

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 Assignment01

{

public partial class Rentals : Form

{

// Classwide objects for validating the fields

ErrorProvider ErrorProvider1 = new ErrorProvider();

Decimal NumberDecimal;

public Rentals()

{

InitializeComponent();

}

private void OdometerBeginningTextBox\_Validating(object sender, CancelEventArgs e)

{

Decimal.TryParse(OdometerBeginningTextBox.Text, out NumberDecimal);

if (NumberDecimal == 0)

{

ErrorProvider1.SetError(OdometerBeginningTextBox, "Entry required!");

OdometerBeginningTextBox.Focus();

e.Cancel = true;

}

else

ErrorProvider1.Clear();

}

private void OdometerEndingTextBox\_Validating(object sender, CancelEventArgs e)

{

Decimal.TryParse(OdometerEndingTextBox.Text, out NumberDecimal);

if (NumberDecimal == 0)

{

ErrorProvider1.SetError(OdometerEndingTextBox, "Entry required!");

OdometerEndingTextBox.Focus();

e.Cancel = true;

}

else

ErrorProvider1.Clear();

}

private void DriversLicenseTextBox\_Validating(object sender, CancelEventArgs e)

{

Decimal.TryParse(DriversLicenseTextBox.Text, out NumberDecimal);

if (NumberDecimal == 0)

{

ErrorProvider1.SetError(DriversLicenseTextBox, "Entry required!");

DriversLicenseTextBox.Focus();

e.Cancel = true;

}

else

ErrorProvider1.Clear();

}

private void CreditCardNumberTextBox\_Validating(object sender, CancelEventArgs e)

{

Decimal.TryParse(CreditCardNumberTextBox.Text, out NumberDecimal);

if (NumberDecimal == 0)

{

ErrorProvider1.SetError(CreditCardNumberTextBox, "Entry required!");

CreditCardNumberTextBox.Focus();

e.Cancel = true;

}

else

ErrorProvider1.Clear();

}

private void DaysRentedTextBox\_Validating(object sender, CancelEventArgs e)

{

Decimal.TryParse(DaysRentedTextBox.Text, out NumberDecimal);

if (NumberDecimal == 0)

{

ErrorProvider1.SetError(DaysRentedTextBox, "Entry required!");

DaysRentedTextBox.Focus();

e.Cancel = true;

}

else

ErrorProvider1.Clear();

}

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

{

Close();

}

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

{

int OdometerBeginning, OdometerEnding, DaysRented, CarSizeInt, Discount = 0;

// Validate that entries have been made for the DL# & days rented, & that a car size is selected

if (DriversLicenseTextBox.Text != "")

{

if (DaysRentedTextBox.Text != "")

{

if (SizeCompactRadioButton.Checked || SizeMidsizeRadioButton.Checked || SizeLuxuryRadioButton.Checked)

{

// Begin seeing if values are parsable

Int32.TryParse(DaysRentedTextBox.Text, out DaysRented);

Int32.TryParse(OdometerBeginningTextBox.Text, out OdometerBeginning);

// If there is a positive value in the beginning mileage, parse the ending mileage

if (OdometerBeginning > 0)

{

Int32.TryParse(OdometerEndingTextBox.Text, out OdometerEnding);

// If days rented, beginning mileage, and ending mileage are all valid, assign CarSizeInt value

if (OdometerEnding > 0)

{

// Casts carsizeint values to the corresponding values in the enum

if (SizeCompactRadioButton.Checked)

CarSizeInt = (int)CarSize.Compact;

else if (SizeMidsizeRadioButton.Checked)

CarSizeInt = (int)CarSize.MidSize;

else

CarSizeInt = (int)CarSize.Luxury;

// Validates that the starting mileage should be lesser than the ending milage

if (OdometerBeginning > OdometerEnding)

{

ErrorProvider1.SetError(OdometerBeginningTextBox, "The beginning mileage must be lower than the ending mileage.");

OdometerBeginningTextBox.Focus();

OdometerBeginningTextBox.SelectAll();

}

else

{

// Determines which discount - if any - to use in CorporateRate class's Discount enum, then calculates amounts owed

if (DiscountCorporateRadioButton.Checked)

{

Discount = 1;

CorporateRate corporateTotal = new CorporateRate(OdometerBeginning, OdometerEnding, CarSizeInt, DaysRented, Discount);

AmountDueTextBox.Text = corporateTotal.getAmountDue().ToString("C");

}

else if (DiscountInsuranceAcctRadioButton.Checked)

{

Discount = 2;

CorporateRate insuranceTotal = new CorporateRate(OdometerBeginning, OdometerEnding, CarSizeInt, DaysRented, Discount);

AmountDueTextBox.Text = insuranceTotal.getAmountDue().ToString("C");

}

else

{

Discount = 0;

RentalRate totalCost = new RentalRate(OdometerBeginning, OdometerEnding, CarSizeInt, DaysRented);

AmountDueTextBox.Text = totalCost.getAmountDue().ToString("C");

}

}

}

// Begin the error handling and focusing back to the required controls

else

{

ErrorProvider1.SetError(OdometerEndingTextBox, "Please enter the ending mileage.");

OdometerEndingTextBox.Focus();

OdometerEndingTextBox.SelectAll();

}

}

else

{

ErrorProvider1.SetError(OdometerBeginningTextBox, "Please enter the beginning mileage.");

OdometerBeginningTextBox.Focus();

OdometerBeginningTextBox.SelectAll();

}

}

else

{

ErrorProvider1.SetError(SizeCompactRadioButton, "Please check one of the car size values.");

}

}

else

{

ErrorProvider1.SetError(DaysRentedTextBox, "Please enter the amount of days the car was rented.");

DaysRentedTextBox.Focus();

DaysRentedTextBox.SelectAll();

}

}

else

{

ErrorProvider1.SetError(DriversLicenseTextBox, "Please enter your drivers' license number.");

DriversLicenseTextBox.Focus();

DriversLicenseTextBox.SelectAll();

}

}

}

}

/\*

'Program: EX02CarRentals

'Programmer: Theresa Berry and John Blyzka

'Date: August 2008

'Class: RentalRate

'Description: This class calculates the car rental rate.

