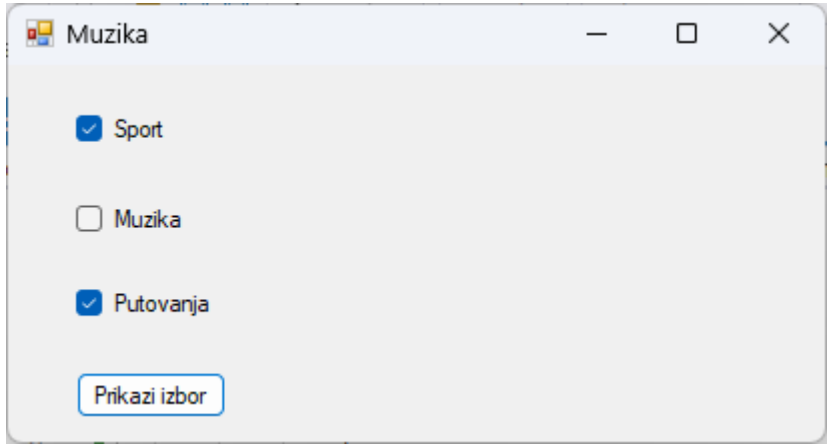
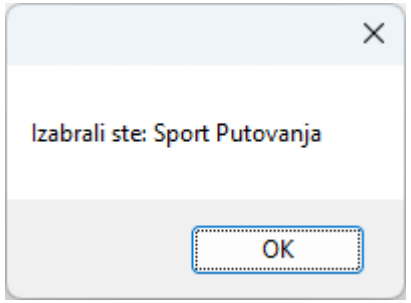
Kontrole CheckBox i RadioButton

CheckBox kontrola

- U jednom trenutku može biti čekirano više od jednog polja za potvrdu(CheckBox kontrole)
- Svojstvo **Text** definiše tekst koji će se pojaviti pored polja za potvrdu
- Svojstvo **Checked** služi za čitanje ili postavljanje (setovanje) stanja CheckBox kontrole
- Događaj **CheckedChanged** se okida kada CheckBox kontrola promeni stanje iz čekiranog u nečekirano ili obrnuto

Upotreba CheckBox kontrole - GUI

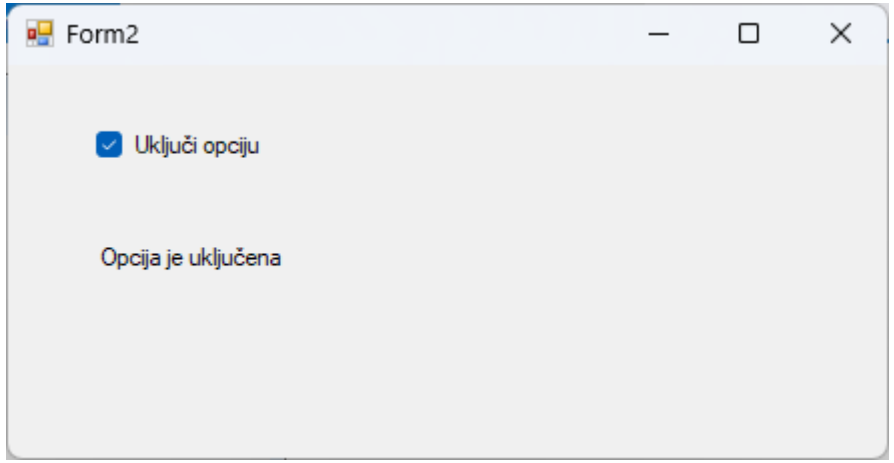




Iščitavanje stanja polja za potvrdu

```
private void button1_Click(object sender, EventArgs e)
{
    string izbor = "";

    if (checkBox1.Checked)
    {
        izbor += "Sport ";
    }

    if (checkBox2.Checked)
    {
        izbor += "Muzika ";
    }

    if (checkBox3.Checked)
    {
        izbor += "Putovanja ";
    }

    if (izbor != "")
    {
        MessageBox.Show("Izabrali ste: " + izbor);
    }
    else
    {
        MessageBox.Show("Niste izabrali nijednu opciju.");
    }
}
```

