Trabajo Obligatorio 2 Programacion2

Integrantes:

Christian Alico 154519

Matías Lucero 171433

Nicolas Aguirre 260084

Índice:

Caratula………………………………………………………………………Pag.1

Índice…………………………………………………………………………..Pag.2

Clase Agencia ………………………………………………………………Pag.3

Clase Compania……………………………………………………………Pag.21

Clase Compra………………………………………………………………. Pag.21

Clase Destino………………………………………………………………. Pag.22

Clase Excursion……………………………………………………………. Pag.24

Clase Internacioanl……………………………………………….…….. Pag.25

Clase Nacional……………………………………………………………… Pag.27

Clase Usuario…………….……………………………….…………………Pag.30

Datos PreCargados……………………………………………………….Pag.31

DiagramaUML………………………………………………………………Pag.33

Credenciales…………………………………………………………………Pag.33

Sitio Web publicado en somee……………………………………..Pag.34

**Clase Agencia**

using Obligatorio2Programacion2.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace obligatorioProgramacion2

{

//Creamos la clase administradora Agencia , la cual va a ser la encargada de " administrar" todas las clases del sistema

public class Agencia

{

//SINGLETON

//genero la instancia nula

private static Agencia instancia = null;

// si no existe una instancia va a crear una nueva y me la va a devolver ,

// si ya existe una instancia me va a devolver la la ya existente

public static Agencia GetInstancia()

{

if (instancia == null)

{

instancia = new Agencia();

}

return instancia;

}

//Creamos una variable booleana para saber si ya se precargaron los datos

private bool Precargado { get; set; }

//Creamos una variable privada del sistema para la cotizacion del dolar actual

private double CotizacionDelDolar { get; set; }

//Generamos las listas pertinentes para guardar los objetos en la clase sistema

private List<Excursion> listaExcursiones = new List<Excursion>();

private List<Compania> listaCompanias = new List<Compania>();

private List<Destino> listaDestinos = new List<Destino>();

private List<Usuario> listaUsuarios = new List<Usuario>();

private List<Compra> listaCompras = new List<Compra>();

private List<Usuario> listaClientes = new List<Usuario>();

//Generamos los metodos para poder acceder a las listas ( ya que se crean privadas para no modificarlas por accidente)

public double GetCotizacion()

{

return CotizacionDelDolar;

}

public bool GetPrecarga() {

return Precargado;

}

public List<Excursion> GetExcursiones()

{

listaExcursiones.Sort();

return listaExcursiones;

}

public List<Compania> GetCompanias()

{

return listaCompanias;

}

public List<Destino> GetDestinos()

{

return listaDestinos;

}

public List<Usuario> GetUsuarios()

{

return listaUsuarios;

}

public List<Compra> GetCompras() {

return listaCompras;

}

public List<Usuario> GetClientes() {

listaClientes.Sort();

return listaClientes;

}

//Generamos los metodos para agreger los objetos a las listas creadas anteriormente

//AGREGAR NACIONAL

public Excursion AgregarNacional(string descripcion, DateTime fechaComienzo, int cantDias, int lugaresDisponibles, bool interesNacional, double taza)

{

Excursion nuevo = new Nacional(descripcion, fechaComienzo, cantDias, lugaresDisponibles, interesNacional, taza);

listaExcursiones.Add(nuevo);

return nuevo;

}

//AGREGAR INTERNACIONAL

public Excursion AgregarInternacional(string descripcion, DateTime fechaComienzo, int cantDias, int lugaresDisponibles, Compania companiaAerea)

{

bool companiaExiste = false;

bool desBien = false;

bool diasBien = false;

bool lugaresBien = false;

if (descripcion.Length >= 3)

{

desBien = true;

}

if (cantDias > 0)

{

diasBien = true;

}

if (lugaresDisponibles > 0)

{

lugaresBien = true;

}

foreach (Compania c in listaCompanias)

{

if (companiaAerea == c)

{

companiaExiste = true;

}

}

if (companiaExiste && desBien && diasBien && lugaresBien) {

Excursion nuevo = new Internacional(descripcion, fechaComienzo, cantDias, lugaresDisponibles, companiaAerea);

listaExcursiones.Add(nuevo);

return nuevo;

}

else

{

return null;

}

}

//AGREGAR DESTINO

public Destino AgregarDestino(string ciudad, string pais, int costoDiario, int cantidadDeDias) {

Destino nuevo = new Destino(ciudad, pais, costoDiario, cantidadDeDias);

listaDestinos.Add(nuevo);

return nuevo;

}

//AGREGAR COMPANIA

public Compania AgregarCompania(int codigo, string pais)

{

Compania nuevo = new Compania(codigo, pais);

listaCompanias.Add(nuevo);

return nuevo;

