc = ReadHelperLibrary.ReadHelper.ReadIn
t("\nOpción: ");
                dias = DiaVuelo(opc);
            } while (dias == "");

            newVuelo = new Vuelo(arrayVuelos.Count+1,numPasaj
eros,origen, destino, dias, capacidad, 0, costo, millas);
            arrayVuelos.Add(newVuelo);
            Console.WriteLine("\n>> VUELO AÑADIDO CON ÉXITO <
```

```
<\n");
            ConfirmarSalida();
        }

        private void AltaCiudades () {
            if (Contador == 10) {
                Console.WriteLine("\n>> YA SE HAN REGISTRADO
LAS 100 CIUDADES <<\n");
                ConfirmarSalida();
                return;
            }

            string clave = ReadHelperLibrary.ReadHelper.ReadS
tring("Introduzca clave de la ciudad: ");

            if (ValidarClave(clave)) {
                Console.WriteLine("\n>> YA EXISTE ESA UNA CIU
DAD CON ESA CLAVE <<\n");
                ConfirmarSalida();
                return;
            }

            string nombre = ReadHelperLibrary.ReadHelper.Read
String("Introduzca nombre de la ciudad: ");
            string estado = ReadHelperLibrary.ReadHelper.Read
String("Introduzca estado de la ciudad: ");
            Ciudad City = new Ciudad(clave, nombre, estado);
            Console.WriteLine("\n>> CIUDAD GUARDADA <<\n");
            Arreglo[Contador] = City;
            Contador++;
            ConfirmarSalida();
        }

        private void CompraBoletos() {

            if (arrayVuelos.Count == 0 || !vuelosDisponibles(
)) {
                Console.WriteLine("\n>> NO HAY VUELOS DISPONI
BLES <<\n");
                ConfirmarSalida();
                return;
            }

            int claveVuelo, claveClub;
            string claveBoleto, nomPasajero;
            int edad, opc;
```

```
            do {
                claveBoleto = RandomKey(15).Trim().ToUpper();
            } while (existeBoleto(claveBoleto));

            Vuelos();

            claveVuelo = ReadHelperLibrary.ReadHelper.ReadInt
("Clave del vuelo: ");

            if (existeVuelo(claveVuelo) == null) {
                Console.WriteLine("Ese vuelo no está disponib
le.");
            } else {

                opc = ReadHelperLibrary.ReadHelper.ReadIntRan
go("¿Club premier? 1.- Si , 2.- No : ", 1,2);

                if (opc == 1) {

                    if (hashClub.Count == 0) {
                        Console.WriteLine("\n>> Primero debe
registrarse en club premier <<\n");
                        ConfirmarSalida();
                        return;
                    }


                    do {
                        ClubPremier();
                        claveClub = ReadHelperLibrary.ReadHel
per.ReadInt("Clave del club premier: ");
                    } while (!hashClub.ContainsKey(claveClub)
);
                    nomPasajero = ((ClubPremier)hashClub[clav
eClub]).Nombre;
                    edad = ReadHelperLibrary.ReadHelper.ReadI
nt("Edad del pasajero: ");

                    newBoleto = new Boleto(claveBoleto, nomPa
sajero, edad, claveVuelo);
                    newBoleto.ClaveClubPremier = claveClub;

                    existePremier(claveClub).Millas = existeV
uelo(claveVuelo).pMillas;
                    existeVuelo(claveVuelo).pBoletosVendidos
```

```
= 1;

                    boletosVendidos.Add(newBoleto);
                    Console.WriteLine("Boleto vendido.");
                } else {
                    nomPasajero = ReadHelperLibrary.ReadHelpe
r.ReadString("Nombre Pasajero: ");
                    edad = ReadHelperLibrary.ReadHelper.ReadI
ntRango("Edad del pasajero: ",1,105);
                    newBoleto = new Boleto(claveBoleto, nomPa
sajero, edad, claveVuelo);
                    boletosVendidos.Add(newBoleto);
                    existeVuelo(claveVuelo).pBoletosVendidos
= 1;
                    Console.WriteLine("\n>> BOLETO VENDIDO <<
\n");
                }
            }
            ConfirmarSalida();
        }

        // MÉTODOS DE IMPRESIONES
        private void Ciudades () {
            if (Contador == 0)
                Console.WriteLine("No se han ingresado ciudad
es.");
            else {
                var table = new ConsoleTable("Clave","Ciudad"
, "Estado");
                for (int i = 0; i < Contador; i++) {
                    table.AddRow(Arreglo[i].pClave,Arreglo[i]
.pNombre, Arreglo[i].pEstado);
                }
                table.Write(Format.Alternative);
                Console.WriteLine();
            }