Obrada CheckedChanged događaja

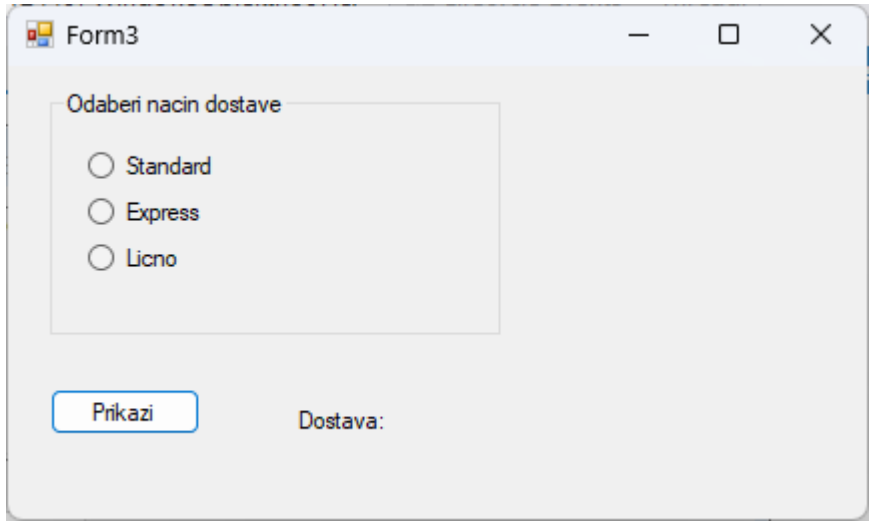



```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        label1.Text = "Opcija je uključena";
    }
    else
    {
        label1.Text = "Opcija je isključena";
    }
}
```

RadioButton kontrola

- Samo jedno radio dugme u istoj grupi može biti čekirano u jednom trenutku
- Svojstvo **Text** definiše tekst koji se prikazuje pored radio dugmeta
- Svojstvo **Checked** služi za čitanje ili setovanje stanja RadioButton kontrole
- Događaj **CheckedChanged** se okida kada RadioButton kontrola promeni stanje (iz čekiranog u dečekirano ili obrnuto)

Iščitavanje stanja RadioButton kontrola




```
private void Form3_Load(object sender, EventArgs e)
{
    rbStandard.Checked = true;
    rbStandard.Checked = false;
}
```

Windows automatski bira prvo RadioButton dugme ako nijedno nije čekirano.
Zato se koristi kratko uključivanje i isključivanje (`true → false`)
da bi grupa ostala prazna pri pokretanju aplikacije.

Iščitavanje stanja RadioButton kontrola

```
private void button1_Click(object sender, EventArgs e)
{
    if (rbStandard.Checked)
    {
        label1.Text = "Izabrali ste: Standard dostavu";
    }
    else if (rbExpress.Checked)
    {
        label1.Text = "Izabrali ste: Express dostavu";
    }
    else if (rbLicno.Checked)
    {
        label1.Text = "Izabrali ste: Lično preuzimanje";
    }
    else
    {
        label1.Text = "Niste izabrali opciju dostave";

    }
}
```

