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 hw2.bowden

{

public partial class frmDraw : Form

{

//form variables

protected bool shouldDraw = false; //whether to use small, medium, or large pen size

protected int PenSize; //this variable holds the thickeness of the pen

protected Color PenColor; //this variable holds the color

protected Bitmap bmpDraw; //this bitmap variable holds the drawing;

//enumeration to control PenSize in one spot

protected enum PenSize\_enum

{

Small = 7,

Medium = 11,

Large = 16

}

//default constructor

public frmDraw()

{

InitializeComponent();

//set the initial Pen size and color

PenSize = (int)PenSize\_enum.Small;

PenColor = Color.Red;

//create a bitmap to hold our drawing

bmpDraw = new Bitmap(pnlDraw.Width, pnlDraw.Height);

}//end constructor

//should draw when the user presses down on mouse button

private void pnlDraw\_MouseDown(object sender, MouseEventArgs e)

{

//user is dragging the mouse button

shouldDraw = true;

}//end method frmDraw\_MouseDown

//should draw stop drawing when mouse button is released

private void pnlDraw\_MouseUp(object sender, MouseEventArgs e)

{

//user has released the mouse button

shouldDraw = false;

}//end method frmDraw\_MouseUp

//user draws circles whenever the mouse moves when the button is held down

private void pnlDraw\_MouseMove(object sender, MouseEventArgs e)

{

//check to see if mouse button is being pressed

if (shouldDraw)

{

//draw a circle where the mouse button is being pressed

using (Graphics graphics = Graphics.FromImage(bmpDraw))

{

//draw an ellipse and then fill it in

graphics.DrawEllipse(new Pen(PenColor), e.X, e.Y, PenSize, PenSize);

graphics.FillEllipse(new SolidBrush(PenColor), e.X, e.Y, PenSize, PenSize);

//force the paint event to fire

pnlDraw.Invalidate();

}//end using; calss graphics.Dispose()

}//end if

}//end method frm Draw\_MouseMove

//user has pushed the rdoRed button

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

{

//set the color to red

PenColor = Color.Red;

}//end PenColor

//user has pushed the rdoBlue button

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

{

//set color to blue

PenColor = Color.Blue;

}//end PenColor

//user has pushed the rdoGreen button

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

{

//set color to green

PenColor = Color.Green;

}//end PenColor

//the user has pushed the rdoBlack button

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

{

//set color to black

PenColor = Color.Black;

}//end PenColor

//the user has pushed the rdoSmall button

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

{

//set the pen size to small

PenSize = (int)PenSize\_enum.Small;

}//end PenSize

//the user has pushed the rdoMedium button

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

{

//set the pen size to medium

PenSize = (int)PenSize\_enum.Medium;

}//end PenSize

//the user has pushed the rdoLarge button

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

{

//set the pen size to large

PenSize = (int)PenSize\_enum.Large;

}//end PenSize

//use the Paint event to write the bitmap to the panel

private void pnlDraw\_Paint(object sender, PaintEventArgs e)

{

//draw the graphics to a bitmap

e.Graphics.DrawImage(bmpDraw, new Point(0, 0));

}//end pnlDraw\_Paint

//frmDraw\_Load event

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

{

//I am leaving this empty event here to preserve a landing spot in the code for

//when I double click the form

}//end frmDraw\_Load

}

}