            ConfirmarSalida();
        }

        private void ClubPremier () {
            if (hashClub.Count == 0)
                Console.WriteLine("\n>> AÚN NO HAY CLIENTES P
ERTENECIENTES A CLUB PREMIER << \n");
            else {
                var table = new ConsoleTable("ID", "NOMBRE",
```

```
"DOMICILIO", "MILLAS");
                foreach (DictionaryEntry val in hashClub) {
                    newCliente = (ClubPremier)val.Value;
                    table.AddRow(val.Key, newCliente.Nombre,
newCliente.Domicilio, newCliente.Millas);
                }
                table.Write(Format.Alternative);
                Console.WriteLine();
            }

            ConfirmarSalida();
        }

        private void Vuelos () {
            if (arrayVuelos.Count == 0)
                Console.WriteLine("\n>> No hay vuelos registr
ados << \n");
            else {
                var table = new ConsoleTable("CVE", "ORIGEN",
 "DESTINO", "# PASAJEROS", "BOLETOS VENDIDOS", "CAPACIDAD", "
MILLAS", "DÍAS", "$");
                int contVuelos = 0;
                foreach (Vuelo val in arrayVuelos)
                    if (val.pBoletosVendidos < val.pNumPasaje
ros) {
                        table.AddRow(val.pClaveVuelo, val.pOr
igen, val.pDestino, val.pNumPasajeros, val.pBoletosVendidos,
val.pCapacidad, val.pMillas, val.pDias, val.pCosto);
                        contVuelos++;
                    }

                if (contVuelos == 0)
                    Console.WriteLine("\n>> No hay vuelos dis
ponibles << \n");
                else
                    table.Write(Format.Alternative);

                Console.WriteLine();
            }

            ConfirmarSalida();
        }

        private void BoletosVendidos () {
            if (boletosVendidos.Count == 0)
                Console.WriteLine("\n>> No hay boletos vendid
```

```
os << \n");
            else {
                var table = new ConsoleTable("CLAVE", "NOMBRE
", "EDAD", "CLAVE VUELO", "CLUBPREMIER");
                string cve="--";
                foreach (Boleto val in boletosVendidos) {
                    if (val.ClaveClubPremier != 0)
                        cve = Convert.ToString(val.ClaveClubP
remier);
                    table.AddRow(val.ClaveBoleto, val.Nompasa
jero, val.EdadPasajero, val.ClaveVuelo, cve);
                }
                table.Write(Format.Alternative);
                Console.WriteLine();
            }
            ConfirmarSalida();
        }

        // VALIDACIONES - EXISTENCIAS
        private Vuelo existeVuelo (int clave) {
            foreach (Vuelo val in arrayVuelos)
                if ((val.pClaveVuelo == clave) && (val.pBolet
osVendidos < val.pNumPasajeros))
                    return val;

            return null;
        }

        private ClubPremier existePremier (int key) {
            foreach (DictionaryEntry entry in hashClub)
                if (entry.Key.Equals(key))
                    return (ClubPremier)entry.Value;

            return null;
        }

        private bool ValidarClave (string cve) {
            for (int i = 0; i < Contador; i++) {
                Ciudad c = Arreglo[i];
                if (c.pClave.CompareTo(cve) == 0)
                    return true;
            }

            return false;
        }
```

```
        public bool ValidarNombre (string n) {
            foreach (DictionaryEntry entry in hashClub) {
                ClubPremier c = (ClubPremier)entry.Value;
                if (c.Nombre.CompareTo(n) == 0)
                    return true;
            }

            return false;
        }

        private Ciudad CityObject () {
            Ciudad ciudad = null;
            if (Contador == 0)
                Console.WriteLine("\nNo se han agregado ciuda
des.\n");
            else {
                string Nombre = ReadHelperLibrary.ReadHelper.
ReadString("Introduzca clave de la ciudad: ");
                for (int i = 0; i < Contador; i++)
                    if (Arreglo[i].pClave.CompareTo(Nombre) =
= 0) {
                        ciudad = Arreglo[i];
                        break;
                    }
            }
            return ciudad;
        }

        private int CityPosition (string Nombre) {
            int Position = -1;

            for (int i = 0; i < Contador; i++)
                if (Arreglo[i].pClave.CompareTo(Nombre) == 0)
 {
                    Position = i;
                    break;
                }

            return Position;
        }

        private bool existeBoleto (string claveBoleto) {
            foreach (Boleto val in boletosVendidos)
                if (val.ClaveBoleto.Equals(claveBoleto))
                    return true;
```