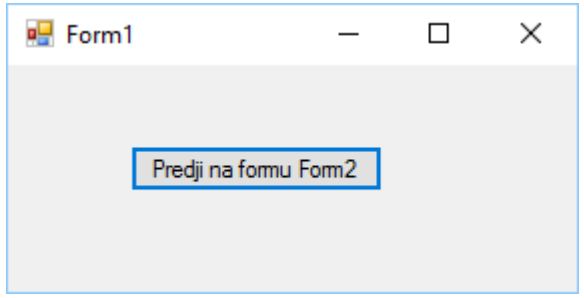
Modalne i nemodalne forme

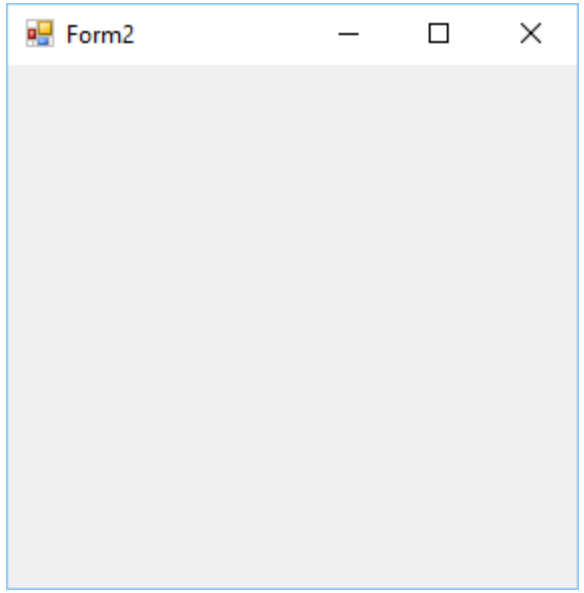
Nemodalna forma

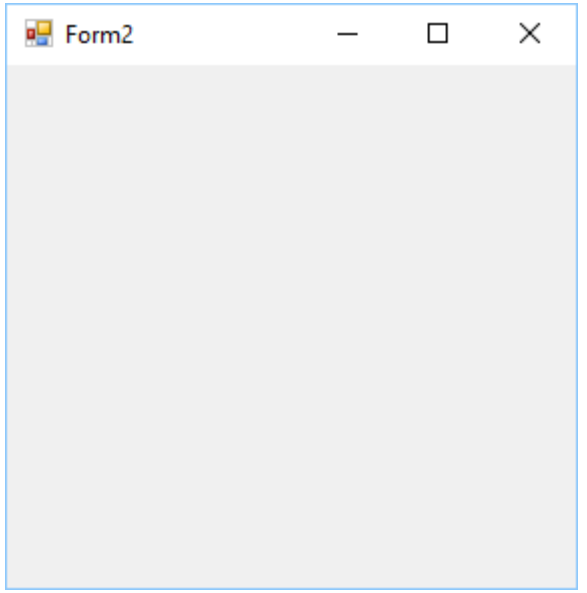
- U aplikacijama koje koriste više formi, forma iz koje se druga forma otvara naziva se parent forma, a forma koja se otvara naziva se child forma
- Nemodalne forme se koriste kada child forma treba da bude otvorena paralelno sa parent formom
- Nemodalna forma je prozor koji se otvara metodom Show() i ne blokira rad glavne forme
- Glavna forma ostaje aktivna i dostupna za korišćenje
- Moguće je otvoriti više instanci iste forme
- Zatvaranjem nemodalne forme njen objekat se uništava (disposed), pa je za sledeće otvaranje potrebna nova instanca forme

