

Lambda izrazi

Delegat `Func<T,TResult>`

- Delegat je tip podataka koji predstavlja referencu na metodu
- Delegat **`Func<TResult>`** je pokazivač na metodu bez ulaznih parametar
- Delegat **`Func<T,TResult>`** je pokazivač na metodu koja ima jedan ulazni parametar tipa T i vraća rezultat tipa TResult
- Delegat **`Func<T1,T2,TResult>`** je pokazivač na metodu sa dva ulazna parametra tipa T1 i T2 koja ima povratnu vrednost tipa TResult

Lambda izrazi

- Anonimna metoda je metoda koja nema ime, može imati ulazne parametre i može vraćati vrednost
- Lambda izrazi omogućavaju kreiranje anonimnih metoda
- Lambda izrazi se definišu korišćenjem => operatora
- Lambda izrazi imaju prirodnu i preciznu sintaksu

Primer Lambda izraza

$$x \Rightarrow x * x$$

Za dato x, izračunaj $x * x$

Visual C# izvodi povratni tip na osnovu tela metode, konteksta

```
private void Button1_Click(object sender, EventArgs e)
{
    Func<double,double> f1 = x => x * x;

    double rezultat = f1(5);

    MessageBox.Show(rezultat.ToString());
}
```

Lambda izraz za metodu sa dva ulazna parametra

```
private void Button2_Click(object sender, EventArgs e)
{
    Func<int, int, int> f1 = (x, y) => x + y;
    int rezultat = f1(5, 6);

    MessageBox.Show(rezultat.ToString());
}
```

Lambda izraz za metodu bez parametara

```
private void Button3_Click(object sender, EventArgs e)
{
    Func<string> f1 = () => DateTime.Now.ToShortDateString();

    string rezultat = f1();

    MessageBox.Show(rezultat);
}
```

List<T>.ForEach(Action<T>) metoda

```
private void button1_Click(object sender, EventArgs e)
{
    List<string> imena = new List<string>
    {
        "Marko",
        "Ana",
        "Lazar",
        "Dejan"
    };

    imena.ForEach(i => richTextBox1.AppendText(i + "\n"));
}
```

Delegat Action<T> može da pokazuje na metodu sa ulaznim parametrom tipa T koja nema povratnu vrednost

Ekstenzione metode

- Ekstenziona metoda omogućava da dodajemo metode u postojeću klasu bez potrebe da kreiramo izvedenu klasu
- To su statičke metode ali se pozivaju kao da su nestatičke metode klase koju proširujemo
- Definiše se statička klasa koja će sadržati ekstenzione metode
- Klasa mora biti vidljiva za kod koji će je pozivati

Kreiranje ekstenzione metode

- Ekstenziona metoda je statička metoda sa najmanje istom vidljivošću kao i klasa koja je sadrži
- Prvi parametar ekstenzione metode je tip sa kojim metoda radi ispred koga se nalazi reč this
- Ovakva metoda se poziva kao da je instance metoda tipa koga proširuje

Primer ekstenzione metode koja proširuje klasu string

```
static class EkstenzioneMetode
{
    public static string VelikoPrvoSlovo(this string s)
    {
        s = s.Trim().ToLower();
        if (s.Length > 1)
        {
            return s.Substring(0, 1).ToUpper() + s.Substring(1);
        }
        return string.Empty;
    }
}
```

Poziv ektenzione metode

```
private void button1_Click(object sender, EventArgs e)
{
    string s1 = "MARKO";
    string ime = s1.VelikoPrvoSlovo();
    richTextBox1.Text = ime;
}
```