```
            return false;
        }

        private bool vuelosDisponibles () {
            foreach (Vuelo val in arrayVuelos)
                if ((val.pBoletosVendidos < val.pNumPasajeros
))
                    return true;

            return false;
        }

        // MÉTODOS AUXILIARES
        private string DiaVuelo (int dia) {
            switch (dia) {
                case 1: return "DIARIA";
                case 2: return "LUNES";
                case 3: return "MARTES";
                case 4: return "MIÉRCOLES";
                case 5: return "JUEVES";
                case 6: return "VIERNES";
                case 7: return "SÁBADO";
                case 8: return "DOMINGO";
                default:
                    Console.WriteLine("\n>> OPCIÓN INVÁLIDA <
< \n");
                    return "";
            }
        }


        public string RandomKey (int length) {
            const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0
123456789";
            return new string(Enumerable.Repeat(chars, length
)
              .Select(s => s[random.Next(s.Length)]).ToArray(
));
        }

        private int Salir () {
            int opc;
            Console.WriteLine("=========================");
            Console.WriteLine("Estás saliendo del sistema");
            Console.WriteLine("=========================");
            Console.WriteLine("  ¡Los datos se perderan!");
```

```
            Console.WriteLine("1.- Para salir.");
            Console.WriteLine("2.- Para volver.");
            opc = ReadHelperLibrary.ReadHelper.ReadInt("=> ")
;
            switch (opc) {
                case 1:
                    Console.Clear();
                    return 8;

                case 2:
                    return 0;

            }
            Console.Clear();
            return 0;
        }

        private void ConfirmarSalida () {
            Console.WriteLine("\n\t\t\t————————————————————
————————");
            Console.WriteLine("\t\t\tPresiona enter para cont
inuar");
            Console.WriteLine("\t\t\t————————————————————
——————");
            Console.ReadLine();
        }

    }
}
```

Boleto.cs

```
using System;
namespace Reservación_Vuelos {
    public class Boleto {

        private string claveBoleto;
        private string nomPasajero;
        private int edadPasajero;
        private int claveVuelo;
        private int claveClubPremier;

        public Boleto(string claveBoleto,string nomPasajero,i
nt edadPasajero, int claveVuelo) {
            this.claveBoleto = claveBoleto;
            this.nomPasajero = nomPasajero;
            this.edadPasajero = edadPasajero;
            this.claveVuelo = claveVuelo;
        }

        public string ClaveBoleto {
            get { return claveBoleto; }
            set { claveBoleto = value; }
        }

        public string Nompasajero {
            get { return nomPasajero; }
            set { nomPasajero = value; }
        }

        public int EdadPasajero {
            get { return edadPasajero; }
            set { edadPasajero = value; }
        }

        public int ClaveVuelo {
            get { return claveVuelo; }
            set { claveVuelo = value; }
        }

        public int ClaveClubPremier {
            get { return claveClubPremier; }
            set { claveClubPremier = value; }
        }

    }
```

Ciudad.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Reservación_Vuelos {
    class Ciudad {
        string Clave;
        string Nombre;
        string Estado;

        public Ciudad(string clave, string nombre, string est
ado) {
            Clave = clave;
            Nombre = nombre;
            Estado = estado;
        }

        public string pClave {
            get {
                return Clave;
            }

            set {
                Clave = value;
            }
        }

        public string pNombre {
            get {
                return Nombre;
            }

            set {
                Nombre = value;
            }
        }

        public string pEstado {
            get {
                return Estado;
            }
```

```
            set {
                Estado = value;
            }
        }
    }
}
```

ClubPremier.cs

```
using System;
namespace Reservación_Vuelos {
    public class ClubPremier {
        //clave del club premier, Nombre, Domicilio, Millas a
comuladas.
        private string nombre;
        private string domicilio;
        private double millas;

        public ClubPremier(string nombre, string domicilio, d
ouble millas) {
            this.nombre = nombre;
            this.domicilio = domicilio;
            this.millas = millas;
        }


        public string Nombre {
            get { return nombre; }
            set { nombre = value; }
        }

        public string Domicilio {
            get { return domicilio; }
            set { domicilio = value; }
        }

        public double Millas {
            get { return millas; }
            set { millas += value; }
        }
    }
}
```