}

//AGREGAR COMPRA

public Compra AgregarCompra(Usuario c, Excursion e, int mayores, int menores) {

int totalPasajeros;

double totalPesos;

totalPasajeros = mayores + menores;

totalPesos = e.CalcularCostoPesos() \* totalPasajeros;

totalPesos = totalPesos \* e.obtenerDescuentoORecargo();

double totalDolares;

totalDolares = e.CalcularCostoDolares() \* totalPasajeros;

totalDolares = totalDolares \* e.obtenerDescuentoORecargo();

double pasajeroPesos = e.CalcularCostoPesos();

double pasajeroDolares = e.CalcularCostoDolares();

double descAutPesos;

double descAutDolares;

descAutDolares = totalDolares - (e.CalcularCostoDolares() \* totalPasajeros);

descAutPesos = totalPesos -(e.CalcularCostoPesos()\*totalPasajeros);

Compra nuevaCompra = new Compra(c, e, mayores, menores, totalPesos, totalDolares,pasajeroPesos,pasajeroDolares, descAutPesos, descAutDolares);

listaCompras.Add(nuevaCompra);

RestarLugaresPorId(e.Id, menores + mayores);

return nuevaCompra;

}

//Generamos un metodo para cambiar la cotizacion del dolar en el sistema

public double CambiarCotizacionDelDolar(double nuevaCotizacion)

{

CotizacionDelDolar = nuevaCotizacion;

return CotizacionDelDolar;

}

//Cambiamos la variable Precarga de false a true

public void CambiarPrecarga()

{

Precargado = true;

}

//Metodo que recorre todas las excursiones del sistema y las muestra por pantalla con todos sus datos

public string MostrarExcursiones()

{

foreach (Excursion ex in listaExcursiones)

{

Console.WriteLine(ex.ToString());

}

return "=====================";

}

//Metodo que recorre todos los destinos del sistema y las muestra por pantalla con todos sus datos

public string MostrarDestinos()

{

foreach (Destino de in listaDestinos)

{

Console.WriteLine(de.ToString());

}

return "=====================";

}

//metodo que recorre las excursiones para saber si estan dentro de las fechas indicadas y van a un destino indicado

// recibe el id del destino , la fecha de inicio y finalizacion de la excursion

public List<Excursion> GetExcursionEntreFechas(int idDestino, DateTime fechaInicio, DateTime fechaFinal)

{

//creamos una lista retorno para devolver las excursiones que estan dentro de las fechas ingresadas

List<Excursion> retorno = new List<Excursion>();

//recorremos la lista de Excursiones

foreach (Excursion e in listaExcursiones)

{

foreach (Destino D in e.GetlistaDestinosDeExcursion())

// si el id del destino y el rango de fechas concuerda con alguna excursion , se guarda esa excursion en la lista retorno

if ((D.Id == idDestino) && (e.FechaComienzo >= fechaInicio) && (e.FechaComienzo <= fechaFinal))

{

retorno.Add(e);

}

}

//se devuelve la lista con las excursiones encontradas

return retorno;

}

// GET DE COMPRAS POR ID USUARIO

public List<Compra> GetComprasPorCliente(int usuario) {

List<Compra> comprasDeCliente = new List<Compra>();

foreach (Compra c in listaCompras) {

if (c.Comprador.NombreUsu == usuario && c.activa) {

comprasDeCliente.Add(c);

}

}

return comprasDeCliente;

}

//Agregar un Cliente a al Sistema

public Usuario agregarCliente(string nombre, string apellido, int cedula, string password)

{

bool nombreCheck = false;

bool apellidoCheck = false;

bool cedulaCheckLargo = false;

bool cedulaCheckNoExiste = true;

bool cedulaCheck = false;

bool passwordCheckLargo = false;

bool passwordCheckMayuscula = password.Any(c => char.IsUpper(c)); ;

bool passwordCheckMinuscula = password.Any(c => char.IsLower(c));

bool passwordCheckNumero = password.Any(c => char.IsDigit(c)); ;

bool passwordCheck = false;

if (nombre.Length > 2)

{

nombreCheck = true;

}

if (apellido.Length > 2)

{

apellidoCheck = true;

}

if (cedula > 99999 && cedula < 1000000000)

{

cedulaCheckLargo = true;

}

foreach (Usuario u in listaUsuarios)

{

if (u.Cedula == cedula)

{

cedulaCheckNoExiste = false;

}

}

if (cedulaCheckNoExiste && cedulaCheckLargo)

{

cedulaCheck = true;

}

if (password.Length > 5)

{

passwordCheckLargo = true;

}

if (passwordCheckLargo && passwordCheckMayuscula && passwordCheckMinuscula && passwordCheckNumero)

{

passwordCheck = true;

}

if (passwordCheck && nombreCheck && apellidoCheck && cedulaCheck)

{

string Nombre = nombre.Substring(0, 1).ToUpper() + nombre.Substring(1);

string Apellido = apellido.Substring(0, 1).ToUpper() + apellido.Substring(1);

Usuario Cliente = new Usuario(Nombre, Apellido, cedula, password);

listaUsuarios.Add(Cliente);

listaClientes.Add(Cliente);

return Cliente;

}

else {

return null;

