//

//

// Houseboat Rental Quote Program

//

//

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace HouseboatRentalQuote

{

 public partial class RentalQuoteForm : Form

 {

 const double BASE\_PRICE = 309.99; // $309.99

 const double ENGINE\_UPGRADE\_RATE = .13; //13%

 const short BASE\_PASSENGERS\_LIMIT = 3;

 const double EXTRA\_PASSENGERS\_RATE = .095; // 9.5%

 const int BASE\_HOURS\_LIMIT = 24;

 const double EXTRA\_HOURS\_RATE = 12.0; // $12.00

 const double UPRIVER\_CHARGE\_RATE = 50.0; // $50.00 (per hop)

 const double DOWNRIVER\_DISCOUNT\_RATE = 30.0; // $30.00 (per hop)

 const double TAX\_RATE = .0605; // 6.05%

 int hoursSpan;

 int daysSpan;

 int hoursTotal;

 int hops;

 bool isUpriver;

 DateTime preparedDate = new DateTime();

 DateTime departDateTime;

 DateTime arriveDateTime;

 short passengersTotal;

 const byte PM\_NEEDS\_LICENSE = 21; // 24-hour time for 9:00 pm

 const byte AM\_NEEDS\_LICENSE = 7; // 24-hour time for 7:00 am;

 const byte ENGINE\_NEEDS\_LICENSE = 3;

 const byte FIELD\_SIZE = 28;

 Quote newQuote = new Quote();

 public RentalQuoteForm()

 {

 InitializeComponent();

 }

 private void RentalQuoteForm\_Load(object sender, EventArgs e)

 {

 departTimeComboBox.SelectedIndex = 8;

 arriveTimeComboBox.SelectedIndex = departTimeComboBox.SelectedIndex;

 engineLevelsListBox.SelectedIndex = 0;

 taxesTagLbl.Text = "Taxes at " + (TAX\_RATE).ToString("p") + ":";

 }

 private void departDateDateTimePicker\_ValueChanged(object sender, EventArgs e)

 {

 if (departDateDateTimePicker.Value < DateTime.Now)

 MessageBox.Show(" Invalid Departure Date.\nDeparture date must be later than\nor equal to the current date.");

 else

 CheckForLicenseNeed();

 }

 private void departTimeComboBox\_SelectedIndexChanged(object sender, EventArgs e)

 {

 CheckForLicenseNeed();

 }

 private void arriveDateDateTimePicker\_ValueChanged(object sender, EventArgs e)

 {

 if (arriveDateDateTimePicker.Value < DateTime.Now)

 {

 MessageBox.Show(" Invalid Arrival Date.\nArrival date must be later than\nor equal to the current date.");

 arriveDateDateTimePicker.Value = departDateDateTimePicker.Value;

 }

 else

 CheckForLicenseNeed();

 }

 private void arriveTimeComboBox\_SelectedIndexChanged(object sender, EventArgs e)

 {

 CheckForLicenseNeed();

 }

 private void engineLevelsListBox\_SelectedIndexChanged(object sender, EventArgs e)

 {

 CheckForLicenseNeed();

 }

 private void CheckForLicenseNeed()

 {

 if (departTimeComboBox.SelectedIndex >= PM\_NEEDS\_LICENSE ||

 departTimeComboBox.SelectedIndex <= AM\_NEEDS\_LICENSE ||

 arriveTimeComboBox.SelectedIndex >= PM\_NEEDS\_LICENSE ||

 arriveTimeComboBox.SelectedIndex <= AM\_NEEDS\_LICENSE ||

 engineLevelsListBox.SelectedIndex >= 3

 )

 {

 warningLbl.Visible = true;

 }

 else

 warningLbl.Visible = false;

 }

 #region Passengers Radio Buttons Methods

 private void passengersRadioButton1\_CheckedChanged(object sender, EventArgs e)

 {

 passengersTotal = 1;

 }

 private void passengersRadioButton2\_CheckedChanged(object sender, EventArgs e)

 {

 passengersTotal = 2;

 }

 private void passengersRadioButton3\_CheckedChanged(object sender, EventArgs e)

 {

 passengersTotal = 3;

 }

 private void passengersRadioButton4\_CheckedChanged(object sender, EventArgs e)

 {

 passengersTotal = 4;

 }

 private void passengersRadioButton5\_CheckedChanged(object sender, EventArgs e)

