# Lambda

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf1**: anvende centrale metoder til at specificere og konstruere algoritmer […] * **1Pf2**: anvende centrale faciliteter i programmeringssproget til realisering af algoritmer, designmønstre, abstrakte datatyper, datastrukturer, designmodeller og brugergrænseflader. |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn ved **Delegates** * [Unistrukturel] Du kan genkende enkelte kendetegn ved **Lambda** |
| **Din forberedelse** | * Overview – Lambda (video: 17:19; se i planen i ItsLearning) * Gense: [Using anonymous delegates](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/using-anonymous-delegates?u=57075649) (video: 2:44) * [Understanding C# lambdas](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/understanding-c-sharp-lambdas) (video: 5:31) * [Using lambdas as delegates](https://www.linkedin.com/learning/c-sharp-delegates-events-and-lambdas/using-lambdas-as-delegates) (video 2:22) * C# - Anonymous Delegate to Lambda (video: 6:41; se i planen i ItsLearning) * C# - Lambda – Local Vars (video: 4:57; se i planen i ItsLearning) * C# - Action Func Predicate (video 12:05; se i planen i ItsLearning) * [Action, Func and Predicate - Predefined Generic Delegates](http://www.abajwa.com/post/Action-Func-and-Predicate-Predefined-delegates) * ”Jon Skeet - C# In Depth - 9.1 Lambda Expression.pdf” (5 sider, se i planen i ItsLearning) * ”Jon Skeet - C# In Depth – 2.1 Delegates.pdf” (6 sider, se i planen i ItsLearning) * Du kan med fordel evt. også genbesøge og repetere øvelse 2 i Ex33-Delegates opgaven (BonusApp-casen), da du skal fortsætte med at arbejde med udleveret kode fra denne case i nærværende opgave. |

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

Me: Give me 10 jokes on delegates and lambda

ChatGPT: Sure, here are 10 jokes related to delegates and lambdas:

* Why did the lambda function break up with the delegate? Because it was too controlling!
* What do you call a lamb with a computer science degree? A lamb-da function!
* What did the delegate say to the lambda function? "I promise to return your call-back."
* Why did the lambda function go to the therapist? Because it had too many arguments!
* Why did the delegate refuse to work with the lambda function? Because it was too anonymous!
* Why did the programmer name his lambdas after his friends? So he could always call them by name!
* What's the difference between a delegate and a lambda function? One is a pointer to a method, and the other is a method to a pointer!
* Why did the lambdas refuse to cross the road? Because they didn't want to invoke any unintended consequences!
* How many delegates does it take to change a light bulb? None, they just pass the function pointer to someone else.
* Why did the lambda function get lost? Because it couldn't find its way through the forest of nested functions!

# Øvelse 1: Terminologi

Del teamet op i mindre grupper og brug **Ordet rundt** til at reflektere over begreberne: ”lambda”, ”lambda expression”, ”lambda statement”, ”Action”, ”Func” og ”Predicate”.  
Sørg for at alle får mulighed for at tale.

*Tidsramme: ca. 20 minutter*

**Benyt parprogrammering til alle følgende øvelser.**

# Øvelse 2: Metodesignaturer og delegate-typer

Kig nøje på følgende 13 metodesignaturer:

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

Udfør følgende:

* Definér så få delegate-typer som muligt, der beskriver alle 13 metodesignaturer. Find selv på navnene på de delegate-typer, du 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

# Øvelse 3: Anonyme delegates i BonusApp

Udfør følgende:

* Hent zip-filen Ex48-BonusApp.Code.zip i GitLab-materialemappen; dette er en mulig løsning til øvelse 2 i den tidligere opgave Ex33-Delegates, som du nu skal arbejde videre med
* Pak koden i Ex48-BonusApp.Code.zip ud, og åbn løsningen i Visual Studio.
* Opret et nyt ’MSTest Test’ projekt og indsæt testen i ”Ex48-Lambda-UnitTest.txt” (også fra materialemappen på GitLab)
* Kig koden igennem, kompilér den, og kør testen. Der skulle ikke være nogen fejl.

## Øvelse 3.1: Konvertér til anonym metode

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

* Indsæt følgende testmetode:

[TestMethod]

public void GetBonusAnonymous\_Test()

{

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

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

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

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

}

* Tilret **testmetoden**, så Bonuses.TenPercent og Bonuses.FlatTwoIfAmountMoreThanFive (gult) ændres til en tilsvarende in-line anonym metode
* Kør testen, så den gennemføres uden fejl

# Øvelse 4: Anonyme metoder til lambda udtryk (en: expressions)

Kig nøje på de følgende tre anonyme metoder:

|  |  |
| --- | --- |
| 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 metoder til deres tilsvarende lambda-udtryk (en: expressions) eller lambda-sætninger (hint: se afsnit 9.1 i Jon Skeet’s bog)

# Øvelse 5: Anvendelse af Lambda i BonusApp

Næste trin er at anvende lambda-udtryk i BonusApp.

## Øvelse 5.1: 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 testmetoden i testen, så Bonuses.TenPercent og Bonuses.FlatTwoIfAmountMoreThanFive (gult) ændres til et tilsvarende lambda-udtryk
* Kør testen, så den gennemføres uden fejl

## Øvelse 5.2: 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.

Du skal nu udvikle en alternativ måde at angive en bonusberegning helt uden brug af Order-klassens Bonus-property, men hvor du stadig bruger delegate-type BonusProvider. I stedet skal du gøre det muligt at angive en bonusberegning via en *delegate-parameter* til GetBonus().

Udfør følgende:

* Overload 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 (gult) i testmetoden med de tilsvarende lambda-udtryk, I har fundet frem til
* Kør testen, og tilret om nødvendigt

## Øvelse 5.3: 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 5.2, så den anvender den valgte indbyggede delegate-type, som præcis erstatter BonusProvider
* Kør testen (fra øvelse 5.2) igen for at sikre dig, at rettelsen fungerer korrekt

## Øvelse 5.4: 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, du har valgt foroven i øvelse 5.3
* GetTotalPrice(*<delegate-type> <parameter-navn>*) må ikke anvende klassens Bonus-property, så du 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 Bonuses.TenPercent og Bonuses.FlatTwoIfAmountMoreThanFive (gult) i testmetoden med de tilsvarende lambda-udtryk, du har fundet frem til
* Kør testen, og tilret om nødvendigt