}

}

//Agregar un Operador Al sistema

public Usuario agregarOperador(int nombreUsu, string password) {

Usuario Operador = new Usuario(nombreUsu, password);

listaUsuarios.Add(Operador);

return Operador;

}

//PRECARGA DE DATOS

public void PrecargaDeDatos() {

//Precarga de Destinos

Destino d1 = AgregarDestino("MONTEVIDEO", "URUGUAY", 80, 3);

Destino d2 = AgregarDestino("RIO DE JANEIRO", "BRASIL", 110, 7);

Destino d3 = AgregarDestino("AMSTERDAM", "HOLANDA", 95, 4);

Destino d4 = AgregarDestino("MOSCU", "RUSIA", 65, 6);

Destino d5 = AgregarDestino("QUITO", "ECUADOR", 55, 5);

Destino d6 = AgregarDestino("SAN JOSE", "URUGUAY", 45, 3);

Destino d7 = AgregarDestino("RIO BRANCO", "URUGUAY", 44, 8);

Destino d8 = AgregarDestino("BEIGIN", "CHINA", 70, 12);

Destino d9 = AgregarDestino("NEW YORK", "ESTADOS UNIDOS", 120, 7);

Destino d10 = AgregarDestino("ARTIGAS", "URUGUAY", 36, 6);

//Precarga de Companiasy y

Compania c1 = AgregarCompania(123, "Uruguay");

Compania c2 = AgregarCompania(234, "Brazil");

Compania c3 = AgregarCompania(345, "EE.UU");

Compania c4 = AgregarCompania(456, "Rusia");

//Precarga de Excursiones

Excursion e1 = AgregarNacional("Estadia de 3 noches en Montevideo y 3 noches en San Jose", DateTime.Parse("2020-12-24"), 6, 100, true, 0.1);

e1.AgregarDestinosExcursion(d1);

e1.AgregarDestinosExcursion(d6);

Excursion e2 = AgregarNacional("Estadia de 3 noches en Montevideo y 8 noches en Rio Branco", DateTime.Parse("2021-01-03"), 11, 1222, false, 0.2);

e2.AgregarDestinosExcursion(d1);

e2.AgregarDestinosExcursion(d7);

Excursion e3 = AgregarNacional("Estadia de 3 noches en Montevideo y 6 noches en Artigas", DateTime.Parse("2021-05-03"), 9, 300, true, 0.15);

e3.AgregarDestinosExcursion(d1);

e3.AgregarDestinosExcursion(d10);

Excursion e4 = AgregarNacional("Estadia de 3 noches en San Jose y 6 noches en Artigas", DateTime.Parse("2020-12-29"), 9, 300, false, 0.25);

e4.AgregarDestinosExcursion(d6);

e4.AgregarDestinosExcursion(d10);

Excursion e5 = AgregarInternacional("Estadia de 3 noches en Montevideo y 6 noches en Moscu", DateTime.Parse("2021-03-10"), 9, 300, c1);

e5.AgregarDestinosExcursion(d1);

e5.AgregarDestinosExcursion(d4);

Excursion e6 = AgregarInternacional("Estadia de 8 noches en Rio de Janeiro y 6 noches en Moscu", DateTime.Parse("2021-01-30"), 14, 200, c2);

e6.AgregarDestinosExcursion(d7);

e6.AgregarDestinosExcursion(d4);

Excursion e7 = AgregarInternacional("Estadia de 3 noches en Montevideo y 7 noches en New York", DateTime.Parse("2021-01-15"), 10, 150, c3);

e7.AgregarDestinosExcursion(d1);

e7.AgregarDestinosExcursion(d9);

Excursion e8 = AgregarInternacional("Estadia de 12 noches en Berlin y 6 noches en Moscu", DateTime.Parse("2020-12-30"), 18, 300, c4);

e8.AgregarDestinosExcursion(d8);

e8.AgregarDestinosExcursion(d4);

CotizacionDelDolar = 42.78;

Usuario o1 = agregarOperador(1212121, "Op123");

Usuario o2 = agregarOperador(2323232, "oP123");

Usuario cl1 = agregarCliente("Juan", "Perez", 1231231, "Jp1234");

Usuario cl2 = agregarCliente("Diego", "Peras", 1234568, "Dt1234");

Usuario cl3 = agregarCliente("Carlos", "Perez", 1234569, "Cp1234");

Usuario cl4 = agregarCliente("Carlos", "Perex", 1234566, "cP1234");

Compra co1 = AgregarCompra(cl1, e4, 2, 2);

co1.FechaCompra = DateTime.Parse("2020-10-29");

Compra co2 = AgregarCompra(cl1, e5, 2, 2);

co2.FechaCompra = DateTime.Parse("2020-09-29");

Compra co3 = AgregarCompra(cl1, e3, 2, 2);

co3.FechaCompra = DateTime.Parse("2020-11-23");

Compra co4 = AgregarCompra(cl2, e8, 3, 2);

co4.FechaCompra = DateTime.Parse("2020-07-26");

Compra co5 = AgregarCompra(cl3, e5, 3, 2);

co5.FechaCompra = DateTime.Parse("2020-08-12");

