МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
“Харківський авіаційний інститут”

Кафедра комп’ютерних систем та мереж

Лабораторна робота № 6

“Разработка и исследование программы построения часов”

По дисциплине “Технологии программирования”

XAI.503.525B.123. 1705067.180

Виконав студент гр. 525В Пеналоза Г.
(№ групи) (П.І.Б.)

15-04-19
(підпис, дата)

Перевірив ст. викладач каф. 503
(науковий ступінь, вчене звання, посада)

(підпис, дата) (П.І.Б.)

2019

Задание:

Постановка задачи: разработать программу для построения на форме часов. Часы должны:
- содержать три стрелки – часовую, минутную и секундную,
- показывать время (часы, минуты и секунды) в цифровом формате.

Проект должен содержать три формы:
о 1-я форма – часы с цифрами 3, 6, 9, 12 на циферблате и штрихами для остальных часов,
о 2-я форма – для задания параметров часов (форма, которая вызывается из контекстного меню формы с часами),
о 3-я форма – для сведений об авторе проекта (форма, которая вызывается из контекстного меню формы с часами).
Разработать модульные тесты (unit-тесты) для методов класса.

Вариант 20

20	json

Текст программы

Form.cs

```
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 Lab2._2._6
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      ///Instance of the configuration form
      Configurations conf = new Configurations();
```

```
///Point to help move
private Point pos = new Point();
/// <summary>
/// Variables for the center
/// </summary>
private int xc, yc;
/// <summary>
/// Size of the min, sec, hor
/// </summary>
private int min, sec, hor;
/// <summary>
/// Radio size
/// </summary>
private int radio;
/// <summary>
/// Clock
/// </summary>
private Graphics g_clock;

private void Form1_Paint(object sender, PaintEventArgs e)
{
}

private void Form1_Click(object sender, EventArgs e)
{

}

private void menuContextual_Opening(object sender, CancelEventArgs e)
{

}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
   Application.Exit();
}

/// <summary>
/// Move the Clock
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
```

```
            switch (MouseButtons)
            {
                case MouseButtons.Left:
                    break;
                case MouseButtons.Right:
                    menuContextual.Show(this,e.Location);
                    break;
            }
            this.pos.X = e.X;
            this.pos.Y = e.Y;
        }

        /// <summary>
        /// Show information about autor
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About about = new About();
            about.ShowDialog();
        }

        /// <summary>
        /// Open configurations
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void configurationsToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            conf.Refresh();
            conf.Show();
            conf.Focus();
        }

        /// <summary>
        /// Move Clock
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button != MouseButtons.Left)
                return;
```

```
            this.Left = Control.MousePosition.X - this.pos.X;
            this.Top = Control.MousePosition.Y - this.pos.Y;
        }

        /// <summary>
        /// Timer to draw the clock with the parameters
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void timer1_Tick(object sender, EventArgs e)
        {
            g_clock = this.CreateGraphics();
            g_clock.Clear(BackColor);
            int[] finalPoint = new int[2];
            this.Opacity = Clock.opacity;
            if (Clock.digitalClock)//Show digital clock
            {
                string digitalClock = DateTime.Now.ToString("h:m:s");
                g_clock.DrawString(digitalClock, new Font("Arial", 15), new
SolidBrush(Color.Violet), new PointF(xc-30, yc+30));
            }
            if (Clock.small)//Size clock
            {
                radio = Width / 4 - 15;
                hor = radio - 20*4;
                min = radio - 10*3;
                sec = radio - 5*2;
            }
            else
            {
                radio = Width / 2 - 15;
                hor = radio - 20*4;
                min = radio - 10*3;
                sec = radio - 5*2;
            }

            //Get and set the time
            Clock.time[0] = DateTime.Now.Hour;
            Clock.time[1] = DateTime.Now.Minute;
            Clock.time[2] = DateTime.Now.Second;

            //Draw the circle
            Pen circle = new Pen(Clock.colors[0]);
            circle.Width = 8.0F;
            circle.LineJoin = System.Drawing.Drawing2D.LineJoin.Bevel;
```

```
            Methods.DrawCircle(g_clock, circle, xc, yc, radio);
            circle.Dispose();

            //Draw numbers in the clock
            g_clock.DrawString("12", new Font("Arial", 12), new
SolidBrush(Clock.colors[0]), new PointF(xc-10, yc-radio+5));
            g_clock.DrawString("3", new Font("Arial", 12), new
SolidBrush(Clock.colors[0]), new PointF(xc+radio-20, yc-5));
            g_clock.DrawString("6", new Font("Arial", 12), new
SolidBrush(Clock.colors[0]), new PointF(xc-6, yc+radio-20));
            g_clock.DrawString("9", new Font("Arial", 12), new
SolidBrush(Clock.colors[0]), new PointF(xc-radio+8, yc-5));

            //Draw sec
            finalPoint = Methods.msCoord(Clock.time[2], sec,xc,yc);
            g_clock.DrawLine(new Pen(Clock.colors[3], 1f), new Point(xc, yc), new
Point(finalPoint[0], finalPoint[1]));

            //Draw min
            finalPoint = Methods.msCoord(Clock.time[1], min,xc,yc);
            g_clock.DrawLine(new Pen(Clock.colors[2], 2f), new Point(xc, yc), new
Point(finalPoint[0], finalPoint[1]));

            //Draw hor
            finalPoint = Methods.hCoord(Clock.time[0] % 12, Clock.time[1],
hor,xc,yc);
            g_clock.DrawLine(new Pen(Clock.colors[1], 3f), new Point(xc, yc), new
Point(finalPoint[0], finalPoint[1]));

            g_clock.Dispose();
        }

        /// <summary>
        /// Load information
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Form1_Load(object sender, EventArgs e)
        {
            timer1.Start();
            this.xc = this.Width / 2;
            this.yc = this.Height / 2;
            Clock.Start();
        }
    }
```

```
}
```

