Övning Exceptions

Övning 1

Ladda ner övningsfiler.

Steg 1: Öppna webblässare, skriv in följande URL: easec.se/c/Exceptions.zip, välj att spara filen i katalogen c:\easec.

(OBS! Finns inte katalogen får du skapa den!)

Steg 3: Förflytta dig till katalogen C:\easec, högerklicka på Exceptions.zip och välj alternativet Extract All. Välj att packa upp till C:\easec\Modul 2. Klicka på Extract.

Övning 2

Observera hur Windows rapporterar undantag som inte är hanterade.

Steg 1: Öppna Visual Studio, om den inte redan är igång.

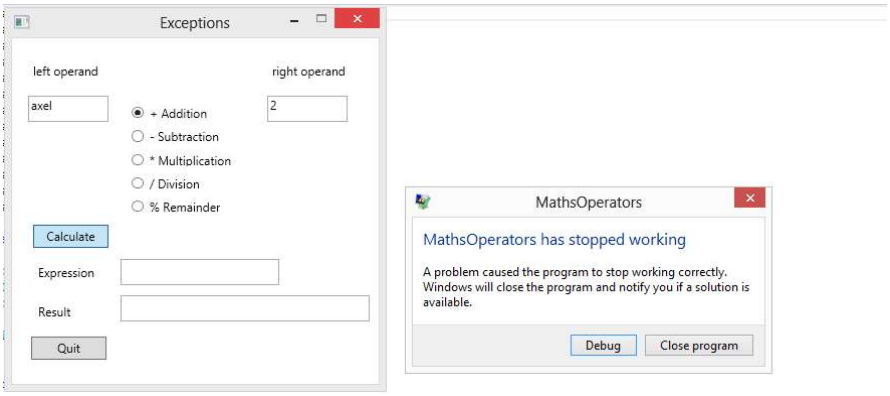
Steg 2: Öppna lösningen för MathsOperators (samma program som vi arbetade med i tidigare modul).

Steg 3: Klicka på alternativet Start Without Debugging, från DEBUG menyn.

OBS! Försäkra dig om att du kör applikationen utan Debugging.

Formuläret kommer att visas, vi kommer att skriva in text i fältet Left Operand, för att visa att programmet saknar funktioner för att hantera undantag.

Steg 4: I fältet för Left Operand, skriv in Axel, skriv in 2 i fältet för Right Operand, klicka först på + Addition, därefter på Calculate.



Värdena kommer att trigga felhanteringen i Windows. I Windows 8 kommer applikationen att avslutas och systemet kommer att lägga ut dialogfönstret MathsOperators. Klicka på alternativet Close program.

Om dialogrutan “Do you want to send information about the problem?”, klicka på Cancel.

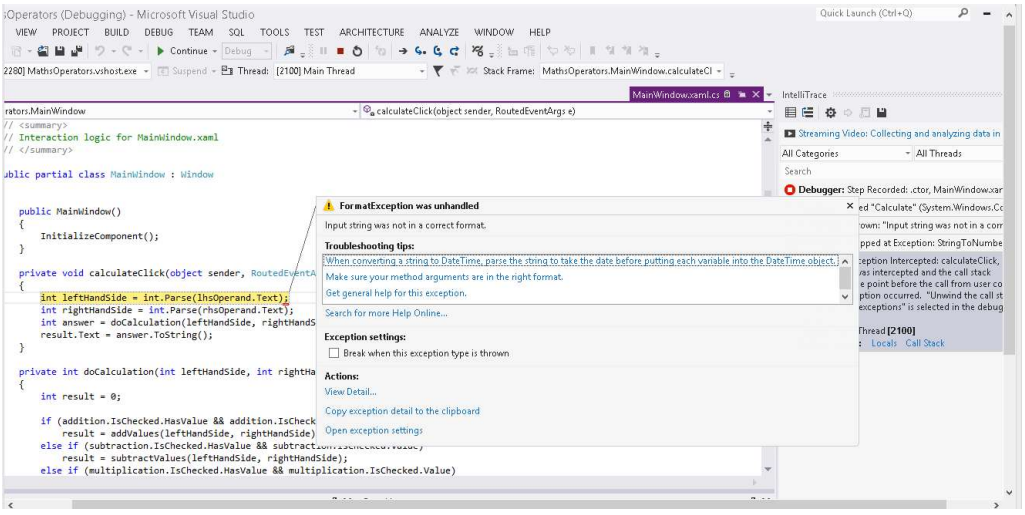
Nu när du har sett hur Windows spårar och hanterar undantag som inte är hanterade, är nästa steg att göra applikation mer robust genom att hantera invärde som inte är korrekta och förhindra denna typ av fel.