Compra co6 = AgregarCompra(cl4, e6, 3, 2);

Precargado = false;

}

// CONVERTIR INTERNACIONAL DE EXCURSION A INTERNACIONAL

public Internacional castInternacional(Excursion e) {

Internacional nuevo = (Internacional)e;

return nuevo;

}

// CONVERTIR NACIONAL DE EXCURSION A NACIONAL

public Nacional castNacional(Excursion e) {

Nacional nuevo = (Nacional)e;

return nuevo;

}

// ENCONTRAR EXCURSION POR ID DE EXCURSION

public Excursion GetExcursionPorId(int id) {

Excursion nueva;

foreach (Excursion e in listaExcursiones) {

if (e.Id == id) {

nueva = e;

return nueva;

}

}

return null;

}

// ENCONTRRAR USUARIO POR NOMBRE DE USUARIO

public Usuario GetUsuarioPorNombreUsu(int id) {

foreach (Usuario u in listaUsuarios) {

if (u.NombreUsu == id)

{ return u;

}

}

return null;

}

// BAJAR STOCK DE UNA EXCURSION DADO SU ID Y LA CANTIDAD DE STOCK

public void RestarLugaresPorId(int id, int cantidad) {

Excursion e = GetExcursionPorId(id);

e.LugaresDisponibles = e.LugaresDisponibles - cantidad;

}

//AGREGAR STOCK A UNA EXCURSION DADO SU ID Y LA CANTIDAD DE STOCK

public void SumarLugaresPorId(int id, int cantidad)

{

Excursion e = GetExcursionPorId(id);

e.LugaresDisponibles = e.LugaresDisponibles + cantidad;

}

//ENCONTRAR COMPRA DADO UN ID

public Compra GetComprasPorId(int id) {

foreach (Compra c in listaCompras) {

if (c.Id == id) {

return c;

}

}

return null;

}

// CANCELAR COMPRA DADO UN ID

public void cancelarCompra(int id) {

Compra c = GetComprasPorId(id);

c.activa = false;

int totalLugares = c.CantidadPersonasMayores + c.CantidadPersonasMenores;

SumarLugaresPorId(c.ExcursionComprada.Id, totalLugares);

}

// ENCONTRAR TODAS LAS COMPRAS QUE NO FUERON CANCELADAS

public List<Compra> GetComprasActivas() {

List<Compra> comprasActivas = new List<Compra>();

foreach (Compra c in listaCompras) {

if (c.activa == true)

{

comprasActivas.Add(c);

}

}

return comprasActivas;

}

//ENCONTRAR UNA COMPRA ENTRE DOS FECHAS

public List<Compra> GetCEntreFechas(DateTime fechaI, DateTime fechaF) {

List<Compra> comprasEntreFechas = new List<Compra>();

foreach (Compra c in listaCompras)

{

if (c.FechaCompra>fechaI && c.FechaCompra<fechaF)

{

comprasEntreFechas.Add(c);

}

}

return comprasEntreFechas;

}

//OBTENER LOS DESTINOS MAS USADOS

public List<Destino> GetDestinoMasUsado() {

List<Destino> destinosMasUsados = new List<Destino>();

int mayorCantDestinos = int.MinValue;

foreach (Destino d in listaDestinos) {

bool vuelta = false;

if (d.ExcursionesDeEsteDestino.Count > mayorCantDestinos) {

destinosMasUsados.Clear();

destinosMasUsados.Add(d);

mayorCantDestinos = d.ExcursionesDeEsteDestino.Count;

vuelta = true;

}

if (d.ExcursionesDeEsteDestino.Count == mayorCantDestinos && vuelta==false ) {

destinosMasUsados.Add(d);

}

}

return destinosMasUsados ;

;

}

//OBTENER UNA EXCURSION DADO UN DESTINO

public List<Excursion> GetExcursionesPorDestino(int id) {

List<Excursion> excursionesPorDestino = new List<Excursion>();

foreach (Excursion e in listaExcursiones) {

foreach (Destino de in e.GetlistaDestinosDeExcursion()) {

if (de.Id == id) {

excursionesPorDestino.Add(e);

}

}

}

return excursionesPorDestino;

}

}

}

**Clase Compania**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace obligatorioProgramacion2

{

//Generamos la clase publica Compania

public class Compania

{

//Creamos las propiedades de la clase Compania

public int Codigo { get; set; }

public string Pais { get; set; }

public Compania(int codigo, string pais)

{

Codigo = codigo;

Pais = pais;

}

//reescribimos el metodo ToString() para mostrar en pantalla el objeto cuando nos sea necesario

public override string ToString()

{

return $"Codigo: {Codigo} Pais: {Pais}";

}

}

}

**Clase Compra**

using obligatorioProgramacion2;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Obligatorio2Programacion2.Models

