PREDMET: Razvoj Programskih Rješenja	**NASTAVNI ANSAMBL**
AK. GOD. : 2017/2018 **RESURS:** Laboratorijska vježba 4 **DATUM OBJAVE:** 06.11.2017.	**PROFESOR:** Dženana Đonko (ddonko@etf.unsa.ba) **ANSAMBL:** Hasić Haris, Hodžić Kerim Đenašević Emir, Hasanbašić Denis, Mahmutović Mustafa, Ramić Benjamin

Cilj vježbi:

- Upoznavanje i rad sa osnovnim Windows Forms kontrolama u konzolnoj aplikaciji
- Upoznavanje i rad sa osnovnim događajima (*events*) u konzolnoj aplikaciji

Napomena: Prije izrade vježbe je obavezno pročitati predavanja vezana za tematiku vježbe i upoznati se sa osnovnim konceptima jezika C# i objektno orjentisanog programiranja.

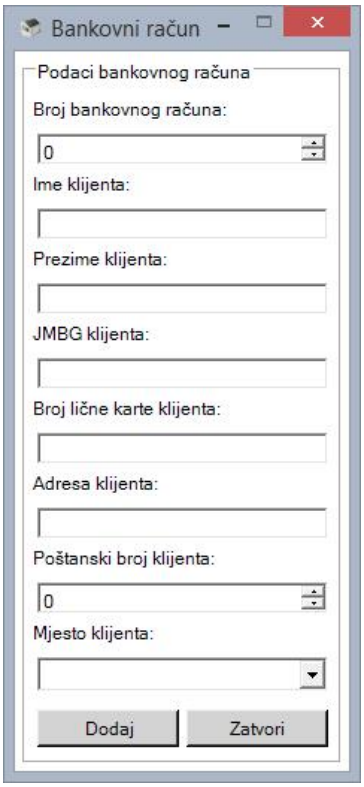
LABORATORIJSKA VJEŽBA 4

Osnovne Windows Forms kontrole u konzolnoj aplikaciji

Windows Forms

Windows Forms (skraćeno WinForms) je skup klâsa za razvoj Windows aplikacija. Windows aplikacije su aplikacije namijenjene za izvršavanje na Windows operativnim sistemima. Namjena *Windows Forms* skupa klâsa je osiguravanje grafičkog korisničkog interfejsa Windows aplikacijama. *Windows Forms* skup klâsa je dio Microsoft .NET biblioteke klâsa.

Primjer 1 – Kreirati konzolnu aplikaciju koja prilikom pokretanja prikazuje formu kao na sljedećoj slici:



Napomena: Konzolna aplikacija nema implicitno dodane reference na imenske prostore koji su potrebni za rad sa osnovnim Windows Forms kontrolama, a to su System.Drawing.dll i System.Windows.Forms.dll. Najprije je potrebno eksplicitno dodati reference na ove imenske prostore (desni klik na References -> Add Reference... -> Assemblies -> Framework -> System.Drawing/System.Windows.Forms -> OK.

Form klasa je prozorski kontejner koji u aplikaciji predstavlja korisnički interfejs i na sebi može sadržavati druge kontrole. Detaljna specifikacija **Form** klase se može naći na sljedećem linku: https://msdn.microsoft.com/en-us/library/system.windows.forms.form(v=vs.110).aspx

Formu kreiramo na sljedeći način:

```
Form form = new Form();
form.StartPosition = FormStartPosition.CenterScreen;
form.Size = new Size(240, 520);
form.MaximumSize = form.Size;
form.MinimumSize = form.Size;
form.MaximizeBox = false;
form.Icon = Properties.Resources.Banka;
form.Text = "Bankovni račun";
form.BackColor = Color.White;
form.ForeColor = Color.Black;
form.Font = new Font("Microsoft Sans Serif", 9, FontStyle.Regular);
form.Padding = new Padding(5, 5, 5, 5);
```

GroupBox je kontejnerska kontrola koja prikazuje okvir oko kontrola koje se na njoj nalaze i opcionalno naslov. Detaljna specifikacija **GroupBox** klase se može naći na sljedećem linku: https://msdn.microsoft.com/en-us/library/system.windows.forms.groupbox(v=vs.110).aspx

GroupBox dodajemo na sljedeći način:

```
GroupBox groupBox = new GroupBox();
groupBox.Dock = DockStyle.Fill;
groupBox.Text = "Podaci bankovnog računa";
form.Controls.Add(groupBox);
```