Marko

Klasa Enumerable

```
public static class Enumerable
```

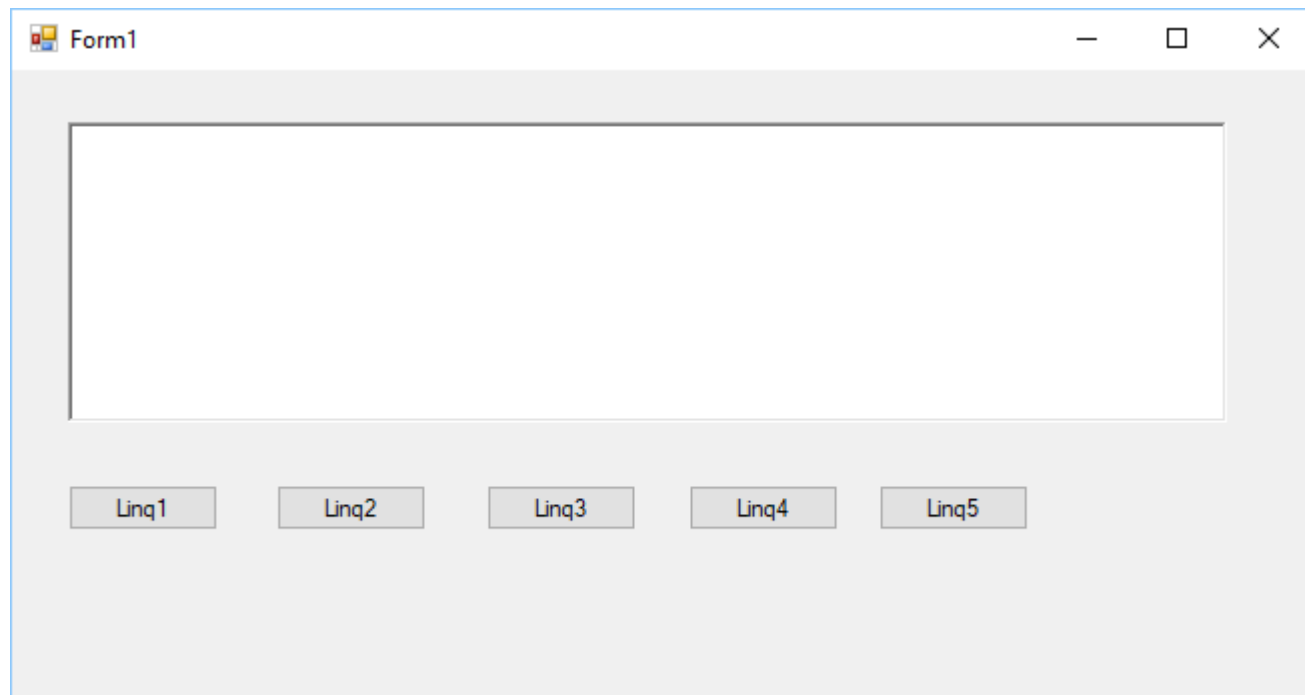
- Nalazi se u prostoru imena System.Linq
- TSource generički tip koga sadrži izvor podataka
- Statičke metode klase Enumerable su ekstenzione metode koje proširuju izvor podataka koji implementira IEnumerable<TSource> intefejs
- Statičke metode klase Enumerable omogućavaju da se pišu upiti nad bilo kojim izvorom podataka koji primenjuje interfejs IEnumerable<T>

Ekstenziona metoda Select klase Enumerable

```
public static IEnumerable<TResult> Select<TSource, TResult>(
    this IEnumerable<TSource> source,
    Func<TSource, TResult> selector
)
```

- Ekstenziona metoda **Select** proširuje tip **IEnumerable<TSource>**
- **source je** izvor podataka koji implementira interfejs **IEnumerable**
- **selector** je anonimna transformaciona funkcija koja se primenjuje nad svakim članom u izvoru podataka
- Povratna vrednost je sekvenca nastala primenom transformacione funkcije nad izvorom podataka

Korisnički interfejs



Primer upotrebe ekstenzione metode Select()

```
private void Button1_Click(object sender, EventArgs e)
{
    int[] brojevi = { 1, 3, 5, 15, 9, 24 };

    IEnumerable<int> kvadrati = brojevi.Select(x => x * x);
    StringBuilder sb = new StringBuilder();
    foreach (int i in kvadrati)
    {
        sb.AppendLine(i.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

Ekstenziona metoda Where klase Enumerable

```
public static IEnumerable<TSource> Where<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

- Where ekstenziona metoda proširuje tip **IEnumerable<TSource>**
- **source** je izvor podataka koji implementira interfejs **IEnumerable**
- Where ekstenziona metoda kao parametar ima instancu delegata **Func<TSource, bool>**, ova instanca je označena sa **predicate**
- Parametar **predicate** može biti bilo kakav lambda izraz (anonimna funkcija) sa ulazom tipa **TSource** i izlazom tipa **bool**