{

public class Compra

{

public int Id { get; set; }

protected static int UltimoId = 0;

public Usuario Comprador { get; set; }

public Excursion ExcursionComprada { get; set; }

public int CantidadPersonasMayores { get; set; }

public int CantidadPersonasMenores { get; set; }

public double CostoPesosPasajero { get; set; }

public double CostoDolaresPasajero { get; set; }

public double costoTotalPesos { get; set; }

public double costoTotalDolares{ get; set; }

public bool activa { get; set; }

public DateTime FechaCompra { get; set; }

public int TotalPasajeros { get; set; }

public double DescuentoAumentoPesos { get; set; }

public double DescuentoAumentoDolares { get; set; }

public Compra(){}

public Compra(Usuario comprador, Excursion excursionComprada, int cantidadPersonasMayores, int cantidadPersonasMenores,double pesos,double dolares,double costoPesosPasajero, double costoDolaresPasajero, double descAutPesos,double descAutoDolares)

{

Id = UltimoId;

UltimoId = UltimoId + 1;

Comprador = comprador;

ExcursionComprada = excursionComprada;

CantidadPersonasMayores = cantidadPersonasMayores;

CantidadPersonasMenores = cantidadPersonasMenores;

costoTotalPesos = pesos;

costoTotalDolares = dolares;

activa = true;

FechaCompra = DateTime.Today;

TotalPasajeros = cantidadPersonasMayores + cantidadPersonasMenores;

CostoPesosPasajero =costoPesosPasajero;

CostoDolaresPasajero =costoDolaresPasajero;

DescuentoAumentoPesos = descAutPesos;

DescuentoAumentoDolares = descAutoDolares;

}

}

}

**Clase Destino**

using System;

using System.Collections.Generic;

using Obligatorio2Programacion2.Models;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace obligatorioProgramacion2

{

//Creamos la clase Destino publica para que se pueda acceder desde todas partes del programa

public class Destino

{

private readonly Agencia a = Agencia.GetInstancia();

//Creamos las propiedades de la clase Destino

public int Id { get; set; }

//creamos un contador interno para que lleve cada destino lleve un Id unico y sea mas facil de encontrarlo a la hora de buscarlo

private static int UltimoId =1 ;

public string Ciudad { get; set; }

public string Pais { get; set; }

public int CostoDiario { get; set; }

public int CantidadDeDias { get; set; }

public List<Excursion> ExcursionesDeEsteDestino = new List<Excursion>();

//Generamos los constructores de la clase Destino sin y con parametros

public Destino()

{

}

public Destino(string ciudad, string pais, int costoDiario, int cantidadDeDias)

{

Id = UltimoId;

UltimoId++;

Ciudad = ciudad;

Pais = pais;

CostoDiario = costoDiario;

CantidadDeDias = cantidadDeDias;

}

public override string ToString()

{

return $@"

-Id: {Id}

-Ciudad: {Ciudad}

-Pais: {Pais}

-Costo por dia: {CostoDiario}

-Cantidad de dias: {CantidadDeDias}

-Costo total del destino: {CostoEnDolares()} USD |o| {CostoEnPesos()}

";

}

public int CostoEnDolares()

{

return CostoDiario\*CantidadDeDias;

}

public double CostoEnPesos()

{

double totalEnPesos;

totalEnPesos =Convert.ToDouble(CantidadDeDias \* CostoDiario);

double cotizacion = a.GetCotizacion();

totalEnPesos = totalEnPesos \* cotizacion;

return totalEnPesos;

}

public List<Excursion> AgregarExcursionDestinos(int id)

{

foreach (Excursion e in a.GetExcursiones()) {

if (e.Id == id) {

ExcursionesDeEsteDestino.Add(e);

}

}

return ExcursionesDeEsteDestino;

}

}

}

**Clase Excursion**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace obligatorioProgramacion2

{

//Generamos la clase padre Excursion publica y abstracta para que sus hijos( Nacional e Internacional) puedan heredar tanto sus propiedasdes como sus metodos

public abstract class Excursion :IComparable<Excursion>

{

//Creamos las propiedades de la clase Excursion ( estas son las que se comparten entre Nacional e Internacional ) para que sus clases hijas puedan heredarlas

public int Id { get; set; }

public string Descripcion { get; set; }

public DateTime FechaComienzo { get; set; }

public int CantDias { get; set; }

public int LugaresDisponibles { get; set; }

public double Taza { get; set; }

public string Tipo { get; set; }

//Creamos un contador incremental para los numeros de excursion

protected static int UltimoId = 1000;

//Generamos una lista de destino propia para cada excursion para que sus clases hijas las hereden

protected List<Destino> listaDestinosDeExcursion = new List<Destino>();

public List<Destino> GetlistaDestinosDeExcursion()

{

return listaDestinosDeExcursion;

}

public abstract List<Destino> AgregarDestinosExcursion(Destino d);

//creamos los metodos que compartiran los hijos de forma abstracta para que tengan que si o si usarlos

public abstract override string ToString();

public abstract double CalcularCostoDolares();

public abstract double CalcularCostoPesos();

public abstract double obtenerDescuentoORecargo();

public int CompareTo(Excursion other)

{

if (this.FechaComienzo.CompareTo(other.FechaComienzo) < 0)