About.cs

```
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 Lab2._2._6
{
   public partial class About : Form
   {
      public About()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         this.Close();
      }
   }
}
```

Configurations.cs

```
using System;
using System.Linq;
using System.Windows.Forms;
using System.IO;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System.Reflection;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Runtime.Serialization.Json;
using Newtonsoft.Json.Converters;
using Newtonsoft.Json.Serialization;


namespace Lab2._2._6
{
    public partial class Configurations : Form
    {
        public Configurations()
        {
            InitializeComponent();
        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Visible = false;
        }
        /// <summary>
        /// Show digital clock
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked)
            {
                Clock.digitalClock = true;
            }
            else
            {
                Clock.digitalClock = false;
            }
        }
        /// <summary>
        /// Show small or large
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
```

```
{
   if (comboBox3.SelectedIndex==0)
   {
      Clock.small = true;
   }
   else
   {
      Clock.small = false;
   }
}

/// <summary>
/// Set opacity
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
   Clock.opacity = Int32.Parse(comboBox2.Text.Substring(0,2));
}

/// <summary>
/// Set default values
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Configurations_Load(object sender, EventArgs e)
{
   comboBox3.SelectedIndex = 0;
   checkBox1.Checked = true;
   comboBox2.SelectedIndex = 0;
}
/// <summary>
/// Set color to clock elements
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
   ColorDialog cd = new ColorDialog();
   switch (comboBox1.SelectedIndex)
   {
      case 0:
         if (cd.ShowDialog() == DialogResult.OK)
            Clock.colors[0] = cd.Color;
```

```
                break;
            case 1:
                if (cd.ShowDialog() == DialogResult.OK)
                    Clock.colors[1] = cd.Color;
                break;
            case 2:
                if (cd.ShowDialog() == DialogResult.OK)
                    Clock.colors[2] = cd.Color;
                break;
            case 3:
                if (cd.ShowDialog() == DialogResult.OK)
                    Clock.colors[3] = cd.Color;
                break;
        }
    }
    /// <summary>
    /// Write conf.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button3_Click(object sender, EventArgs e)
    {
        Clock.ToARGB();
        SaveFileDialog sfd = new SaveFileDialog { Filter = @"JSON files (*.json)|
*.json" };
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            var w_conf = new Clock();
            var json = JsonConvert.SerializeObject(w_conf);
            File.WriteAllText(sfd.FileName, json);
            MessageBox.Show("The configuration was saved!");
        }
    }
    /// <summary>
    /// Read conf.
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button2_Click(object sender, EventArgs e)
    {
        Clock.ToARGB();
        OpenFileDialog sfd = new OpenFileDialog { Filter = @"JSON files
(*.json)|*.json" };
        if (sfd.ShowDialog() == DialogResult.OK)
        {
```

```
            var source =
JsonConvert.DeserializeObject<JToken>(File.ReadAllText(sfd.FileName));
            MapJTokenToStaticClass(source);
         }
         this.checkBox1.Checked = Clock.digitalClock;
         comboBox2.SelectedItem = Clock.opacity.ToString() + "%";
         if (Clock.small)
            comboBox3.SelectedIndex = 0;