Kreiranje nemodalne forme

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.Show();
}
```





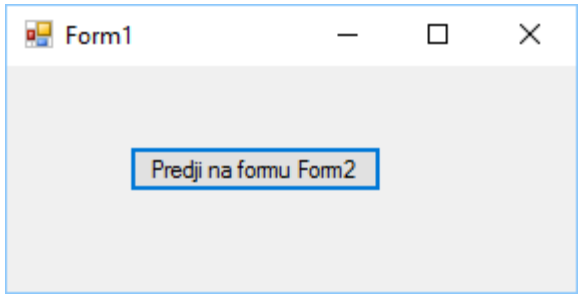


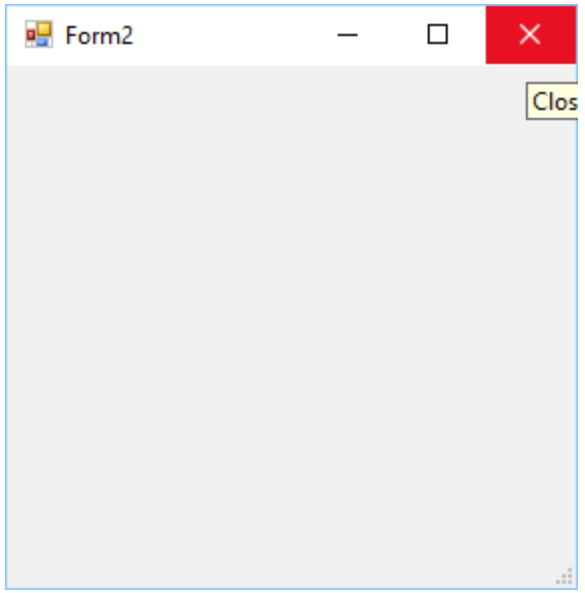
Modalna forma

- Modalna forma se otvara metodom ShowDialog() i blokira rad glavne forme dok je otvorena
- Glavna forma nije dostupna za korišćenje dok se modalna forma ne zatvori
- Koristi se za unos podataka, potvrde i dijaloge (OK/Cancel)
- ShowDialog() vraća vrednost tipa DialogResult, što omogućava proveru kako je forma zatvorena
- Objekat modalne forme se ne uništava automatski zatvaranjem, može se pristupiti njenim podacima nakon zatvaranja

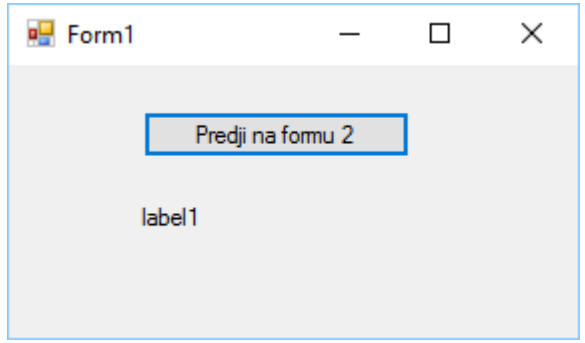
Kreiranje modalne forme

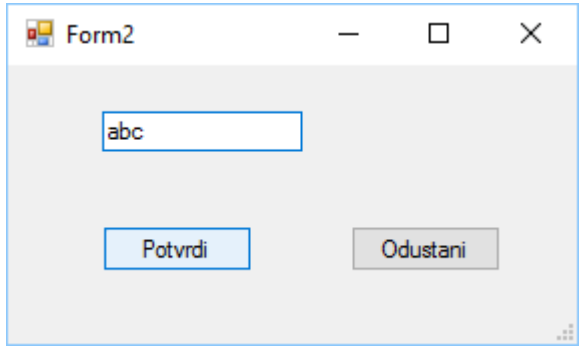
```
private void button1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.ShowDialog();
}
```

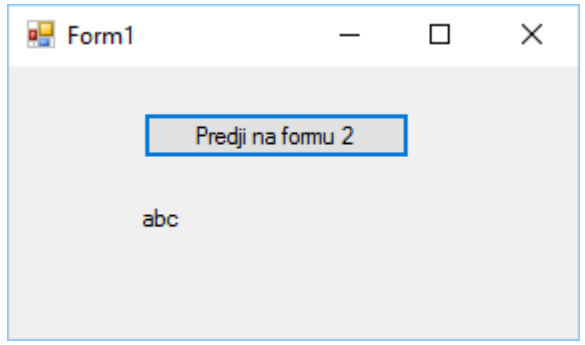




Slanje podataka sa child forme na parent formu







Forma Form2