{

return 1;

}

else if (this.FechaComienzo.CompareTo(other.FechaComienzo) > 0)

{

return -1;

}

else

{

return 0;

}

}

}

}

**Clase Internacioanl**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace obligatorioProgramacion2

{

//Generamos la clase Internacional Hija de la clase Excursion

public class Internacional : Excursion

{

private readonly Agencia a = Agencia.GetInstancia();

//Creamos las propiedades de la clase Internacional

public Compania CompaniaAerea { get; set; }

public string ciudades;

public int totalDias;

// Generamos el constructor de la clase Internacional con parametros

public Internacional( string descripcion, DateTime fechaComienzo, int cantDias, int lugaresDisponibles, Compania companiaAerea)

{

Id = UltimoId;

UltimoId = UltimoId+100;

Taza = 1.1;

Descripcion = descripcion;

FechaComienzo = fechaComienzo;

CantDias = cantDias;

LugaresDisponibles = lugaresDisponibles;

CompaniaAerea= companiaAerea;

Tipo = "Internacional";

}

public Internacional()

{

}

// Usamos los metodos heredados de la clase padre Excursion y los " sobreescribimos " con el atributo override

public override List<Destino> AgregarDestinosExcursion(Destino d)

{

listaDestinosDeExcursion.Add(d);

d.AgregarExcursionDestinos(Id);

return listaDestinosDeExcursion;

}

public override string ToString()

{

ciudades = "";

totalDias = CantDias;

//recorremos los destinos de la excursion para saber por que ciudades pasa , concatenando las mismas en un string

// y el acumulado de dias por destino

foreach (Destino d in listaDestinosDeExcursion)

{

totalDias = totalDias + d.CantidadDeDias ;

ciudades = ciudades + d.Ciudad+"("+d.Pais+") , ";

}

// ya que el destino terminaria en " , " borramos los ultimos 3 caracteres del string para no mostrar de mas en pantalla

ciudades = ciudades.Substring(0, ciudades.Length - 3);

return $@"

Excursion

########################################################

-ID: {Id}

-Descripcion: {Descripcion}

-Fecha de Comienzo: {FechaComienzo}

-Lugares disponibles: {LugaresDisponibles}

-Cantidad de dias de traslado: {CantDias}

-Total de dias de la excurcion : {totalDias}

-Costo total de la excursion : {CalcularCostoDolares()} USD |o| {CalcularCostoPesos()} Pesos

-Compania Aerea: {CompaniaAerea.ToString()}

-Destinos: {ciudades}

";

}

//usamos el metodo heredado que sobreescribimos para calcular el costo en pesos de la excursion

public override double CalcularCostoDolares()

{

//creamos una variable en donde vamos a guardar el acumulado de los destinos

int costoDolares = 0;

foreach (Destino d in listaDestinosDeExcursion)

{

//recorremos cada destino y multiplicamos el costo diario por la cantidad de dias

costoDolares = costoDolares + (d.CostoDiario \* d.CantidadDeDias);

}

//devolvemos el costo total en dolares de la excursion

return costoDolares;

}

//hacemos lo mismo que en el metodo anterior

public override double CalcularCostoPesos()

{

//creamos una variable en donde guardamos el acumulado en dolares

int costoPesos = 0;

double costoTotalPesos;

foreach (Destino d in listaDestinosDeExcursion)

{

// calculamos el costo por destino multiplicando la cantidad de dias del mismo por el precio diario del mismo

costoPesos = costoPesos + (d.CostoDiario \* d.CantidadDeDias);

}

//creamos una variable para importar la cotizacion de la clase Agencia( clase administradora)

double cotizacion = a.GetCotizacion();

//conevertimos el tipo de variable int en double para que no genere problemas entre las operaciones de variables

costoTotalPesos = Convert.ToDouble(costoPesos);

costoTotalPesos = costoTotalPesos \* cotizacion;

//retornamos el costo total en pesos de la excursion

return costoTotalPesos;

}

public override double obtenerDescuentoORecargo()

{

return Taza;

}

}

}

**Clase Nacional**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace obligatorioProgramacion2

{

//Generamos la clase Nacional Hija de la clase Excursion

public class Nacional :Excursion

{

private readonly Agencia a = Agencia.GetInstancia();

//Creamos las propiedades de la clase Nacional

public bool InteresNacional { get; set; }

public string MSJ;

public string ciudades;

public double totalDias;

// Generamos el constructor de la clase Nacional con parametros

public Nacional()

{

}

public Nacional( string descripcion, DateTime fechaComienzo, int cantDias, int lugaresDisponibles,bool interesNacional, double taza)

{

Id = UltimoId;

UltimoId=UltimoId+100;

Descripcion = descripcion;

FechaComienzo = fechaComienzo;

CantDias = cantDias;

LugaresDisponibles = lugaresDisponibles;

InteresNacional = interesNacional;

Taza = taza;

Tipo = "Nacional";