Övning 3

Steg 1: Återvänd till Visual Studio, klicka på alternativet Start Debugging, från DEBUG menyn.

Steg 2: När formuläret dyker upp, skriv in Axel i fältet för Left Operand, skriv in 2 i fältet för Right Operand, klicka först på + Addition, därefter på Calculate.

Invärdet kommer att orsaka samma undantag som tidigare, skillnaden är att du kör i läget debug. Det gör att Visual Studio kommer att spåra undantaget och rapportera detta.



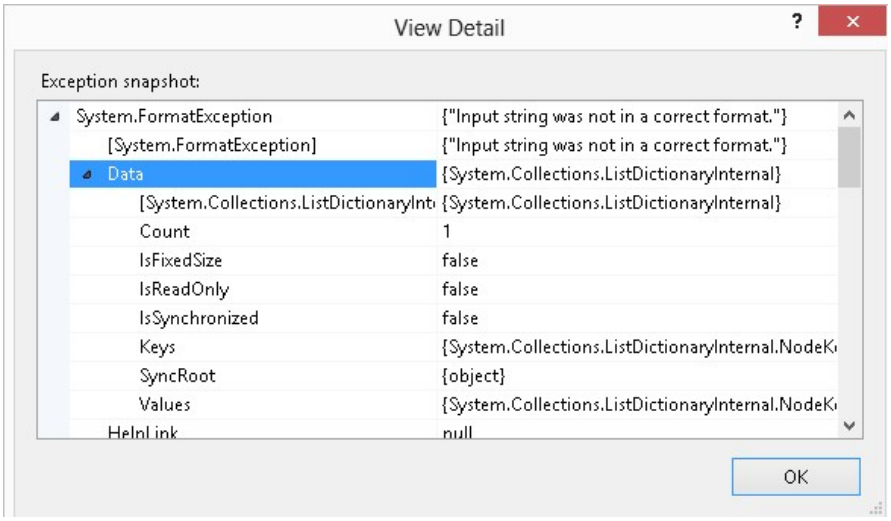
OBS! Om meddelanderuta dyker upp som talar om att break mode har fallerat pga fil App.g.i.c inte tillhör projekt som blir debuggad, klicka på OK och rutan kommer att försvinna och undantaget kommer att visas.

Steg 3: Visual Studio kommer att visa din kod och markera uttryck som orsakade undantaget tillsammans med dialogruta som visar undantaget. I detta fallet ”Input string was not in a correct format.”

Undantaget skickades av anropet till int.Parse i metoden addValues.

Problemet är att denna metod inte kan konvertera texten ”Axel” till ett giltigt nummer.

Steg 4: I dialogrutan för undantaget, klicka på alternativet View Detail.



Ytterligare dialogruta visas, som ger möjlighet att se mer information om detta undantag. Om du expanderar System.FormatException, kan du se mer information.

Steg 5: Klicka på OK i dialogrutan View Detail, klicka därefter på Stop Debugging i DEBUG meny.

Steg 6: Visa programkod för MainWindow.xaml.cs i fönstret Code and Text Editor och lokalisera metoden addValues.

Steg 7: Lägg till följande kod som visas med fet text:

```
try
{
int lhs = int.Parse(lhsOperand.Text);
int rhs = int.Parse(rhsOperand.Text);
```

```
int outcome = 0;

outcome = lhs + rhs;

expression.Text = lhsOperand.Text + " + " +
rhsOperand.Text

result.Text = outcome.ToString();

}

catch (FormatException fEx)

{

result.Text = fEx.Message;

}
```

Om ett undantag av typen FormatException uppstår, kommer catchhandler att visa text som finns i egenskapen Message i formulärets nederkant.

Steg 8: Klicka på alternativet Start Debugging, från DEBUG menyn.

Steg 9: När formuläret dyker upp, skriv in Axel i fältet för Left Operand, skriv in 2 i fältet för Right Operand, klicka först på + Addition, därefter på Calculate.

Catchhandler kommer nu att fånga undantaget och kommer visa meddelandet “Input string was not in a correct format”.

Steg 10: Ersätt Axel med siffran 10, skriv in Sharp i textrutan för Right Operand, klicka därefter på Calculate.

Samma sak kommer att hända som tidigare, vårt tillägg hanterar fel invärde i bägge rutorna.