```
public partial class Form2 : Form
{
    // Podatak koji će biti vraćen parent formi
    public string Podaci { get; set; }
    public Form2()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
    }
    private void button2_Click(object sender, EventArgs e)
    {
    }
}
```

DialogResult enumeracija

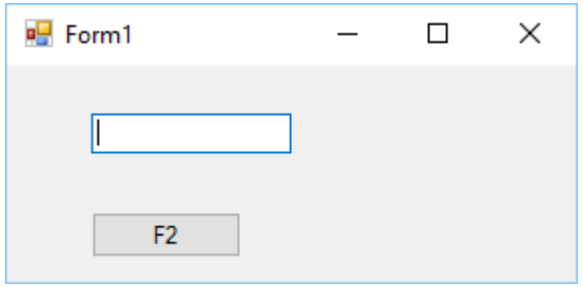
```
private void button1_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(textBox1.Text))
    {
        MessageBox.Show("Unesite podatke");
        textBox1.Focus();
    }
    else
    {
        Podaci = textBox1.Text; // čuvamo podatke u property
        DialogResult = DialogResult.OK; // zatvaramo formu sa rezultatom OK
    }
}
```

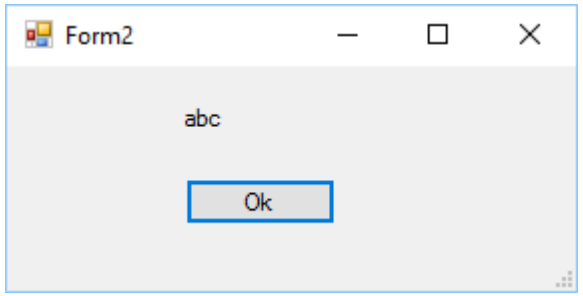
```
private void button2_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.Cancel;
}
```

Forma Form1

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    if (f2.ShowDialog() == DialogResult.OK)
    {
        label1.Text = f2.Podaci; // preuzimanje podataka sa child forme
    }
    else
    {
        label1.Text = "Cancel"; // korisnik je otkazao unos
    }
}
```

Slanje podataka sa parent forme na child formu






```
public partial class Form2 : Form
{
    public string Podaci { get; set; } // podatak koji dolazi iz parent forme
    public Form2()
    {
        InitializeComponent();
    }

}
```

Load procedura forme Form2

```
private void Form2_Load(object sender, EventArgs e)
{
    label1.Text = Podaci; // prikazujemo podatke koje je poslala parent forma
}
```

Kod na formi Form1

```
private void button1_Click(object sender, EventArgs e)
    {
        if (!string.IsNullOrWhiteSpace(textBox1.Text))
        {
            Form2 f2 = new Form2();
            f2.Podaci = textBox1.Text;
            f2.ShowDialog();
        }
        else
        {
            MessageBox.Show("Unesite podatke");
            textBox1.Focus();
        }
    }
}
```

Klasa Object

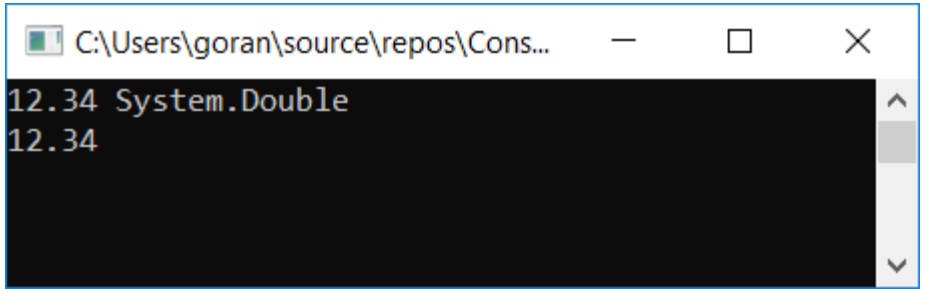
- Svaki objekat u C# izveden je iz bazne klase System.Object
- Nadimak (alias) za ovu klasu je object
- Tipu object može se dodeliti vrednost bilo kog tipa
- **public virtual string ToString():** metoda vraća string koji opisuje instancu klase
- Metoda **GetType**() vraća tip instance objekta

Boxing, unboxing i klasa Type

- **Boxing** je pretvaranje vrednosnog tipa (value type) u referentni tip (object)
- Kod boxinga se vrednost kopira sa steka na heap i pakuje u object
- **Unboxing** je eksplicitno konvertovanje object instance nazad u konkretan vrednosni tip
- Unboxing zahteva kastovanje (cast), a pri neusklađenosti tipova baca se **InvalidCastException**
- Klasa **Type** predstavlja metapodatke o tipu tokom izvršavanja programa
- Metoda **GetType**() klase System.**Object** vraća objekat klase System.Type,koji opisuje instancu — njen naziv, namespace, baznu klasu, članove itd.