         else
            comboBox3.SelectedIndex = 1;

      }
      public void MapJTokenToStaticClass(JToken source)
      {
         var destinationProperties = typeof(Clock)
            .GetProperties(BindingFlags.Public | BindingFlags.Static);
         int c = 0;
         foreach (JProperty prop in source)
         {
            var destinationProp = destinationProperties
               .SingleOrDefault(p => p.Name.Equals(prop.Name,
StringComparison.OrdinalIgnoreCase));
            if (c<3) {
               var value = ((JValue)prop.Value).Value;
               destinationProp.SetValue(null, Convert.ChangeType(value,
destinationProp.PropertyType));
               c++;
            }
            else
            {
               var value = ((JArray)prop.Value).ToArray();
               Clock.argb_Colors[0] = (Int32)value[0];
               Clock.argb_Colors[1] = (Int32)value[1];
               Clock.argb_Colors[2] = (Int32)value[2];
               Clock.argb_Colors[3] = (Int32)value[3];
               Clock.FromARGB();
            }
         }
      }

      private void label2_Click(object sender, EventArgs e)
      {
      }
   }
}
```

Methods.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Runtime.Serialization;
using System.Windows.Forms;
using System.Runtime.Serialization.Json;
using Newtonsoft.Json;

namespace Lab2._2._6
{
   public class Clock
   {
      [JsonProperty]
      public static bool digitalClock { get; set; }
      [JsonProperty]
      public static bool small { get; set; }
      [JsonProperty]
      public static int opacity { get; set; }
      [JsonProperty]
      public static int[] argb_Colors { get; set; }
      public static int[] time { get; set; }
      public static Color[] colors { get; set; }

      public Clock()
      {
      }
      //Default values
      public static void Start()
      {
         time = new int[3];
         colors = new Color[4];
         colors[0] = Color.Black;
         colors[1] = Color.Red;
         colors[2] = Color.Green;
         colors[3] = Color.Blue;
         digitalClock = true;
         small = true;
         opacity = 20;
```

```
        }
        //Convert to ARGB colors
        public static void ToARGB()
        {
            argb_Colors = new int[4];
            for (int i = 0; i < colors.Length; i++)
                argb_Colors[i] = colors[i].ToArgb();
        }
        //Convert from ARGB colors
        public static void FromARGB()
        {
            colors = new Color[4];
            for (int i = 0; i < colors.Length; i++)
                colors[i] = Color.FromArgb(argb_Colors[i]);
        }
    }
    public class Methods
    {
        //Draw circle
            public static void DrawCircle(Graphics g, Pen pen, float centerX, float centerY, float radius)
        {
                g.DrawEllipse(pen, centerX - radius, centerY - radius, radius + radius, radius + radius);
        }
        //Get coordenate for min and sec
        public static int[] msCoord(int val, int hlen,int cx,int cy)
        {
            int[] coord = new int[2];
            val *= 6;

            if (val >= 0 && val <= 180)
            {
                coord[0] = cx + (int)(hlen * Math.Sin(Math.PI * val / 180));
                coord[1] = cy - (int)(hlen * Math.Cos(Math.PI * val / 180));
            }
            else
            {
                coord[0] = cx - (int)(hlen * -Math.Sin(Math.PI * val / 180));
                coord[1] = cy - (int)(hlen * Math.Cos(Math.PI * val / 180));
            }
            return coord;
        }
        //Get coordenate for hr
        public static int[] hCoord(int hval, int mval, int hlen,int cx,int cy)
```