}

// Usamos los metodos heredados de la clase padre Excursion y los " sobreescribimos " con el atributo override

public override List<Destino> AgregarDestinosExcursion(Destino d)

{

listaDestinosDeExcursion.Add(d);

d.AgregarExcursionDestinos(Id);

return listaDestinosDeExcursion;

}

public override string ToString()

{

totalDias= CantDias;

ciudades = "";

//elegimos el mensaje a mostrar deacuerdo a la excursion nacional si es de interes o no

if (InteresNacional == true)

{

MSJ = "Es de interes nacional por el ministerio de turismo";

}

if(InteresNacional == false)

{

MSJ = "No es de interes nacional por el ministerio de turismo";

}

//recorremos los destinos de la excursion para saber por que ciudades pasa , concatenando las mismas en un string

foreach (Destino d in listaDestinosDeExcursion)

{

totalDias = totalDias + d.CantidadDeDias ;

ciudades = ciudades +d.Ciudad+ " , ";

}

// ya que el destino terminaria en " , " borramos los ultimos 3 caracteres del string para no mostrar de mas en pantalla

ciudades = ciudades.Substring(0, ciudades.Length - 3);

return $@"

Excursion

########################################################

-ID: {Id}

-Descripcion: {Descripcion}

-Fecha de Comienzo: {FechaComienzo}

-Lugares disponibles: {LugaresDisponibles}

-Cantidad de dias de traslado: {CantDias}

-Total de dias de la excurcion : {totalDias}

-Costo total de la excursion : {CalcularCostoDolares()} USD |o| {CalcularCostoPesos()} Pesos

-{MSJ}

-Destinos: {ciudades}

";

}

//usamos el metodo heredado que sobreescribimos para calcular el costo en pesos de la excursion

public override double CalcularCostoDolares()

{

//creamos una variable en donde vamos a guardar el acumulado de los destinos

int costoDolares = 0;

foreach (Destino d in listaDestinosDeExcursion)

{

//recorremos cada destino y multiplicamos el costo diario por la cantidad de dias

costoDolares = costoDolares + (d.CostoDiario \* d.CantidadDeDias);

}

//devolvemos el costo total en dolares de la excursion

return costoDolares;

}

//hacemos lo mismo que en el metodo anterior

public override double CalcularCostoPesos()

{

//creamos una variable en donde guardamos el acumulado en dolares

int costoPesos = 0;

double costoTotalPesos;

foreach (Destino d in listaDestinosDeExcursion)

{

// calculamos el costo por destino multiplicando la cantidad de dias del mismo por el precio diario del mismo

costoPesos = costoPesos + (d.CostoDiario \* d.CantidadDeDias);

}

//creamos una variable para importar la cotizacion de la clase Agencia( clase administradora)

double cotizacion = a.GetCotizacion();

//conevertimos el tipo de variable int en double para que no genere problemas entre las operaciones de variables

costoTotalPesos = Convert.ToDouble(costoPesos);

costoTotalPesos = costoTotalPesos \* cotizacion;

//retornamos el costo total en pesos de la excursion

return costoTotalPesos;

}

public override double obtenerDescuentoORecargo()

{

if (FechaComienzo.Month > 04 && FechaComienzo.Month < 09)

{

return Taza;

}

else {

return 1;

}

}

}

}

**Clase Usuario**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Obligatorio2Programacion2.Models

{

public class Usuario: IComparable<Usuario>

{

public string Nombre { get; set; }

public string Apellido { get; set; }

public int Cedula { get; set; }

public int NombreUsu { get; set; }

public string Password { get; set; }

public string Rol { get; set; }

public Usuario(string nombre, string apellido, int cedula, string password) {

Nombre = nombre;

Apellido = apellido;

Cedula = cedula;

NombreUsu = cedula;

Password = password;

Rol = "Cliente";

}

public Usuario() { }

public Usuario(int nombreUsu, string password) {

NombreUsu = nombreUsu;

Password = password;

Rol = "Operador";

}

public int CompareTo(Usuario other)

{

if (this.Apellido.CompareTo(other.Apellido) > 0) {

return 1;

}

else if (this.Apellido.CompareTo(other.Apellido) < 0) {

return -1;

}

else {

if (this.Nombre.CompareTo(other.Nombre) > 0) {

return 1; }

else if (this.Nombre.CompareTo(other.Nombre) < 0) {

return -1; }

else {

return 0;

}

}

}

}

}

**Datos PreCargados**

public void PrecargaDeDatos() {

//Precarga de Destinos

Destino d1 = AgregarDestino("MONTEVIDEO", "URUGUAY", 80, 3);

Destino d2 = AgregarDestino("RIO DE JANEIRO", "BRASIL", 110, 7);

Destino d3 = AgregarDestino("AMSTERDAM", "HOLANDA", 95, 4);

Destino d4 = AgregarDestino("MOSCU", "RUSIA", 65, 6);

Destino d5 = AgregarDestino("QUITO", "ECUADOR", 55, 5);

Destino d6 = AgregarDestino("SAN JOSE", "URUGUAY", 45, 3);