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Assignment01

{

public enum CarSize : int

{

Compact,

MidSize,

Luxury,

}

class RentalRate

{

protected int BeginInteger, EndInteger, SizeInteger, NumberDaysInteger, NumberMilesInteger;

protected Decimal DailyRateDecimal, MileageRateDecimal, AvgDailyMilesDecimal, AmountDueDecimal;

const Decimal COMPACT\_DAILY\_RATE\_Decimal = 26.95M;

const Decimal MIDSIZE\_DAILY\_RATE\_Decimal = 32.95M;

const Decimal LUXURY\_DAILY\_RATE\_Decimal = 50.95M;

const Decimal COMPACT\_MILEAGE\_RATE\_Decimal = 0.12M;

const Decimal MIDSIZE\_MILEAGE\_RATE\_Decimal = 0.15M;

const Decimal LUXURY\_MILEAGE\_RATE\_Decimal = 0.2M;

protected const int MAX\_MILES\_Integer = 100;

public RentalRate(int BeginOdometerInteger, int EndOdometerInteger, int CarSizeInteger, int DaysInteger)

{

//Assign the property values.

BeginInteger = BeginOdometerInteger;

EndInteger = EndOdometerInteger;

SizeInteger = CarSizeInteger;

NumberDaysInteger = DaysInteger;

FindRentalRates();

FindNumberMiles();

FindAmountDue();

}

// default constructor required

public RentalRate()

{

BeginInteger = 0;

}

private void FindNumberMiles()

{

NumberMilesInteger = EndInteger - BeginInteger; // Determine the number of miles.

// Determine the average number of miles per day.

Decimal.TryParse((NumberMilesInteger / NumberDaysInteger).ToString(), out AvgDailyMilesDecimal);

}

public Decimal getAmountDue()

{

return AmountDueDecimal;

}

private void FindAmountDue()

{

// There is no mileage charge if the average of the miles does not exceed an average of 100 miles per day rented.

if (AvgDailyMilesDecimal <= MAX\_MILES\_Integer)

AmountDueDecimal = (DailyRateDecimal \* NumberDaysInteger);

else

AmountDueDecimal = (DailyRateDecimal \* NumberDaysInteger) + (NumberMilesInteger \* MileageRateDecimal);

}

private void FindRentalRates()

{

// Calculate the rental rate.

switch ((CarSize)SizeInteger) // cast integer into type CarSize

{

case CarSize.Compact:

DailyRateDecimal = COMPACT\_DAILY\_RATE\_Decimal;

MileageRateDecimal = COMPACT\_MILEAGE\_RATE\_Decimal;

break;

case CarSize.MidSize:

DailyRateDecimal = MIDSIZE\_DAILY\_RATE\_Decimal;

MileageRateDecimal = MIDSIZE\_MILEAGE\_RATE\_Decimal;

break;

case CarSize.Luxury:

DailyRateDecimal = LUXURY\_DAILY\_RATE\_Decimal;

MileageRateDecimal = LUXURY\_MILEAGE\_RATE\_Decimal;

break;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment01

{

public enum Discount : int

{

Corporate = 1,

Insurance

}

// Corporate Rate Class

class CorporateRate : RentalRate

{

private int DiscountRateInteger;

const Decimal CORPORATE\_DISCOUNT\_Decimal = 0.05M;

const Decimal INSURANCE\_DISCOUNT\_Decimal = 0.10M; // Original value was 0.01M, corrected to 0.10M

public CorporateRate(int BeginnerOdometerInteger, int EndingOdometerInteger, int CarSizeInteger, int DaysInteger, int DiscountInteger)

: base (BeginnerOdometerInteger, EndingOdometerInteger, CarSizeInteger, DaysInteger)

{

DiscountRateInteger = DiscountInteger;

FindAmountDue();

}

// overrides base form method FindAmountDue()

private new Decimal FindAmountDue()

{

// Determine the amount due.

Decimal SubTotalDecimal, DiscountedDailyRateDecimal;

switch ((Discount)DiscountRateInteger) // uses enum Discount defined above

{

case Discount.Corporate:

// do math to determine AmountDueDecimal

// Corporate accounts waive the mileage rate and have a 5 percent discount.

SubTotalDecimal = ((DailyRateDecimal - (DailyRateDecimal \* CORPORATE\_DISCOUNT\_Decimal)) \* NumberDaysInteger);

AmountDueDecimal = SubTotalDecimal;

break;

case Discount.Insurance:

// Insurance accounts have a 10 percent discount on the daily rate.

// do math here

// There is no mileage charge if the average of the miles

// does not exceed an average of 100 miles per day rented.

DiscountedDailyRateDecimal = ((DailyRateDecimal - (DailyRateDecimal \* INSURANCE\_DISCOUNT\_Decimal)) \* NumberDaysInteger);

// if statement to decide the correct calculation

if (AvgDailyMilesDecimal <= MAX\_MILES\_Integer)

AmountDueDecimal = DiscountedDailyRateDecimal;

else

{

SubTotalDecimal = DiscountedDailyRateDecimal + (NumberMilesInteger \* MileageRateDecimal);

AmountDueDecimal = SubTotalDecimal;

}

break;

}

return AmountDueDecimal;

}

}

}