 {

 passengersTotal = 5;

 }

 private void passengersRadioButton6\_CheckedChanged(object sender, EventArgs e)

 {

 passengersTotal = 6;

 }

 #endregion

 private void calculateQuoteBtn\_Click(object sender, EventArgs e)

 {

 if (departTownComboBox.SelectedItem == null)

 MessageBox.Show("Please select a town for departure.");

 else if (arriveTownComboBox.SelectedItem == null)

 MessageBox.Show("Please select a town for arrival.");

 else

 {

 int year;

 int month;

 int day;

 int hour;

 year = departDateDateTimePicker.Value.Year;

 month = departDateDateTimePicker.Value.Month;

 day = departDateDateTimePicker.Value.Day;

 hour = departTimeComboBox.SelectedIndex;

 departDateTime = new DateTime(year, month, day, hour, 00, 00);

 year = arriveDateDateTimePicker.Value.Year;

 month = arriveDateDateTimePicker.Value.Month;

 day = arriveDateDateTimePicker.Value.Day;

 hour = arriveTimeComboBox.SelectedIndex;

 arriveDateTime = new DateTime(year, month, day, hour, 00, 00);

 TimeSpan timeSpan = arriveDateTime - departDateTime;

 hoursSpan = timeSpan.Hours;

 daysSpan = timeSpan.Days;

 hoursTotal = hoursSpan + daysSpan \* 24;

 if (hoursTotal < 1)

 MessageBox.Show(" Invalid Arrival date or time.\nThe Arrival date and time must be later\nthan the Departure date and time.");

 else if (departDateTime < DateTime.Now)

 MessageBox.Show(" Departure date or time is invalid.\nThe Departure date and time must be \nlater than the current date and time.");

 else

 {

 preparedDate = DateTime.Now;

 PopulateQuote(); // populate the quote instance for passing back to the resevation system.

 DisplayQuote();

 }

 }

 }

 private void PopulateQuote()

 {

 newQuote.PreparedDate = preparedDate;

 newQuote.BasePrice = BASE\_PRICE;

 newQuote.DepartTown = departTownComboBox.SelectedItem.ToString();

 newQuote.DepartDateTime = departDateTime;

 newQuote.ArriveTown = arriveTownComboBox.SelectedItem.ToString();

 newQuote.ArriveDateTime = arriveDateTime;

 newQuote.TotalRentalHours = hoursTotal;

 newQuote.SubTotal = newQuote.BasePrice; // Price components (if expressed in percentage, calculate based on the RUNNING total)

 newQuote.EngineIndex = (byte)engineLevelsListBox.SelectedIndex;

 newQuote.EngineLevel = engineLevelsListBox.SelectedItem.ToString();

 newQuote.EngineUpgradeCharge = (ENGINE\_UPGRADE\_RATE \* newQuote.EngineIndex) \* newQuote.SubTotal;

 newQuote.SubTotal += newQuote.EngineUpgradeCharge;

 if (newQuote.TotalRentalHours > BASE\_HOURS\_LIMIT)

 newQuote.ExtendedHoursCharge = (newQuote.TotalRentalHours - BASE\_HOURS\_LIMIT) \* EXTRA\_HOURS\_RATE;

 else

 newQuote.ExtendedHoursCharge = 0;

 newQuote.SubTotal += newQuote.ExtendedHoursCharge;

 newQuote.PassengersTotal = passengersTotal;

 if (passengersTotal > BASE\_PASSENGERS\_LIMIT)

 newQuote.ExtraPassengersCharge = ((passengersTotal - BASE\_PASSENGERS\_LIMIT) \* EXTRA\_PASSENGERS\_RATE) \* newQuote.SubTotal;

 else

 newQuote.ExtraPassengersCharge = 0;

 newQuote.SubTotal += newQuote.ExtraPassengersCharge;

 SetIsUpriver();

 hops = Math.Abs(departTownComboBox.SelectedIndex - arriveTownComboBox.SelectedIndex);

 if (hops == 0)

 {

 newQuote.UpstreamCharge = 0;

 newQuote.DownstreamDiscount = 0;

 }

 else if (isUpriver)

 {

 newQuote.UpstreamCharge = hops \* UPRIVER\_CHARGE\_RATE;

 newQuote.DownstreamDiscount = 0;