Destino d7 = AgregarDestino("RIO BRANCO", "URUGUAY", 44, 8);

Destino d8 = AgregarDestino("BEIGIN", "CHINA", 70, 12);

Destino d9 = AgregarDestino("NEW YORK", "ESTADOS UNIDOS", 120, 7);

Destino d10 = AgregarDestino("ARTIGAS", "URUGUAY", 36, 6);

//Precarga de Companiasy y

Compania c1 = AgregarCompania(123, "Uruguay");

Compania c2 = AgregarCompania(234, "Brazil");

Compania c3 = AgregarCompania(345, "EE.UU");

Compania c4 = AgregarCompania(456, "Rusia");

//Precarga de Excursiones

Excursion e1 = AgregarNacional("Estadia de 3 noches en Montevideo y 3 noches en San Jose", DateTime.Parse("2020-12-24"), 6, 100, true, 0.1);

e1.AgregarDestinosExcursion(d1);

e1.AgregarDestinosExcursion(d6);

Excursion e2 = AgregarNacional("Estadia de 3 noches en Montevideo y 8 noches en Rio Branco", DateTime.Parse("2021-01-03"), 11, 1222, false, 0.2);

e2.AgregarDestinosExcursion(d1);

e2.AgregarDestinosExcursion(d7);

Excursion e3 = AgregarNacional("Estadia de 3 noches en Montevideo y 6 noches en Artigas", DateTime.Parse("2021-05-03"), 9, 300, true, 0.15);

e3.AgregarDestinosExcursion(d1);

e3.AgregarDestinosExcursion(d10);

Excursion e4 = AgregarNacional("Estadia de 3 noches en San Jose y 6 noches en Artigas", DateTime.Parse("2020-12-29"), 9, 300, false, 0.25);

e4.AgregarDestinosExcursion(d6);

e4.AgregarDestinosExcursion(d10);

Excursion e5 = AgregarInternacional("Estadia de 3 noches en Montevideo y 6 noches en Moscu", DateTime.Parse("2021-03-10"), 9, 300, c1);

e5.AgregarDestinosExcursion(d1);

e5.AgregarDestinosExcursion(d4);

Excursion e6 = AgregarInternacional("Estadia de 8 noches en Rio de Janeiro y 6 noches en Moscu", DateTime.Parse("2021-01-30"), 14, 200, c2);

e6.AgregarDestinosExcursion(d7);

e6.AgregarDestinosExcursion(d4);

Excursion e7 = AgregarInternacional("Estadia de 3 noches en Montevideo y 7 noches en New York", DateTime.Parse("2021-01-15"), 10, 150, c3);

e7.AgregarDestinosExcursion(d1);

e7.AgregarDestinosExcursion(d9);

Excursion e8 = AgregarInternacional("Estadia de 12 noches en Berlin y 6 noches en Moscu", DateTime.Parse("2020-12-30"), 18, 300, c4);

e8.AgregarDestinosExcursion(d8);

e8.AgregarDestinosExcursion(d4);

CotizacionDelDolar = 42.78;

Usuario o1 = agregarOperador(1212121, "Op123");

Usuario o2 = agregarOperador(2323232, "oP123");

Usuario cl1 = agregarCliente("Juan", "Perez", 1231231, "Jp1234");

Usuario cl2 = agregarCliente("Diego", "Peras", 1234568, "Dt1234");

Usuario cl3 = agregarCliente("Carlos", "Perez", 1234569, "Cp1234");

Usuario cl4 = agregarCliente("Carlos", "Perex", 1234566, "cP1234");

Compra co1 = AgregarCompra(cl1, e4, 2, 2);

co1.FechaCompra = (DateTime.Parse("2020-10-29"));

Compra co2 = AgregarCompra(cl1, e5, 2, 2);

co2.FechaCompra = (DateTime.Parse("2020-09-29"));

Compra co3 = AgregarCompra(cl1, e3, 2, 2);

co3.FechaCompra = (DateTime.Parse("2020-11-23"));

Compra co4 = AgregarCompra(cl2, e8, 3, 2);

co4.FechaCompra = (DateTime.Parse("2020-07-26"));

Compra co5 = AgregarCompra(cl3, e5, 3, 2);

co5.FechaCompra = (DateTime.Parse("2020-08-12"));

Compra co6 = AgregarCompra(cl4, e6, 3, 2);

Precargado = false;

}

**Diagrama UML**

**Diagrama, Escala de tiempo

Descripción generada automáticamente**

**Credenciales**

**Cliente1 : Nombre de usuario: 1231231 Password: Jp1234**

**Cliente2 : Nombre de usuario:** **1234568 Password:Dt1234**

**Cliente3 : Nombre de usuario: 1234569 Password: Cp1234**

**Cliente4 : Nombre de usuario: 1234566 Password: cP1234**

**Operador1:Nombre de usuario:1212121 Password:Op123**

**Operador2: Nombre de usuario:2323232 Password:oP123**

**SitioWeb publicado en somee:**

http://Obligat2077531927orioProgramacion2.somee.com