Vuelo.cs

```
namespace Reservación_Vuelos {
    public class Vuelo {

        private string Origen, Destino, Dias;
        private int BoletosVendidos, claveVuelo, numPasajeros
;
        private double Costo, Millas, Capacidad;

        public Vuelo (int claveVuelo, int numPasajeros, strin
g Origen, string Destino, string Dias, double Capacidad,
                      int BoletosVendidos, double Costo, doub
le Millas) {
            this.claveVuelo = claveVuelo;
            this.numPasajeros = numPasajeros;
            this.Origen = Origen;
            this.Destino = Destino;
            this.Dias = Dias;
            this.Capacidad = Capacidad;
            this.BoletosVendidos = BoletosVendidos;
            this.Costo = Costo;
            this.Millas = Millas;
        }

        public int pClaveVuelo {
            get { return claveVuelo; }
            set { claveVuelo = value; }
        }

        public int pNumPasajeros {
            get { return numPasajeros; }
            set { numPasajeros = value; }
        }

        public string pOrigen {
            get { return Origen; }
            set { Origen = value; }
        }

        public string pDestino {
            get { return Destino; }
            set { Destino = value; }
        }

        public string pDias {
```

```
            get { return Dias; }
            set { Dias = value; }
        }

        public double pCapacidad {
            get { return Capacidad; }
            set { Capacidad = value; }
        }

        public int pBoletosVendidos {
            get { return BoletosVendidos; }
            set {
                if (BoletosVendidos < numPasajeros)
                    BoletosVendidos += value;
                else
                    System.Console.WriteLine("\n>> YA NO HAY
ESPACIO EN ESE VUELO <<\n");
            }
        }

        public double pCosto {
            get { return Costo; }
            set { Costo = value; }
        }

        public double pMillas {
            get { return Millas; }
            set { Millas = value; }
        }
    }
}
```

ReadHelperLibrary

```
using System;
namespace ReadHelperLibrary {
    public class ReadHelper {
        public static string ReadString (string title) {
            string dato;

            do {
                Console.Write(title);
                dato = Console.ReadLine().Trim().ToUpper();
                if (dato.Equals(""))
                    Console.WriteLine("No se permiten cadenas
 vacías, teclee de nuevo.\n");
```

```
            } while (dato.Equals(""));
            return dato;
        }

        public static double ReadDouble (string title) {

            double dato = 0.0;

            bool helper;
            do {
                try {
                    Console.Write(title);
                    dato = double.Parse(Console.ReadLine());

                    if (dato <= 0)
                        Console.WriteLine("El dato no puede s
er menor o igual a cero , teclea de nuevo.");

                    helper = true;
                } catch (FormatException) {
                    helper = false;
                    Console.WriteLine("Error al castear númer
os, teclea de nuevo.");
                } catch (OverflowException) {
                    helper = false;
                    Console.WriteLine("El número es demasiado
 grande, teclea de nuevo.");
                }

            } while (dato <= 0 || !helper);


            return dato;
        }

        public static int ReadInt (string title) {
            int num = 0;
            bool helper;
            do {
                try {
                    Console.Write(title);
                    num = int.Parse(Console.ReadLine());
                    if (num <= 0)
                        Console.WriteLine("El dato no puede s
er menor o igual cero, teclea de nuevo.");
```

```
                    helper = true;
                } catch (FormatException) {
                    helper = false;
                    Console.WriteLine("Error al castear número
os, teclea de nuevo.");
                } catch (OverflowException) {
                    helper = false;
                    Console.WriteLine("El número es demasiado
 grande, teclea de nuevo.");
                }

            }
            while (num <= 0 || !helper);

            return num;
        }

        public static int ReadIntRango (string title, int val
Min, int valMax) {
            int num = 0;
            bool helper = false;
            do {
                try {
                    Console.Write(title);
                    num = int.Parse(Console.ReadLine());
                    if (num < valMin || num > valMax)
                        Console.WriteLine("El dato no está de
ntro del rango {0} - {1}. Teclee de nuevo.", valMin, valMax);
                    else
                        helper = true;
                } catch (FormatException) {
                    helper = false;
                    Console.WriteLine("Error al castear número
os, teclea de nuevo.");
                } catch (OverflowException) {
                    helper = false;
                    Console.WriteLine("El número es demasiado
 grande, teclea de nuevo.");
                }
            }
            while (!helper);

            return num;
        }
    }
}
```