Primer upotrebe Where ekstenzione metode -1

```
private void Button2_Click(object sender, EventArgs e)
{
    int[] brojevi = { 0, 1, 2, 3, 4, 5, 6 };

    // definisanje upita
    IEnumerable<int> upit = brojevi.Where(n => n % 2 == 0);

    StringBuilder sb = new StringBuilder();

    foreach (int i in upit)
    {
        sb.AppendLine(i.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

Primer upotrebe Where ekstenzione metode -2

```
private void Button3_Click(object sender, EventArgs e)
{
    string[] imena = { "Marko", "Lazar", "Ivan", "Marija", "Jovana", "Jovan", "Desimir" };

    IEnumerable<string> upit = imena
        .Where(s => s.StartsWith("m", StringComparison.CurrentCultureIgnoreCase));

    StringBuilder sb = new StringBuilder();

    foreach (string n in upit)
    {
        sb.AppendLine(n);
    }

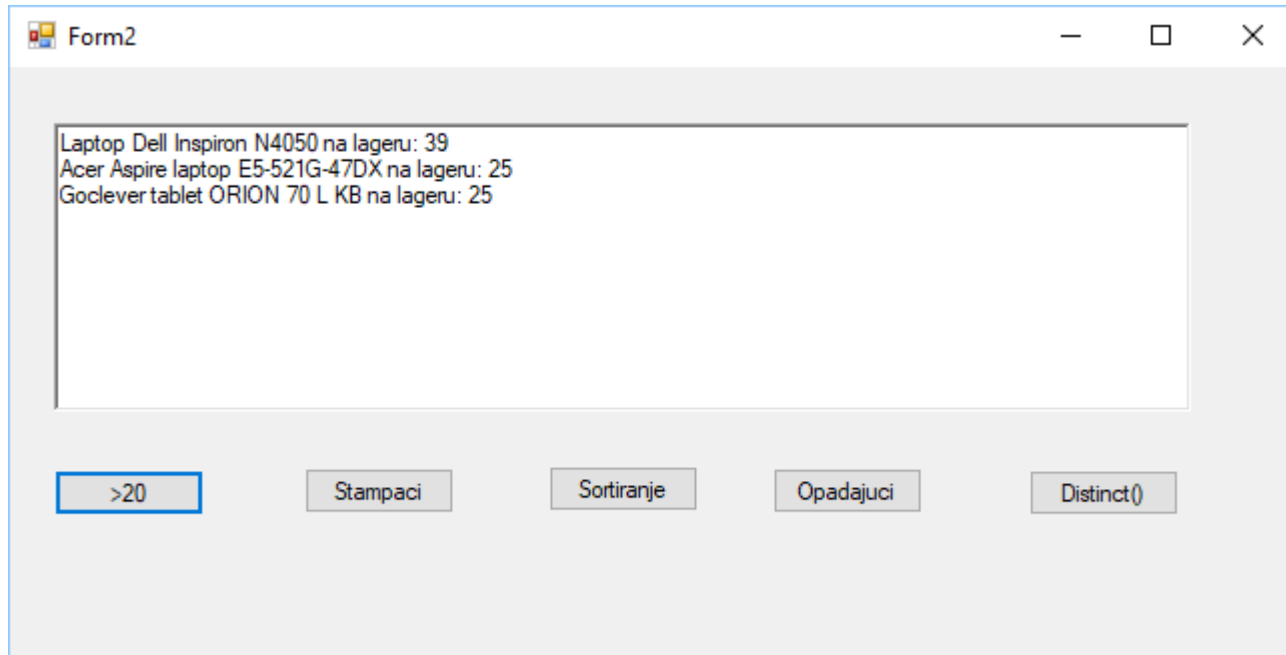
    richTextBox1.Text = sb.ToString();
}
```

Kombinovanje metoda Where() i Select()

```
private void Button4_Click(object sender, EventArgs e)
{
    string[] imena = { "Aca", "Pera", "Mika", "Lazar", "Ivana", "Jovan", "Jovana", "Djordje" };
    IEnumerable<string> upit = imena
        .Where(s => s.Length == 5)
        .Select(s => s.ToUpper());
    StringBuilder sb = new StringBuilder();

    foreach (string clan in upit)
    {
        sb.AppendLine(clan);
    }
    richTextBox1.Text = sb.ToString();
}
```

Linq upiti nad kolekcijama objekata



Klasa Kategorija