```
        {
            int[] coord = new int[2];

            int val = (int)((hval * 30) + (mval * 0.5));

            if (val >= 0 && val <= 180)
            {
                coord[0] = cx + (int)(hlen * Math.Sin(Math.PI * val / 180));
                coord[1] = cy - (int)(hlen * Math.Cos(Math.PI * val / 180));
            }
            else
            {
                coord[0] = cx - (int)(hlen * -Math.Sin(Math.PI * val / 180));
                coord[1] = cy - (int)(hlen * Math.Cos(Math.PI * val / 180));
            }
            return coord;
        }

    }
}
```

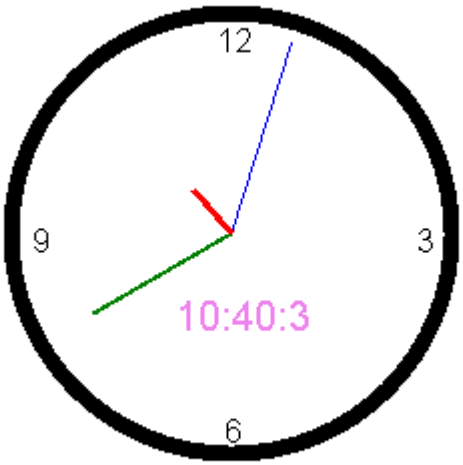
Диаграмму классов

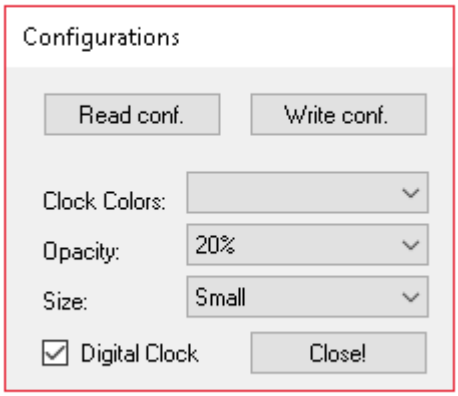
UnitTests

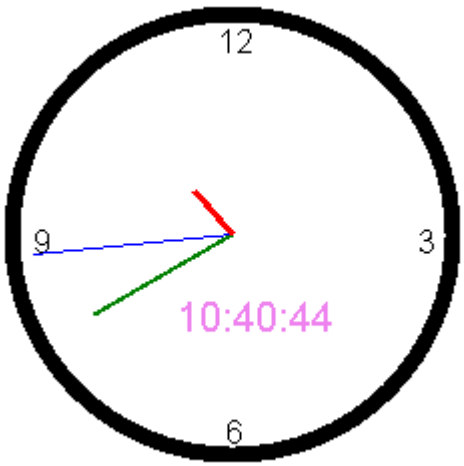
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Drawing;
using Lab2._2._6;

namespace UnitTest
{
   [TestClass()]
   public class SettingsTests
   {
      [TestMethod()]
      //-16777216,-65536,-16744448,-16776961
      public void ChecktoARGBcolors()
      {
         Clock.Start();
         Clock.ToARGB();
         Assert.AreEqual(-16777216, Clock.argb_Colors[0]);
         Assert.AreEqual(-65536, Clock.argb_Colors[1]);
         Assert.AreEqual(-16744448, Clock.argb_Colors[2]);
         Assert.AreEqual(-16776961, Clock.argb_Colors[3]);
      }

      [TestMethod()]
      public void CheckfromARGBcolors()
      {
         Clock.Start();
         Clock.ToARGB();
         Clock.FromARGB();
         Assert.AreEqual(Color.Black.ToArgb(), Clock.colors[0].ToArgb());
         Assert.AreEqual(Color.Red.ToArgb(), Clock.colors[1].ToArgb());
         Assert.AreEqual(Color.Green.ToArgb(), Clock.colors[2].ToArgb());
         Assert.AreEqual(Color.Blue.ToArgb(), Clock.colors[3].ToArgb());
      }

      [TestMethod()]
      public void InitTest()
      {
         Clock.Start();
         Assert.AreEqual(3, Clock.time.Length);
         Assert.AreEqual(4, Clock.colors.Length);
         Assert.AreEqual(Color.Black, Clock.colors[0]);
         Assert.AreEqual(Color.Red, Clock.colors[1]);
         Assert.AreEqual(Color.Green, Clock.colors[2]);
```

