Programmering

# Lambda

|  |  |
| --- | --- |
| **Navn** | Ex59 – Lambda |
| **Læringsmål** | * At genopfriske navngivne og anonyme delegate-typer * At forstå brugen af de indbyggede Action, Func og Predicate delegate-typer * At have forståelse for og kunne implementere lambda-udtryk |
| **Forventet resultat** | * En praktisk forståelse af delegates og lambda-udtryk * Løsning af opgavens øvelser |

I denne opgave skal I arbejde mere i dybden med lambda-udtryk og dets anvendelse i forskellige sammenhænge. I har sikkert allerede lagt mærke til lambda-udtryk, som kan anvendes stort set alle steder, hvor man bruger delegates. Dette vil I se flere varianter af i opgaven. Derudover finder lambda-udtryk udbredt anvendelse i LINQ (.NET Language-Integrated Query), hvilket I også skal se nærmere på i en senere opgave.

Bemærk at denne opgave flere steder tager udgangspunkt i vedlagte kode (BonusApp.Code.zip), som er en mulig løsning til øvelse 1 i den tidligere opgave Ex37-Delegates.

# Forberedelse

Brug **Ordet Rundt** og diskutér i gruppen, hvad du fik ud af forberedelsen til dagens opgave:

* Hvad var let og svært at forstå?
* Har I forstået materialet på samme måde?
* Hvad er I enige og uenige om?

*Tidsramme: 10 minutter*

# Øvelse 1: Metodesignaturer og delegate-typer

Kig nøje på følgende 13 metodesignaturer:

1. double Calculate(string expression) Func<string, double>
2. int CompareTo(int other) Func<int,int>
3. int Factorial (int n) Func<int, int>
4. void Delete(int index) Action<int>
5. bool Exist(int id) Predicate<int>
6. string GetUserChoice() Func<string>
7. int Fibonacci(int n) Func<int, int>
8. void NotifySubscribers(string message) Action<string>
9. string GetAllPets() Func<String>
10. bool IsMatchesFinished() Func<bool>
11. void ShowScore(string tournamentName) Action<string>
12. bool Contains(int id) Predicate<int>
13. int RectangleArea(int height, int width) Func<int,int,int>

## Øvelse 1.1: Beskriv som delegate-typer

Udfør følgende:

* Definér så få delegate-typer som muligt, der beskriver alle 13 metodesignaturer. Find selv på navnene på de delegate-typer, I finder.
* Konvertér hver af de fundne delegate-typer til de indbyggede delegate-typer: *Action*, *Func* og *Predicate*. Nogle delegate-typer kan beskrives med mere end en af de indbyggede. Find dem alle.
* Overvej om to eller flere forskellige delegate-typer (indbyggede eller brugerdefinerede), der beskriver den samme metodesignatur, er kompatible.

*Tidsramme: 30 minutter*

## Øvelse 1.2: Opsamling

* Benyt ”30 til buffet” til at opsamle og reflektere over jeres fund.

*Tidsramme: 10 minutter*

# Øvelse 2: Anonyme delegates i BonusApp

Udfør følgende:

* Pak koden i BonusApp.Code.zip ud og åbn løsningen i Visual Studio.
* Opret et nyt testprojekt og indsæt nedenstående i projektet:

[TestClass]

public class UnitTest1

{

Order order;

[TestInitialize]

public void InitializeTest()

{

order = new Order();

order.AddProduct(new Product

{

Name = "Mælk",

Value = 10.0

});

order.AddProduct(new Product

{

Name = "Smør",

Value = 15.0

});

order.AddProduct(new Product

{

Name = "Pålæg",

Value = 20.0

});

}

[TestMethod]

public void TenPercent\_Test()

{

Assert.AreEqual(4.5, Bonuses.TenPercent(45.0));

Assert.AreEqual(40.0, Bonuses.TenPercent(400.0));

}

[TestMethod]

public void FlatTwoIfAMountMoreThanFive\_Test()

{

Assert.AreEqual(2.0, Bonuses.FlatTwoIfAmountMoreThanFive(10.0));

Assert.AreEqual(0.0, Bonuses.FlatTwoIfAmountMoreThanFive(4.0));

}

[TestMethod]

public void GetValueOfProducts\_Test()

{

Assert.AreEqual(45.0, order.GetValueOfProducts());

}

[TestMethod]

public void GetBonus\_Test()

{

order.Bonus = Bonuses.TenPercent;

Assert.AreEqual(4.5, order.GetBonus());

order.Bonus = Bonuses.FlatTwoIfAmountMoreThanFive;

Assert.AreEqual(2.0, order.GetBonus());

}

[TestMethod]

public void GetTotalPrice\_Test()

{

order.Bonus = Bonuses.TenPercent;

Assert.AreEqual(40.5, order.GetTotalPrice());

order.Bonus = Bonuses.FlatTwoIfAmountMoreThanFive;

Assert.AreEqual(43.0, order.GetTotalPrice());

}

}

* Kig koden igennem, kompilér den, og kør testen. Der skulle ikke være nogen fejl.

Order-klassen indeholder en Bonus-property, som er erklæret med delegate-typen *BonusProvider* (defineret i *Bonuses.cs*). Det er denne property og dens type, vi har fokus på i det følgende:

* Indsæt følgende testmetode:

[TestMethod]

public void GetBonusAnonymous\_Test()

{

order.Bonus = Bonuses.TenPercent; // <- Change to anonymous delegate

Assert.AreEqual(4.5, order.GetBonus());

order.Bonus = Bonuses.FlatTwoIfAmountMoreThanFive; // <- Change to anonymous delegate

Assert.AreEqual(2.0, order.GetBonus());

}

* Tilret koden, så Bonuses.TenPercent og Bonuses.FlatTwoIfAmountMoreThanFive ændres til en tilsvarende in-line anonym delegate.
* Kør testen, så den gennemføres uden fejl.

