**28. Радна недеља (30.03. 2020. - 03.04. 2020.)**

Predmet : Programiranje
Odeljenje: IV1
Tema : Liste u programskom jeziku c#, Kontrola ListBox
Pitanja i zadaci biće naknadno poslati. **Provežbati primere.**

**Liste ( sažeto )**

- **Lista je skup objekata, istog tipa - generička lista, bez unapred poznatog broja, ondnosno sa mogućnošću da se naknadno u već formiranu listu dodaju ili izbacuju elementi.**
- **Ako se radi o generičkoj listi:**
- `List<tip> t = new List<tip>();`
- **Dodavanje elementa listi:**
- `t.Add(22); t.Add(123);`
- **Brisanje cele listi:**
- `t.Clear();`
- **Izbacivanje elementa iz liste**
- `t.Remove(obj);` **ili**
- `t.RemoveAt(2);`

**ListBox**

U mnogim aplikacijama se vrši izbor jedne od više mogućih stavki iz neke liste stavki. Kontrola koja pruža ovu mogućnost se naziva **List Box** kontrola. Ovo je složenija kontrola, koja se sastoji od više stavki. Na primer, može predstavljati listu predmeta u školi. Korisnik može izabrati predmet i na primer upisati ocene i slično. Liste se mogu "puniti" za vreme dizajna, ali je ipak najčešći slučaj punjenje u vreme izvršavanja i to iz neke baze podataka. Za sada ćemo je napuniti pomoću jednostavne petlje iz koda. Ako koristimo objektno orijentisanu terminologiju, možemo reći da je lista kolekcija stavki tipa **Object**.**Kolekcija** označava skup objekata istog tipa. Svaki član kolekcije ima svoj jedinstveni **indeks** koji počinje od broja 0 do ukupnog broja stavki u kolekciji umanjenog za jedan. Bitno je razumeti da kolekcija ima svoja specifična svojstva i metode, ali takođe i svaki objekat u kolekciji. Na primer svaka kolekcija ima svojstvo**Count** koje je samo za čitanje i daje broj objekata u kolekciji. Kolekcije imaju metode kojima se može dodavati nov objekat u kolekciju (**Add**), obrisati pojedinačni objekat (**RemoveAt**) ili obrisati sve članove kolekcije (**Clear**). Svojstva i metode pojedinačnih objekata kolekcije zavise od tipa objekata.

Ako postoji kolekcija pod nazivom "**MojaKolekcija**", metodima i svojstvima te kolekcije pristupamo na sledeći način:

```
MojaKolekcija. Add (...)
MojeKolekcija.Clear ()
```

i slično
Pojedinačnim objektima kolekcije se pristupa preko indeksa objekta:
MojaKolekcija[5].Text

Indeksi objekata – članova kolekcije idu od nula do MojaKolekcija.Count – 1

U **List Box** kontroli ova kolekcija se zove **Items** i sastoji se od članova pod nazivom Item (stavka, element).

Sve kolekcije .NET Framework-a poštuju istu nomenklaturu: kolekcija je imenica u množini a svaki član kolekcije je ista imenica u jednini. Na primer kolekcija **Tables** se sastoji od objekata tipa **Table**. Na glavnu formu dodajte novo dugme i imenujte ga sa btnListBox a Text osobinu postavite na ListBox. Kreirajte zatim novu formu i imenujte je sa ListBoxForm. Na formu postavite **List Box** i jednu **Button**kontrolu sa Text osobinom „Dodaj u listu". Ostavićemo inicijalne nazive ovih kontrola **listBox1** i **button1**. ListBoxForm forma se prikazuje kao rezultat dogadjaja Click na dugme btnListBox.

Želimo da klikom na dugme napunimo listu brojevima od 1 do 10. Koristimo metodu **Add** kolekcije **Items**gde je ulazni parametar generički tip **Object** što znači da možemo "podmetnuti" promenljive numeričkog ili string tipa – svejedno, sve su interno nasleđene od istog **Object** tipa. Duplim klikom na dugme otvorite prozor za pisanje kodai unesite slede'i kod:

```
private void button1_Click(object sender, EventArgs e)
{
int i;
for (i = 1; i <= 5; i++)
{
listBox1.Items.Add(i);
}
}
```

Pokrenite primer i testirajte ga. Primetićete da svaki put kada pritisnete dugme dodajete još 5 stavki u listu. Nakon 3 klika na dugme "Dodaj u listu" lista će izgledati kao što je to prikazano naslici 16.