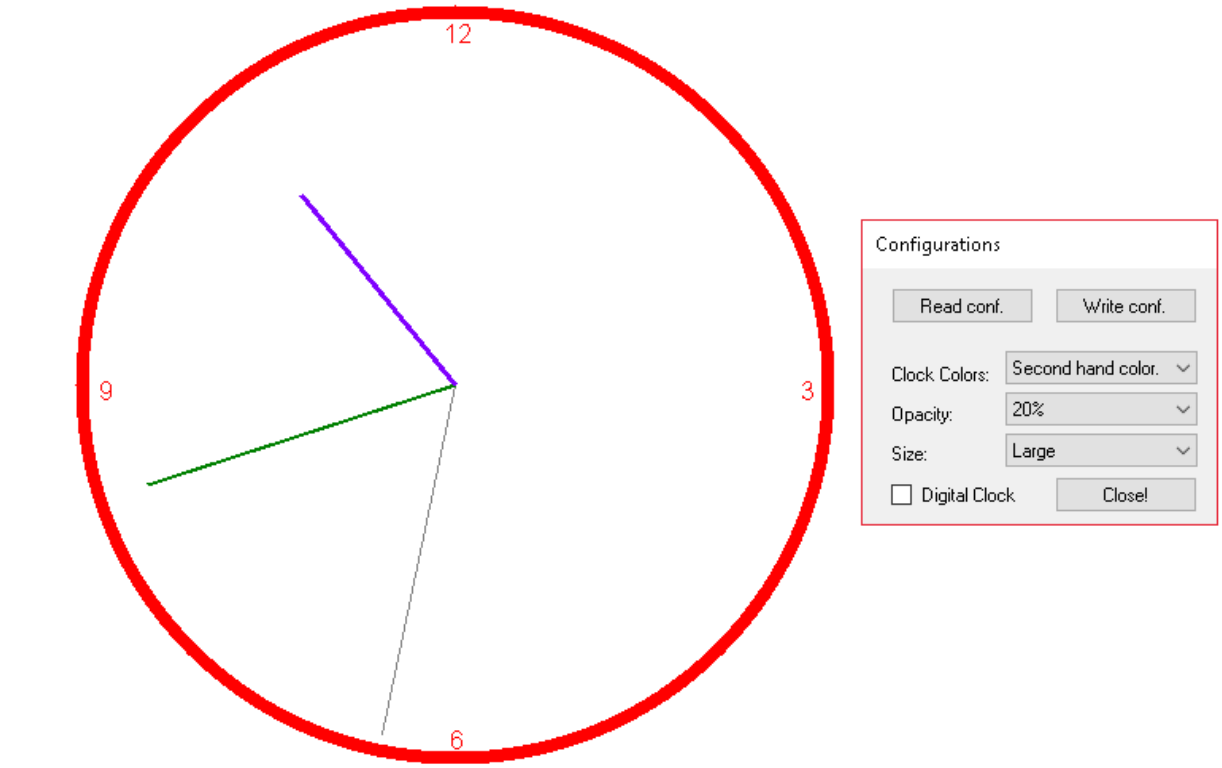
```
            Assert.AreEqual(Color.Blue, Clock.colors[3]);
        }
    }
}
```