```
class Kategorija
{
    public int KategorijaId { get; set; }
    public string NazivKategorije { get; set; }
    public string OpisKategorije { get; set; }
}
```

Klasa Proizvod

```
class Proizvod
{
    public int ProizvodId { get; set; }
    public int KategorijaId { get; set; }
    public string NazivProizvoda { get; set; }
    public decimal Cena { get; set; }
    public int KolicinaNaLageru { get; set; }
}
```

Metoda Vratikategorije(), statička klasa Podaci

```
public static List<Kategorija> Vratikategorije()
{
    List<Kategorija> listaKategorija = new List<Kategorija>() {
        new Kategorija{KategorijaId=1,NazivKategorije="Laptopovi", OpisKategorije="Laptopovi razlicitih proizvođaca" },
        new Kategorija{KategorijaId=2,NazivKategorije="Stampaci", OpisKategorije="Stampaci razlicitih proizvođaca" },
        new Kategorija{KategorijaId=3,NazivKategorije="Tableti", OpisKategorije="Tablet racunari razlicitih proizvođaca" }
    };
    return listaKategorija;
}
```

Metoda VratiProizvode() statička klasa Podaci

```
public static List<Proizvod> VratiProizvode()
{
    List<Proizvod> listaProizvoda = new List<Proizvod>() {
        new Proizvod{ProizvodId=1, KategorijaId=1,NazivProizvoda="Laptop Dell Inspiron N4050",Cena=30999.25M,KolicinaNaLageru=39 },
        new Proizvod{ProizvodId=2, KategorijaId=1,NazivProizvoda="Laptop Asus X55U-SX009D",Cena=32990.12M,KolicinaNaLageru=17 },
        new Proizvod{ProizvodId=3, KategorijaId=1,NazivProizvoda="Acer Aspire laptop E5-521G-47DX",Cena=41989,KolicinaNaLageru=25 },

        new Proizvod{ProizvodId=4, KategorijaId=2,NazivProizvoda="Stampač Laser A4 Lexmark E260",Cena=8549.1M,KolicinaNaLageru=13 },
        new Proizvod{ProizvodId=5, KategorijaId=2,NazivProizvoda="Canon laserski stampac LBP-6670DN",Cena=31989.1M,KolicinaNaLageru=18 },
        new Proizvod{ProizvodId=6, KategorijaId=2,NazivProizvoda="Canon stampač imageCLASS LBP6030W",Cena=13589.12M,KolicinaNaLageru=11 },

        new Proizvod{ProizvodId=7, KategorijaId=3,NazivProizvoda="Acer tablet B1-730HD 8GB",Cena=11999.3M,KolicinaNaLageru=12 },
        new Proizvod{ProizvodId=8, KategorijaId=3,NazivProizvoda="Asus tablet MeMO Pad 7 ME70C-1A003A",Cena=12999.9M,KolicinaNaLageru=14 },
        new Proizvod{ProizvodId=9, KategorijaId=3,NazivProizvoda="Goclever tablet ORION 70 L KB",Cena=5699.45M,KolicinaNaLageru=25 }
    };

    return listaProizvoda;
}
```

Inicijalizacija izvora podataka

```
private List<Kategorija> listaKategorija = Podaci.VratiKategorije();  
private List<Proizvod> listaProizvoda = Podaci.VratiProizvode();
```

Proizvodi kojih na lageru ima više od 20

```
private void Button1_Click(object sender, EventArgs e)
{
    IEnumerable<Proizvod> upit = listaProizvoda
        .Where(p => p.KolicinaNaLageru > 20);

    StringBuilder sb = new StringBuilder();

    foreach (Proizvod p1 in upit)
    {
        sb.AppendLine("Naziv: " + p1.NazivProizvoda);
        sb.AppendLine("Cena: " + p1.Cena);
        sb.AppendLine("Kolicina: " + p1.KolicinaNaLageru);
        sb.AppendLine("".PadRight(100, '.'));
    }

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

Ekstenziona metoda SingleOrDefault() klase Enumerable

```
public static TSource SingleOrDefault<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

Vraća jedinstveni element sekvence koji zadovoljava uslov.
Vraća podrazumavanu vrednost za element u izvoru podataka
ukoliko ne postoji element.

Metoda SingleOrDefault() upotreba