 }

 else

 {

 newQuote.UpstreamCharge = 0;

 newQuote.DownstreamDiscount = hops \* DOWNRIVER\_DISCOUNT\_RATE;

 }

 newQuote.SubTotal += newQuote.UpstreamCharge;

 newQuote.SubTotal -= newQuote.DownstreamDiscount;

 newQuote.TaxRate = TAX\_RATE;

 newQuote.Tax = newQuote.SubTotal \* newQuote.TaxRate;

 newQuote.GrandTotal = newQuote.SubTotal + newQuote.Tax;

 newQuote.LicenseNeeded = warningLbl.Visible;

 }

 private void SetIsUpriver()

 {

 if (departTownComboBox.SelectedIndex < arriveTownComboBox.SelectedIndex)

 isUpriver = false;

 else

 isUpriver = true;

 }

 private void DisplayQuote()

 {

 datePreparedLbl.Text = String.Format("(Prepared on {0}.)", newQuote.PreparedDate.ToString("MMMM d', 'yyyy' at 'hh':00 'tt"));

 departOutputLbl.Text = String.Format("{0} on {1} at {2}.",

 newQuote.DepartTown,

 newQuote.DepartDateTime.ToString("MMMM d', 'yyyy"),

 newQuote.DepartDateTime.ToString("hh':00 'tt"));

 arriveOutputLbl.Text = String.Format("{0} on {1} at {2}.",

 newQuote.ArriveTown,

 newQuote.ArriveDateTime.ToString("MMMM d', 'yyyy"),

 newQuote.ArriveDateTime.ToString("hh':00 'tt"));

 totalRentalHoursLbl.Text = "For a total of " + newQuote.TotalRentalHours + " hours.";

 basePriceOutputLbl.Text = newQuote.BasePrice.ToString("c2");

 ExtendedHoursOutputLbl.Text = newQuote.ExtendedHoursCharge.ToString("c2");

 ExtraPassengersOutputLbl.Text = newQuote.ExtraPassengersCharge.ToString("c2");

 upstreamChargeOutputLbl.Text = newQuote.UpstreamCharge.ToString("c2");

 engineLevelOutputLbl.Text = newQuote.EngineUpgradeCharge.ToString("c2");

 downstreamDiscountOutputLbl.Text = newQuote.DownstreamDiscount.ToString("c2");

 TaxOutputLbl.Text = newQuote.Tax.ToString("c2");

 grandTotalOutputLbl.Text = newQuote.GrandTotal.ToString("c2");

 warningDisplayLbl.Visible = warningLbl.Visible;

 }

 } // end RentalQuoteForm class

 //

 //

 /\*

 create a class from which to instantiate an object that holds all the quote data and can be passed to the reservation system that is using this application (from Asssigment Agenda, paragraph 1).

 \*/

 //

 class Quote

 {

 private DateTime preparedDate { get; set; }

 public string DepartTown { get; set; }

 private DateTime departDateTime;

 public string ArriveTown { get; set; }

 private DateTime arriveDateTime = new DateTime();

 public int TotalRentalHours { get; set; }

 public double BasePrice { get; set; }

 public double ExtendedHoursCharge { get; set;}

 public short PassengersTotal { get; set; }

 public double ExtraPassengersCharge { get; set; }

 public byte EngineIndex { get; set; }

 public string EngineLevel { get; set; }

 public double EngineUpgradeCharge {get; set;}

 public double UpstreamCharge { get; set; }

 public double DownstreamDiscount { get; set; }

 public bool LicenseNeeded { get; set; }

 public double SubTotal { get; set; }

 public double TaxRate { get; set; }

 public double Tax { get; set; }

 public double GrandTotal { get; set; }

 public DateTime PreparedDate

 {

 get { return preparedDate; }

 set

 {

 preparedDate = value;

 }

 }

 public DateTime DepartDateTime

 {

 get { return departDateTime; }

 set

 {

 departDateTime = value;

 }

 }

 public DateTime ArriveDateTime

 {

 get { return arriveDateTime; }

 set

 {

 arriveDateTime = value;

 }

 }

 public Quote()

 {

 // set defaults for possible unused variables

 ExtendedHoursCharge = 0;

 ExtraPassengersCharge = 0;

 UpstreamCharge = 0;

 DownstreamDiscount = 0;

 EngineUpgradeCharge = 0;

 }

 } // end Quote class

} // end HouseboatRentalQuote namespace