Slika 16: ListBox kontrola nakon 3 klika na dugme "Dodaj u listu"

Ako želimo da pre dodavanja obrišemo sve stavke (ako ih ima), dodaćemo još jednu liniju koda pre petlje koja briše sadržaj kolekcije **Items**:

```
private void button1_Click(object sender, EventArgs e)
{
```

```
int i; listBox1.Items.Clear();
for (i = 1; i <= 5; i++)

{

listBox1.Items.Add(i);

}

}
```

Sada, pre punjenja liste uvek brišemo sve stavke.
Ako želimo da pročitamo vrednost izabrane stavke iz liste, postoje dva načina:

- jednostavno pročitamo svojstvo **Text**: **listBox1.Text**
- pomoću svojstva **SelectedItem** koja vraća **Object** prezentaciju izabrane stavke:**listBox1.SelectedItem**

(uvek treba proveriti da li je selectedItem različit od null da bi se izbegle eventualne greške!)
Brisanje pojedinačne stavke iz liste se vrši metodom **RemoveAt** kolekcije **Items**. Na primer ako želimo da obrišemo treću stavku u listi:

```
listBox1.Items.RemoveAt (2);
```

**Napomena:** Obratite pažnju da je indeks treće stavke = 2 jer se indeksi broje od nule. Isti tako removeAt ne može da se zove iz petlje.
Svojstvo **SelectedIndex** pokazuje indeks izabrane stavke iz liste. U pitanju je tip **int**, a ako ni jedna stavka nije izabrana vrednost **SelectedIndex** svojstva je jednaka **-1**. Ovo se koristi kao indikacija da korisnik nije izabrao ni jednu stavku iz liste. **SelectedIndex** svojstvo je predviđeno i za čitanje i za pisanje, tako da ako iz koda želimo da izaberemo na primer četvrtu stavku u listi: `listBox1.SelectedIndex = 3;`

Dodajmo još jedno dugme na formu ListBoxForm i za dogadjaj Click za ovo dugme dodajmo kod

koji će da ispiše u MessageBoxu indeks selektovane stavke u ListBox kontroli (prethodno je potrebno napuniti listu klikomna dugme „Dodaj u listu") tj

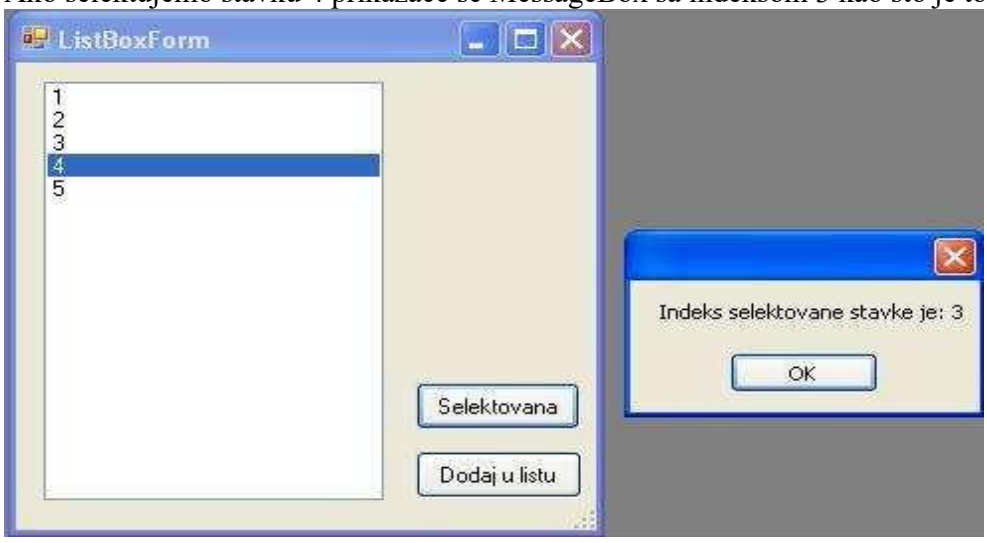
```
private void button2_Click(object sender, EventArgs e)

{

MessageBox.Show("Indeks selektovane stavke je: " +
listBox1.SelectedIndex);

}
```

Ako selektujemo stavku 4 prikazaće se MessageBox sa indeksom 3 kao što je to prikazano na slici 17.




Slika 17: Selektovana četvrta stavka