```
private void Button2_Click(object sender, EventArgs e)
{
    Proizvod p1 = listaProizvoda.SingleOrDefault(p => p.ProizvodId == 1);

    if (p1 != null)
    {
        label1.Text = p1.NazivProizvoda;
    }
    else
    {
        MessageBox.Show("Ne postoji proizvod");
    }
}
```

Metoda FirstOrDefault()

```
public static TSource FirstOrDefault<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

Vraća prvi element sekvence koji zadovoljava uslov ili podrazumevanu vrednost ukoliko takav element ne postoji

Metoda FirstOrDefault upotreba

```
private void Button3_Click(object sender, EventArgs e)
{
    Proizvod p1 = listaProizvoda.FirstOrDefault(p => p.KategorijaId == 1);
    if (p1 != null)
    {
        label1.Text = p1.NazivProizvoda;
    }
    else
    {
        MessageBox.Show("Ne postoji proizvod");
    }
}
```

Metoda OrderBy

```
public static IOrderedEnumerable<TSource> OrderBy<TSource, TKey>(
    this IEnumerable<TSource> source,
    Func<TSource, TKey> keySelector
)
```

Sortira elemente sekvence u rastućem poretku prema ključu
keySelektor - anonimna funkcija koja izdvaja ključ iz svakog elementa sekvence

Sortiranje rezultata

```
private void Button4_Click(object sender, RoutedEventArgs e)
{
    IEnumerable<Proizvod> sortiraniProizvodi = listaProizvoda
        .Where(p => p.KategorijaId == 1)
        .OrderBy(p => p.Cena);

    StringBuilder sb = new StringBuilder();
    foreach (Proizvod p in sortiraniProizvodi)
    {
        sb.AppendLine(p.NazivProizvoda + " " + p.Cena);
    }
    richTextBox1.Text = sb.ToString();
}
```

Sortiranje u opadajućem poretku

```
private void Button5_Click(object sender, RoutedEventArgs e)
{
    IEnumerable<Proizvod> sortiraniProizvodi = listaProizvoda
    .Where(p => p.KategorijaId == 1)
    .OrderByDescending(p => p.Cena);

    StringBuilder sb = new StringBuilder();
    foreach (Proizvod p in sortiraniProizvodi)
    {
        sb.AppendLine(p.NazivProizvoda + " " + p.Cena);
    }
    richTextBox1.Text = sb.ToString();
}
```

Višestruko sortiranje

```
private void Button9_Click(object sender, EventArgs e)
{
    IEnumerable<Proizvod> upit = listaProizvoda
        .OrderBy(p => p.KategorijaId)
        .ThenByDescending(p => p.Cena);

    StringBuilder sb = new StringBuilder();
    foreach (Proizvod p in upit)
    {
        sb.AppendLine(p.KategorijaId + " " + p.NazivProizvoda + " " + p.Cena);
    }
    richTextBox1.Text = sb.ToString();
}
```

Metoda Distinct()

Vraća jedinstvene elemente sekvence korišćenjem podrazumevanog komparatora jednakosti.

```
private void Button1_Click(object sender, EventArgs e)
{
    List<int> brojevi = new List<int> { 21, 46, 46, 55, 17, 21, 55, 55 };
    IEnumerable<int> razlicitiBrojevi = brojevi.Distinct();

    StringBuilder sb = new StringBuilder();
    foreach (int b in razlicitiBrojevi)
    {
        sb.AppendLine(b.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

Metoda Count()

```
public static int Count<TSource>(
    this IEnumerable<TSource> source
)
```

```
public static int Count<TSource>(
    this IEnumerable<TSource> source,
    Func<TSource, bool> predicate
)
```

Metoda Count()

```
private void Button2_Click(object sender, EventArgs e)
{
    List<int> brojevi = new List<int>{ 5, 4, 1, 3, 9, 8, 6, 7, 1, 0 };
    int ukupno = brojevi.Count();
    int neparnih = brojevi.Count(b => b % 2 == 1);
    int parnih = brojevi.Count(b => b % 2 == 0);