Boxing i unboxing

```
static void Main(string[] args)
{
    double a = 12.34;

    object o1 = a; // boxing
    Type t1 = o1.GetType();
    Console.WriteLine("{0} {1}", o1.ToString(), t1.ToString());

    double d = (double)o1; // unboxing
    Console.WriteLine(d);
    Console.ReadLine();
}
```



Boxing – vrednosni tip se smešta u heap i tretira kao object
Unboxing – object se eksplicitno kastuje nazad u vrednosni tip

Rad sa stringovima

Stringovi

- String je nepromenljiva (immutable) sekvenca Unicode karaktera
- Kada metoda “menja” string, ona zapravo vraća novi string, dok original ostaje nepromenjen sve dok ga ne ukloni Garbage Collector
- Stringovi se mogu sortirati, kopirati, konvertovati u druge tipove i mogu se enumerisati pomoću **foreach** petlje(jer string implementira **IEnumerable**)
- Operator + se koristi za nadovezivanje stringova (konkatenaciju)

```
public sealed class String : IComparable, ICloneable, IConvertible, IEnumerable
```

Metode i svojstva klase String

Metoda / Svojstvo	Opis	Primer
Length	Broj karaktera	`int n = s.Length;`
ToLower() / ToUpper()	Mala / velika slova	`s.ToLower();`
Trim()	Skida praznine	`" test ".Trim()`
Contains(str)	Da li sadrži podstring	`s.Contains("abc")`
StartsWith/EndsWith	Početak / kraj stringa	`s.StartsWith("http")`
IndexOf(str)	Prvi indeks podstringa	`s.IndexOf("a")`
Substring(p, d)	Izdvajanje dela stringa	`s.Substring(2,3)`
Replace(a,b)	Zamena teksta	`s.Replace("lo","XX")`
Split(separator)	Razdvajanje	`s.Split(',')`
Join(sep,array)	Spajanje	`String.Join("-", arr)`
Equals(str)	Poređenje stringova	`s.Equals("OK")`
IsNullOrEmpty(str)	null ili ""	`String.IsNullOrEmpty(s)`
IsNullOrWhiteSpace(str)	null/prazno/beline	`String.IsNullOrWhiteSpace(s)`
ToCharArray()	U niz karaktera	`char[] c = s.ToCharArray()`

String vrednost	Metoda / Kod	Rezultat
original = " Hello world from C# "	**s = original.Trim()**	`"Hello world from C#"`
s = "Hello world from C#"	`s.Length`	`20`
s = "Hello world from C#"	`s.ToUpper()`	`HELLO WORLD FROM C#`
s = "Hello world from C#"	`s.StartsWith("Hello")`	`True`
s = "Hello world from C#"	`s.EndsWith("C#")`	`True`
s = "Hello world from C#"	`s.Contains("world")`	`True`
s = "Hello world from C#"	`s.IndexOf("o")`	`4`
s = "Hello world from C#"	`s.Replace("world", "C#")`	`Hello C# from C#`
s = "Hello world from C#"	`s.Substring(6)`	`"world from C#"`
s = "Hello world from C#"	`s.Substring(6, 5)`	`"world"`
s = "Hello world from C#"	`delovi = s.Split(' ')`	`"Hello","world","from","C#"`
s = "Hello world from C#"	`string.Join(" | ", delovi)`	`Hello | world | from | C#`
s = "Hello world from C#"	`string.IsNullOrWhiteSpace(s)`	`False`

Promena stringa u petlji

```
private void button1_Click(object sender, EventArgs e)
{
    string s = "test";
    for (int i = 0; i < 100; i++)
    {
        s += i.ToString();
    }
    richTextBox1.Text = s.ToString();
}
```

Osobina nepromenljivosti stringova

- Stringovi su nepromenljivi (immutable) — kada se jednom kreiraju, njihova memorijska lokacija se više ne menja.
- Linija koda s += i.ToString(); kreira novi string, a rezultat se čuva na novoj memorijskoj lokaciji.
- I stara i nova verzija stringa kratko vreme koegzistiraju u memoriji.
- Stara verzija stringa biće obrisana tokom narednog ciklusa garbage collection-a.
- Ako aplikacija često menja stringove, u memoriji se nakupljaju privremeni objekti koji čekaju GC, što može usporiti program.
- Rešenje je korišćenje klase **StringBuilder** (namespace System.Text), jer omogućava efikasno menjanje teksta bez kreiranja novih objekata