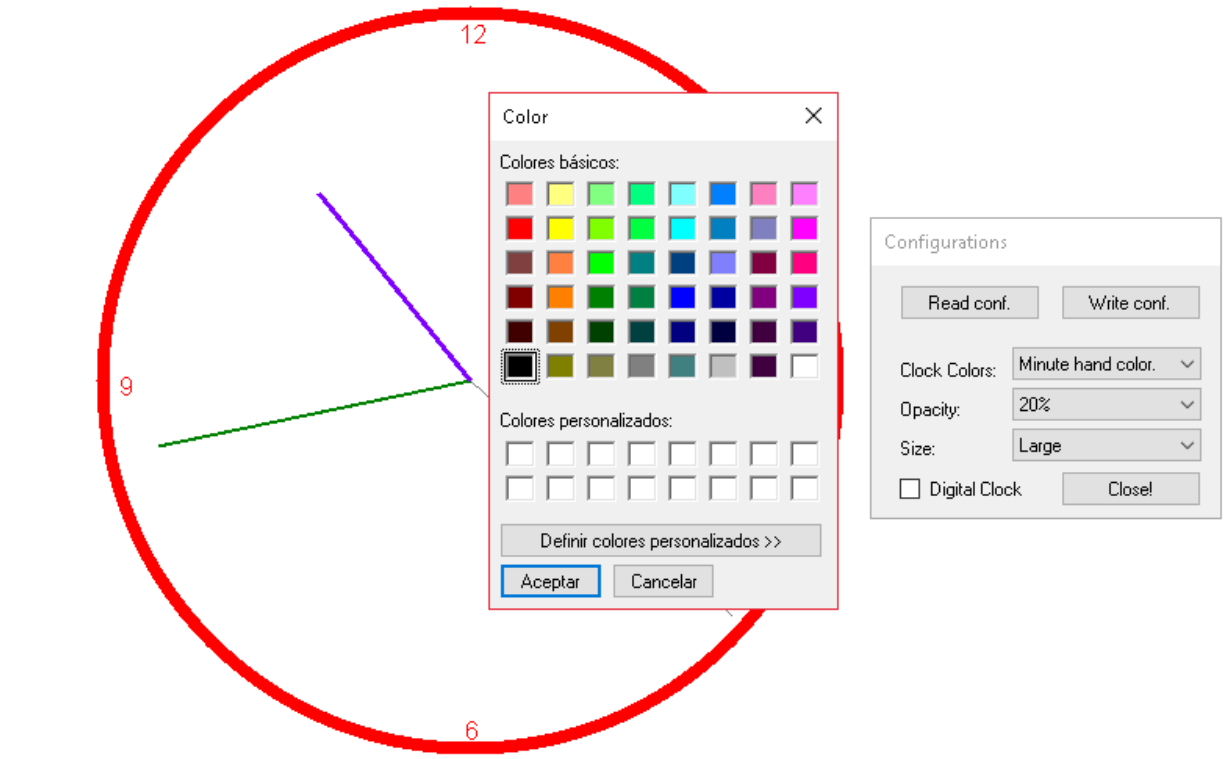
Скриншоты

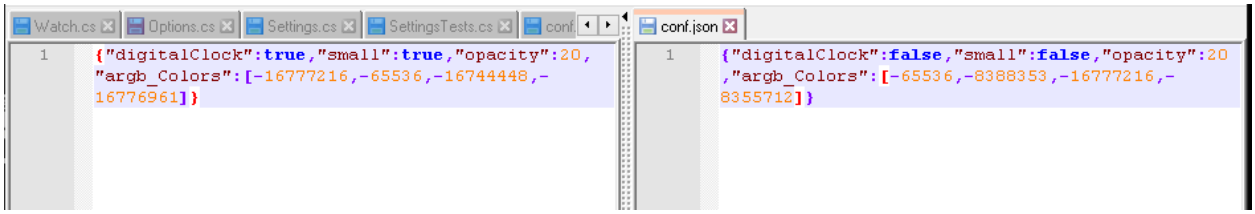







12
3
6
9
Configurations
Read conf.
Write conf.
Clock Colors:
Second hand color.
Opacity:
20%
Size:
Large
Digital Clock
Close!