    StringBuilder sb = new StringBuilder();
    sb.AppendLine("Ukupno: " + ukupno.ToString());
    sb.AppendLine("Neparnih: " + neparnih.ToString());
    sb.AppendLine("Parnih: " + parnih.ToString());
    richTextBox1.Text = sb.ToString();
}
```

Metoda Min()

```
public static int Min(  
    this IEnumerable<int> source  
)
```

```
public static decimal Min<TSource>(  
    this IEnumerable<TSource> source,  
    Func<TSource, decimal> selector  
)
```

selector - transformaciona funkcija koja svaku vrednost sekvence pretvara u decimal

Metoda Max()

```
public static int Max(  
    this IEnumerable<int> source  
)
```

```
public static decimal Max<TSource>(  
    this IEnumerable<TSource> source,  
    Func<TSource, decimal> selector  
)
```

Primer Min()/Max()

```
private void Button3_Click(object sender, EventArgs e)
{
    int[] brojevi = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
    StringBuilder sb = new StringBuilder();

    int minBroj = brojevi.Min();
    sb.AppendLine(minBroj.ToString());

    decimal najJeftiniji = listaProizvoda.Min(p => p.Cena);
    sb.AppendLine(najJeftiniji.ToString());

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

Metoda Average()

```
private void Button4_Click(object sender, EventArgs e)
{
    List<int> brojevi = new List<int> { 78, 92, 100, 37, 81 };

    StringBuilder sb = new StringBuilder();

    double prosek = brojevi.Average();
    sb.AppendLine(prosek.ToString());

    decimal prosecnaCena = listaProizvoda.Average(p => p.Cena);
    sb.AppendLine(prosecnaCena.ToString());

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

Metoda Sum()

```
private void Button5_Click(object sender, RoutedEventArgs e)
{
    StringBuilder sb = new StringBuilder();

    List<int> brojevi = new List<int> { 78, 92, 100, 37, 81 };
    int suma = brojevi.Sum();
    sb.AppendLine(suma.ToString());

    int ukupnoLaptopova = listaProizvoda
        .Where(p => p.KategorijaId == 1)
        .Sum(p => p.KolicinaNaLageru);

    sb.AppendLine(ukupnoLaptopova.ToString());

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

Metoda Take()

```
public static IEnumerable<TSource> Take<TSource>(
    this IEnumerable<TSource> source,
    int count
)
```

Vraća specificirani broj uzastopnih elemenata od početka sekvence.

Metoda Take() - primer

```
private void Button6_Click(object sender, EventArgs e)
{
    int[] poeni = { 59, 82, 70, 56, 92, 98, 85 };

    IEnumerable<int> najbolja3 =
        poeni.OrderByDescending(p => p).Take(3);

    StringBuilder sb = new StringBuilder();
    foreach (int p in najbolja3)
    {
        sb.AppendLine(p.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

Metoda Skip()

Preskače određeni broj elemenata sa početka sekvence.

```
private void Button7_Click(object sender, EventArgs e)
{
    int[] poeni = { 59, 82, 70, 56, 92, 98, 85 };

    IEnumerable<int> preskociPrvaTri =
        poeni.OrderByDescending(p => p).Skip(3);

    StringBuilder sb = new StringBuilder();
    foreach (int p in preskociPrvaTri)
    {
        sb.AppendLine(p.ToString());
    }
    richTextBox1.Text = sb.ToString();
}
```

Pitanje 1

Delegat je tip koji:

- a. pokazuje na metodu
- b. pokazuje na klasu
- c. pokazuje na svojstvo

Odgovor: a

Pitanje 2

Pomoću lambda izraza definiše se:

- a. anonimna metoda
- b. sql upit
- c. metoda koja ima svoje ime

Odogovor: a

Pitanje 3

Za pisanje upita primenom lambda izraza koriste se

- a. Metode klase Linq
- b. Ekstenzione metode klase Enumerable
- c. Metode interfejsa IEnumerable

Odogovor: b

Pitanje 4

Ako je sa TSource označen tip podataka u izvoru podataka. Ekstenzione metode statičke klase Enumerable proširuju :

- a. Izvor podataka koji implementira IEnumerable<TSource> interfejs
- b. Izvor podataka koji implementira ISqlSource<TSource> interfejs
- c. Izvor podataka koji implementira IComparable<TSource> interfejs

Odogovor: a