Klasa StringBuilder

- **StringBuilder** alocira početni kapacitet (obično 16 karaktera)
- Kako se sadržaj povećava, njegov interni bafer se automatski proširuje da bi mogao da primi novi tekst (slično kao kod kolekcija koje povećavaju kapacitet)
- Za razliku od tipa string, koji je nepromenljiv (immutable) i za svaku izmenu kreira novi objekat, StringBuilder menja postojeći sadržaj u memoriji, što ga čini mnogo efikasnijim kada se tekst menja u petlji ili često nadograđuje
- Klasa StringBuilder se nalazi u namespace-u **System.Text**

StringBuilder objekat u petlji

```
private void button1_Click(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder("test");

    for (int i = 0; i < 100; i++)
    {
        sb.Append(i.ToString());
    }

    richTextBox1.Text = sb.ToString();
}
```

Metode klase StringBuilder

Metoda / Svojstvo	Opis / Namena	Primer
Append(text)	Dodaje tekst na kraj postojećeg sadržaja	`sb.Append(" world");`
AppendLine(text)	Dodaje tekst i prelazak u novi red	`sb.AppendLine("Linija");`
Insert(index, text)	Ubacuje tekst na određenu poziciju	`sb.Insert(6, "C# ");`
Remove(start, length)	Briše deo teksta (od start, length karaktera)	`sb.Remove(6, 3);`
Replace(old, new)	Menja sve instance podstringa	`sb.Replace("na", "XY");`
Clear()	Briše ceo sadržaj	`sb.Clear();`
ToString()	Vraća finalni string	`sb.ToString();`
Length	Broj karaktera u sadržaju	`int len = sb.Length;`

Pitanje 1

Stanje CheckBox kontrole se isčitava korišćenjem :

a. metode Selected()
b. svojstva Checked
c. metode Checked()
d. svojstva IsSelected

Odgovor: b

Pitanje 2

Unutar GroupBox kontejnera se nalazi 5 RadioButton kontrola. U jednom trenutku može biti selektovano:

a. samo jedno radio dugme
b. najviše 5 radio dugmeta
c. najviše 2 radio dugmeta

Odgovor: a

Pitanje 3

Da li je moguće kreirati više modalnih formi (koristi se metoda ShowDialog()) jedne iste parent forme koje ce egzistirati istovremeno?

a. zavisi od konteksta u kome radi aplikacija
b. da
c. ne

Odgovor: c

Pitanje 4

Unutar Click dodadaja za dugme na formi Form1 instancira se forma Form2 na sledeci nacin:

```
Form2 f2 = new Form2();
f2.Show();
```

Nakon što je korisnik kliknuo na dugme i prikazala mu se druga forma on želi da se vrati na pocetnu formu. Korisnik:

a. mora da zatvori formu f2
b. ne mora da zatvori formu f2
c. zavisi od konteksta u kome radi aplikacija

Odgovor: b

Pitanje 5

Da li se zatvaranjem modalne forme, objekat forme uništava?

a. Da
b. Ne

Odogovor: b

Pitanje 6

Neka je Stampaj() metoda koja služi da prikaže tekst na korisničkom interfejsu windows aplikacije. Aplikacija izvršava sledeće linije koda:

```
string s = "Pera Peric";
Stampaj(s.Substring(5, 3));
```

Šta se ispisuje na korisničkom interfejsu?

a. Pera Pera Pera
b. Per
c. Peric
d. Peri

Odgovor: b

Pitanje 7

Koja klasa u .NET-u se koristi za kreiranje dinamičkog stringa, tj. stringa koji može efikasno da menja svoju dužinu i sadržaj bez kreiranja novih instanci?

a. DinamicString
b. StringBuffer
c. String
d. StringBuilder

Odgovor: d

Pitanje 8

Da li je u C# moguće kreirati klasu koja je izvedena iz sistemske klase String?

a. zavisi od konteksta u kome radi aplikacija
b. ne
c. da

Odgovor: b