*Tidsramme: 30 minutter*

# Øvelse 3: Anonyme delegates og lambda

Kig nøje på de følgende tre anonyme delegates:

|  |  |
| --- | --- |
| 1: | delegate (int x, int y)  {  int result = x + y;  return result;  }; |
|  |  |
| 2: | delegate (int x)  {  int result = 0;  for (int i = 0; i < 10; i++)  {  result += x;  }  return result;  }; |
|  |  |
| 3: | delegate (int x)  {  return x \* x;  }; |

Udfør følgende:

* Konvertér disse 3 anonyme delegates til deres tilsvarende lambda-udtryk eller lambda-sætninger (tip: se afsnit 9.1 i Jon Skeet’s bog).

*Tidsramme: 30 minutter*

* Benyt ”30 til buffet” til at opsamle og reflektere over jeres fund.

*Tidsramme: 10 minutter*

# Øvelse 4: Anvendelse af Lambda i BonusApp

Næste trin er at anvende lambda-udtryk i BonusApp.

## Øvelse 4.1: Konvertér til Lambda

Udfør følgende

* Indsæt følgende testmetode i BonusApp-testprojektet:

[TestMethod]

public void GetBonusLambda\_Test()

{

order.Bonus = Bonuses.TenPercent; // <- Change to lambda expression

Assert.AreEqual(4.5, order.GetBonus());

order.Bonus = Bonuses.FlatTwoIfAmountMoreThanFive; // <- Change to lambda expression

Assert.AreEqual(2.0, order.GetBonus());

}

* Tilret koden i testen, så Bonuses.TenPercent og Bonuses.FlatTwoIfAmountMoreThanFive ændres til et tilsvarende lambda-udtryk.
* Kør testen, så den gennemføres uden fejl.

*Tidsramme: 10 minutter*

## Øvelse 4.2: Lambda-udtryk i GetBonus-parameter

I den udleverede kode beregner GetBonus()-metoden (i Order-klassen) den samlede bonus for en ordre ved at kalde Order-klassens delegate Bonus af typen BonusProvider.

I skal nu udvikle en alternativ måde at angive en bonusberegning helt uden brug af Order-klassens Bonus-property. I stedet skal I gøre det muligt at angive en bonusberegning via en *delegate-parameter* til GetBonus().

Udfør følgende:

* Afgør først hvilken delegate type denne parameter til GetBonus() skal være.
* Overload nu GetBonus() med metoden GetBonus(*<delegate-type> <parameter-navn>*), hvor det er muligt at angive bonusberegningen i parameteren.
* Implementér metodens krop, så den anvender parameteren til at udføre bonusberegningen i stedet for klassens Bonus-property.
* Indsæt testmetoden GetBonusByLambdaParameter\_Test() forneden til testprojektet:

[TestMethod]

public void GetBonusByLambdaParameter\_Test()

{

// Use TenPercent lambda expresssion as parameter to GetBonus

Assert.AreEqual(4.5, order.GetBonus(Bonuses.TenPercent));

// Use FlatTwoIfAmountMoreThanFive lambda expresssion as parameter to GetBonus

Assert.AreEqual(2.0, order.GetBonus(Bonuses.FlatTwoIfAmountMoreThanFive));

}

* Erstat henholdsvis Bonuses.TenPercent og Bonuses.FlatTwoIfAmountMoreThanFive i testmetoden med de tilsvarende lambda-udtryk, I har fundet frem til.
* Kør testen, og tilret om nødvendigt.

*Tidsramme: 20 minutter*

## Øvelse 4.3: Action, Func og Predicate i GetBonus()

I stedet for at angive delegate-typen BonusProvider kan man bruge en af de allerede indbyggede delegate-typer Action, Func eller Predicate.

* Overvej hvorvidt det er Action, Func eller Predicate, som kan anvendes i stedet for BonusProvider.
* Tilret GetBonus(*<delegate-type> <parameter-navn>*)-metoden, som I udviklede i forrige øvelse 4.2, så den anvender den valgte indbyggede delegate-type, som præcis erstatter BonusProvider.
* Kør testen for at sikre jer, at rettelsen fungerer korrekt.

*Tidsramme: 15 minutter*

## Øvelse 4.4: Action, Func og Predicate i GetTotalPrice()

Metoden GetTotalPrice() er den eneste anden metode i Order-klassen, som anvender GetBonus().

Udfør følgende:

* Overload GetTotalPrice() med GetTotalPrice(*<delegate-type> <parameter-navn>*), hvor man kan angive bonusberegningen som parameter. Parameterens type skal være den indbyggede delegate-type, I har valgt foroven i øvelse 4.3.
* GetTotalPrice(*<delegate-type> <parameter-navn>*) må ikke anvende klassens Bonus-property, så I skal implementere metodens krop, så den anvender GetBonus(*<delegate-type> <parameter-navn>*).
* Indsæt testmetoden GetTotalPriceByLambdaParameter\_Test() forneden til testprojektet:

[TestMethod]

public void GetTotalPriceByLambdaParameter\_Test()

{

Assert.AreEqual(40.5, order.GetTotalPrice(Bonuses.TenPercent));

Assert.AreEqual(43.0, order.GetTotalPrice(Bonuses.FlatTwoIfAmountMoreThanFive));

}

* Erstat henholdsvis TenPercent() og FlatTwoIfAmountMoreThanFive() med de tilsvarende lambda-udtryk, I har fundet frem til.
* Kør testen, og tilret om nødvendigt.

*Tidsramme: 15 minutter*