12
6
9
Color
Colores básicos:
Colores personalizados:
Definir colores personalizados >>
Aceptar
Cancelar
Configurations
Read conf.
Write conf.
Clock Colors:
Minute hand color.
Opacity:
20%
Size:
Large
Digital Clock
Close!



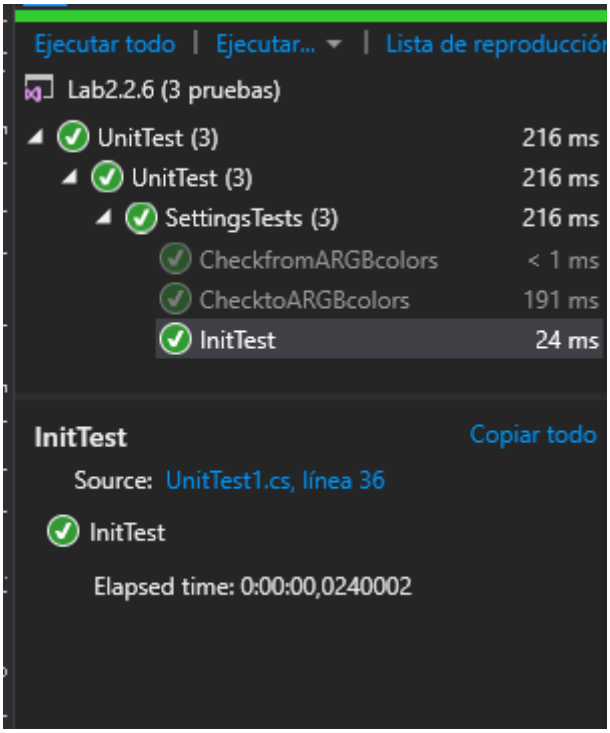


Таблица 1 - Поля и методы класса A и их назначение

N°	Поле	Назначение
1	g_clock	Clock
2	min	Size min
3	hor	Size hr
4	pos	Temp move
5	radio	Size radio
6	sec	Size sec
7	xc	Center x
8	yc	Center y
9	button1	Close conf
10	button2	Read conf
11	button3	Write conf
12	checkBox1	Show digital
13	comboBox1	Colors elements

14	comboBox2	Opacity
15	comboBox3	Size Clock
16	label1	Labels conf
17	label2	Labels conf
18	label3	Labels conf
19	argb_Colors	ARGB colors
20	colors	Colors of the elements
21	digitalClock	Show digital
22	opacity	Opacity
23	small	Size clock
24	time	Time
25	button1	Exit about
26	label1	Text of credits
	Метод	**Назначение**
27	Clock	Empty contructor
28	FromARGB	Convert from ARGB colors
29	Start	Start default values
30	ToARGB	Convert to ARGB colors
31	DrawCircle	Draw circle
32	hCoord	Get coordenate hr
33	msCoord	Get coordenate min and seg
34	MapJTokenToStaticClass	Read Json file to static class elements

Таблица 2 - Обработчики событий проекта и их назначение

N°	**Обработчик события**	**Назначение**
1	button2_Click	Read json conf
2	button3_Click	Write json conf
3	button1_Click	Close conf
4	checkBox1_CheckedChanged	Show digital clock
5	comboBox1_SelectedIndexChanged	Set color to clock elements

6	comboBox2_SelectedIndexChanged	Set opacity
7	comboBox3_SelectedIndexChanged	Show small or large
8	Configurations_Load	Set default values

Выводы

В лабораторной практике изучалось "Разработка и исследование программы построения часов”.
Разработанные часы сохраняют конфигурацию в файл json. Были проблемы с чтением статических свойств классов конфигурации.
Как улучшение: Реализуйте больше функций для настройки часов.

Использованные источники

1. https://docs.microsoft.com/es-es/dotnet/