Label je jednostavna i često korištena kontrola koja služi za ispis teksta. Detaljna specifikacija **Label** klase se može naći na sljedećem linku: https://msdn.microsoft.com/en-us/library/system.windows.forms.label(v=vs.110).aspx

Labelu dodajemo na sljedeći način:

```
Label label1 = new Label();
label1.Size = new Size(195, 25);
label1.Text = "Broj bankovnog računa:";
label1.Location = new Point(5, 25);
groupBox.Controls.Add(label1);
```

NumericUpDown je kontrola koja omogućava odabir broja iz liste ponuđenih brojeva. NumericUpDown dodajemo na sljedeći način:

```
NumericUpDown numericUpDown1 = new NumericUpDown();
numericUpDown1.Size = new Size(195, 25);
numericUpDown1.Location = new Point(10, 50);
groupBox.Controls.Add(numericUpDown1);
```

TextBox je kontrola koja omogućava unos i ispis teksta. TextBox dodajemo na sljedeći način:

```
            TextBox textBox1 = new TextBox();
            textBox1.Size = new Size(195, 25);
            textBox1.Location = new Point(10, 100);
            groupBox.Controls.Add(textBox1);
```

ComboBox je kontrola koja omogućava odabir jedne od više ponuđenih opcija. ComboBox dodajemo na sljedeći način:

```
            ComboBox comboBox1 = new ComboBox();
            comboBox1.Size = new Size(195, 25);
            comboBox1.Location = new Point(10, 400);
            comboBox1.Items.AddRange(new string[] { "Sarajevo", "Banja Luka", "Tuzla",
"Mostar", "Zenica" });
```

Button je kontrola koja se koristi za interakciju sa korisnikom aplikacije. Najčešći se koristi korištenja za odrađivanje neke programske logike aplikacije u trenutku kada korisnik aplikacije klikne mišem na button. Button dodajemo na sljedeći način:

```
            Button button1 = new Button();
            button1.Size = new Size(95, 25);
            button1.Location = new Point(10, 435);
            button1.Text = "Dodaj";
            button1.BackColor = Color.LightGray;
            groupBox.Controls.Add(button1);
```

Za kraj, formu pokrećemo sa sljedećom komandom:

```
            Application.Run(form);
```

Primjer 2 – Doraditi konzolnu aplikaciju iz prethodnog primjera tako da je moguće:

1. Dodavati bankovne račune klikom na dugme Dodaj;
2. Obavještavati korisnika aplikacije da je bankovni račun uspješno dodan nakon klika na dugme Dodaj;
3. Zatvoriti formu klikom na dugme Zatvori;
4. Ispisivati sve dodane bankovne račune na konzolu nakon klika na dugme Zatvori;

Napomena: Podrazumijevati da će korisnik aplikacije ispunjavati sva ulazna polja i da će unositi isključivo validne vrijednosti u njih.

1. Dodavanje bankovnih računa klikom na dugme Dodaj;

Potrebno je kreirati klasu za bankovni račun koja sadrži odgovarajuće atribute i metode za pohranjivanje i ispisivanje bankovnih računa.

```
    public class BankovniRacun
    {
        private int brojBankovnogRacuna;
        private string imeKlijenta;
        private string prezimeKlijenta;
        private string jMBGKlijenta;
        private string brojLicneKarteKlijenta;
        private string adresaKlijenta;
        private int postanskiBrojKlijenta;
        private string mjestoKlijenta;

        public BankovniRacun(int brojBankovnogRacuna, string imeKlijenta, string
prezimeKlijenta, string jMBGKlijenta, string brojLicneKarteKlijenta, string adresaKlijenta,
int postanskiBrojKlijenta, string mjestoKlijenta)
```

```
        {
            this.brojBankovnogRacuna = brojBankovnogRacuna;
            this.imeKlijenta = imeKlijenta;
            this.prezimeKlijenta = prezimeKlijenta;
            this.jMBGKlijenta = jMBGKlijenta;
            this.brojLicneKarteKlijenta = brojLicneKarteKlijenta;
            this.adresaKlijenta = adresaKlijenta;
            this.postanskiBrojKlijenta = postanskiBrojKlijenta;
            this.mjestoKlijenta = mjestoKlijenta;
        }

        public void ispisi()
        {
            Console.WriteLine("{0} {1} {2} {3} {4} {5} {6} {7}", brojBankovnogRacuna,
imeKlijenta, prezimeKlijenta, jMBGKlijenta, brojLicneKarteKlijenta, adresaKlijenta,
postanskiBrojKlijenta, mjestoKlijenta);
        }
    }
```