Steg 11: Ersätt Sharp med siffran 20 i rutan för Right Operand, klicka sedan på + Addition, därefter på Calculate.

Applikationen fungerar på ett förväntat sätt och visar värdet 30 i resultatrutan.

Steg 12: I fältet för Left Operand, ersätta 10 med Axel och klicka därefter på – Substraction.

Visual Studio kommer att gå i debugläge och rapportera ett FormatException undantag. Denna gång kommer felet från metoden substractValues som inte innehåller nödvändig try/catch.

Steg 13: Klicka på Stop Debugging, från DEBUG meny.

Att lägga till try/catch metoden i addValues har gjort denna metod mer robust, men du behöver applicera samma hantering för metoderna: subtractValues, multiplyValues, divideValues och remainderValues. Programkod för varje hantering kommer att vara väldigt lika och kommer att resultera i att du skriver samma programkod i varje metod.

Varje metod kallar på metoden calculateClick när användare klicka på Calculate. För att undvika duplicering av programkod, kan du omlokalisera den till metoden calculateClick. Om ett undantag uppstår i metoderna subtractValues, multiplyValues, divideValues och remainderValues, kommer detta att propageras tillbaks till metoden calculateClick för hantering.

Övning 4

Steg 1: Visa programkod för MainWindow.xaml.cs i fönstret Code and Text Editor. Lokalisera metoden addValues.

Steg 2: Ta bort den programkod som vi la till tidigare. Koden skall se ut så här:

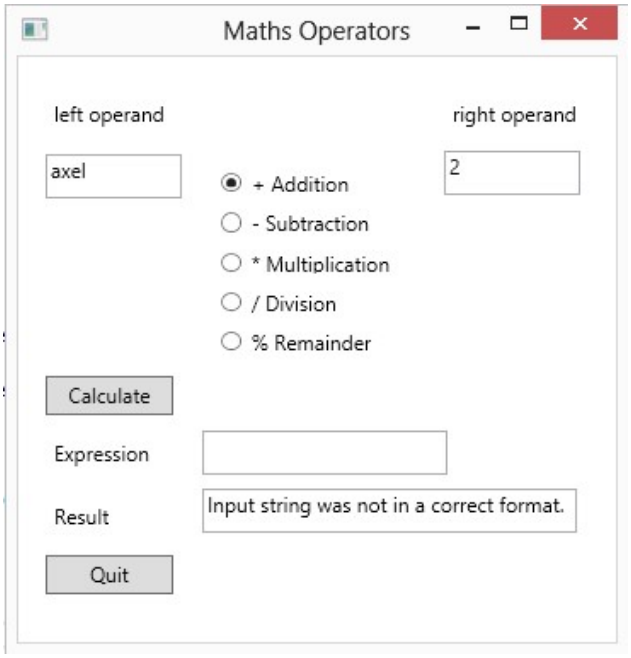
```
private void addValues()

{

int lhs = int.Parse(lhsOperand.Text);

int rhs = int.Parse(rhsOperand.Text);

int outcome = 0;

outcome = lhs + rhs;

expression.Text = lhsOperand.Text + " + " +
rhsOperand.Text

result.Text = outcome.ToString();

}
```

Steg 3: Leta upp metoden calculateClick. Lägg till följande programkod som visas med fet text:

```
private void calculateClick(object sender,
RoutedEventArgs e)
{
try
{
if ((bool)addition.IsChecked)
addValues();
else if ((bool)subtraction.IsChecked)
subtractValues();
else if ((bool)multiplication.IsChecked)
multiplyValues();
else if ((bool)division.IsChecked)
divideValues();
else if ((bool)remainder.IsChecked)
remainderValues();
}
catch (FormatException fEx)
{
result.Text = fEx.Message;
}
}
```

Steg 4: On the DEBUG menu, click Start Debugging.

Steg 5: När formuläret dyker upp, skriv in Axel i fältet för Left Operand, skriv in 2 i fältet för Right Operand, klicka först på + Addition, därefter på Calculate.



Catchhandler kommer nu att fånga undantaget och kommer visa meddelandet "Input string was not in a correct format".

Steg 6: Klicka på – Substraction, klicka därefter på Calculate. Samma sak som i tidigare steg kommer att hända.

Steg 7: Testa * Multiplication, / Division och % Remainder och verifiera att funktionen fungerar som det är tänkt.

Steg 8: Återvänd till Visual Studio och klicka på alternativet Stop Debugging.