Potrebno je kreirati listu objekata tipa klase za bankovni račun za pohranjivanje dodanih bankovnih računa.

```
private static List<BankovniRačun> bankovniRačuni = new List<BankovniRačun>();
```

Nakon toga je potrebno kreirati odgovarajući događaj za dugme Dodaj. Događaj (Event) je poruka koju šalje neki objekat kako bi signalizirao da se desila neka akcija. Na primjer, kada pritisnemo dugme Start u Windows-u desi se događaj koji uzrokuje da nam se pokaže panel sa dokumentima i aplikacijama. Eventi su implementirani koristeći EventHandler delegat čiji je prototip `void delegat (object sender, EventArgs e).` Prvi parametar delegata je referenca na objekat koji pokreće događaj (event) i on se zove pošiljalac (sender), drugi parametar su podaci koje događaj sadrži.

Klikom na button Dodaj želimo da unesene podatke proslijedimo konstruktoru klase BankovniRačun i da tu instancu klase dodamo u listu bankovnih računa.

```
        static void klikNaDugmeDodaj(object sender, EventArgs e)
        {
            bankovniRačuni.Add(new BankovniRačun(Convert.ToInt32(numericUpDown1.Value),
textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text,
Convert.ToInt32(numericUpDown2.Value), comboBox1.Text));
        }
```

Potrebno je povezati dugme Dodaj sa kreiranim događajem.

```
        button1.Click += new EventHandler(klikNaDugmeDodaj);
```

2. Obavještavati korisnika aplikacije da je bankovni račun uspješno dodan nakon klika na dugme Dodaj.

Dopunićemo klikNaDugmeDodaj porukom o uspješnosti akcije. To je urađeno koristeći MessageBox klasu (konkretno njenu Show metodu):

```
        static void klikNaDugmeDodaj(object sender, EventArgs e)
        {
            bankovniRačuni.Add(new BankovniRačun(Convert.ToInt32(numericUpDown1.Value),
textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text,
Convert.ToInt32(numericUpDown2.Value), comboBox1.Text));
            MessageBox.Show("Bankovni račun je uspješno dodan.", "Informacija",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
```

Na ovaj način smo postigli da kada se klikne na button, desiće se njegov Click event i onda će se izvršiti naša metoda klikNaDugmeDodaj.

3. Zatvoriti formu klikom na dugme Zatvori.

Na sličan način kreiramo događaj koji želimo da se izvrši klikom za dugme Zatvori.

```
        static void klikNaDugmeZatvori(object sender, EventArgs e)
        {
            form.Close();
            Console.WriteLine("Broj bankovnog računa Ime klijenta Prezime klijenta JMBG
klijenta Broj lične karte klijenta Adresa klijenta Poštanski broj klijenta Mjesto
klijenta");
            foreach (BankovniRačun bankovniRačun in bankovniRačuni)
            {
                bankovniRačun.ispiši();
            }
        }
```

Povezujemo kreiranu metodu Click eventom druge Button kontrole.

```
        button2.Click += new EventHandler(klikNaDugmeZatvori);
```

Kôd za primjer 1: http://pastebin.com/aF1m7UnZ
Kôd za primjer 2: http://pastebin.com/m06Y8m1J

Zadaci za samostalan rad i vježbu

Zadatak 1

Kreirati konzolnu aplikaciju koja prilikom pokretanja prikazuje formu kao na sljedećoj slici:

Potrebno je implementirati sve funkcionalnosti digitrona u skladu sa slikom (sabiranje, oduzimanje, množenje i dijeljenje). Digitron treba raditi samo sa cijelim brojevima. Ako rezultat operacije nije cijeli broj, onda treba vratiti 0 kao rezultat i obavijestiti korisnika digitrona da rezultat operacije nije cijeli broj. Za kreiranje dugmadi obavezno je koristiti petlju. Za preuzimanje vrijednosti sa dugmadi obavezno je koristiti parametar događaja *sender*. Početna pozicija forme digitrona je centar ekrana. Veličinu forme digitrona nije moguće mijenjati. Sve osobine klasâ Form, TextBox i Button dostupne su respektivno na sljedećim linkovima: osobine kontrole Form, osobine kontrole TextBox i osobine kontrole Button.

Zadatak 2

Modificirati zadatak 1 tako da se kreira naučni kalkulator sa tipkama za trigonometrijske funkcije.