

מבוא לתכנות II

הרצאה 1 - חריגות

סמסטר 1

- ישנן 3 סוגי שגיאות בתכנות:

1. טעות קומפילציה – כל מה שהקומפלייר מצליח למצוא.
2. שגיאת זמן ריצה – חריגה, תגרום לתוכנית לקרוס/לעוף.
3. טעות לוגית – שונה מהשניים הנ"ל. לעתים הקשה ביותר לתקן...

- אנו נדון בטעויות מסוג 2.

- ראו קטע קוד הבא:

```
int num1, num2;  
double result;  
Console.Write("insert num1: \t\t");  
num1 = int.Parse(Console.ReadLine());  
Console.WriteLine("will be devided by \t/  ");  
Console.Write("insert num2: \t\t");  
num2 = int.Parse(Console.ReadLine());  
result = num1 / num2;  
Console.WriteLine(result );
```

- אתם יכולים לחשוב על דוגמאות לקלט שיגרום לבעיות?

שגיאות זמן ריצה - חריגות

Unhandled Exception:
System.DivideByZeroException:
Attempted to divide by zero...

• נכניס את המספרים הבאים:

num1= 7 , num2=0 –

Unhandled Exception:
System.FormatException: Input
string was not in a correct format...

num1= 1.1 , num2=5 –

num1= ab7 , num2=5 –

num1= 7 , num2= +/- 12345678901 –

Unhandled Exception:
System.OverflowException: Value
was either too large or too small
for an Int32...

בלוק Try Catch

- בלוק try-catch מכיל בתוכו קוד "בעייתי" שעלול לגרום לחריגה בזמן הריצה.
- את הקוד הבעייתי רושמים בתוך הבלוק של try. כאשר שגיאת ריצה מתרחשת נוצרת חריגה והתוכנית קופצת מבלוק הtry אל בלוק הcatch.
- בלוק הcatch יתפוס את החריגה שנזרקה ובתוכו נוכל לרשום איזה קוד שנרצה על מנת להגיב לחריגה. למשל, הצגה למשתמש את השגיאה וסיבתה.
- בלוק הcatch ירוץ רק במידה ונזרקת חריגה.
- לאחר תום ביצוע הקוד בבלוק הcatch, התוכנית תמשיך להריץ את הקוד שרשום מתחת לבלוק.

Try- Catch בלוק

```
try
{
    //problematic code
}
catch (Exception)
{
    //there was an exception
}

//keep running...
```

Try- Catch בלוק

```
try
{
    int num1, num2;
    double result;
    Console.Write("insert num1: \t\t");
    num1 = int.Parse(Console.ReadLine());
    Console.WriteLine("will be devided by \t/  ");
    Console.Write("insert num2: \t\t");
    num2 = int.Parse(Console.ReadLine());
    result = num1 / num2;      //will generate exception when /0
    //result = (double)num1 / num2;
    Console.WriteLine(result);
}
catch (Exception)
{
    Console.WriteLine("something is wrong!!!");
}

Console.WriteLine("The END :) ");
```

מחלקת חריגה

- נוכל להיעזר במחלקת "חריגה" (Exception) על מנת לקבל פרטים על החריגה:

- Message
- Function
- Line
- File

```
...
catch (Exception Ex)
{
    Console.WriteLine("something is wrong!!!");
    Console.WriteLine(Ex.Message );
    Console.WriteLine( Ex.StackTrace.Substring(Ex.StackTrace.Length -
8, 8));
}
...
```


מחלקת חריגה ספציפית

- מחלקת חריגה היא הכללית ביותר וקיימות מחלקות ספציפיות יותר:

- DivideByZeroException
- NullPointerException
- FormatException
- OverflowException
- ...

- כשאנו משתמשים במספר בלוקי catch עבור חריגות שונות (ספציפיות) נוכל להגיב בהתאם לכל חריגה בנפרד.

- סדר כתיבת בלוקי catch **חשוב**: הבלוק הראשון שנמצא מתאים לחריגה שנזרקה ייבחר, ומהשאר **מתעלמים**.

- מהסיבה הנ"ל, נרשום תמיד את בלוק catch של החריגה הכללית (Exception) **אחרון**.

מחלקת חריגה

```
...
catch (FormatException ex)
{
    Console.WriteLine("the format of the numbers was incorrect - go
to school!");
}
catch (OverflowException ex)
{
    Console.WriteLine("the number is too big\\small - go to
school!");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
...
```

לתפוס את החריגה

- בכל פעם שחריגה מתרחשת, התוכנית מנסה למצוא את ה catch הקרוב ביותר לחריגה. ב"קרוב ביותר" הכוונה לקריאת הפונקציה הקרובה ביותר.
- לדוגמה: אם בתוך פונקציה א' יש קריאה לפונקציה ב', ובתוך פונקציה ב' יש בלוק try-catch שתופס חריגה כלשהי – אזי הקוד ימשיך לרוץ בפונקציה ב' לאחר בלוק ה catch.
- אבל, במידה ובפונקציה ב' אין בלוק catch שמתאים לחריגה, התוכנית תחפש אחריו בפונקציה א' וכך הלאה עד שמגיעים ל Main. אם גם שם אין catch הולם, תיזרק השגיאה למסך והתוכנית תופסק.
- התוכנית לא תחזור למקום המקורי בה נוצרה החריגה.

לתפוס את החריגה

```
try
{
    int num1, num2;
    double result;
    num1 = ReadNum1();
    Console.WriteLine("will be devided by \t/  ");
    Console.Write("insert num2: \t\t");
    num2 = int.Parse(Console.ReadLine());
    result = num1 / num2;      //will generate exception when /0
    //result = (double)num1 / num2;
    Console.WriteLine(result);
}
catch (FormatException ex)
{
    Console.WriteLine("the format of the num2 was incorrect - go to school!");
}
catch (OverflowException ex)
{
    Console.WriteLine("the number is too big\\small - go to school!");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

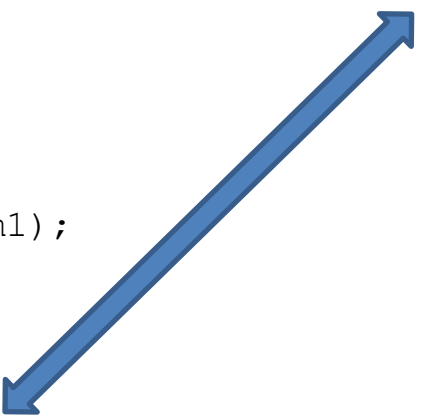
לתפוס את החריגה

```
static int ReadNum1()  
{  
    Console.Write("insert num1: \t\t");  
  
    //return int.Parse(Console.ReadLine()); //undo this remark to understand  
why it  
    //is important to write the tryCatch in the most inner function! - this  
    // will catch the num1 error but will show the message for incorrect num2!!  
  
    try  
    {  
        return int.Parse(Console.ReadLine() );  
    }  
    catch (FormatException ex)  
    {  
        Console.WriteLine("num1 - wrong format");  
        return 0;  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine(ex.Message);  
        return 0;  
    }  
    Console.WriteLine("ReadNum1 END"); //will never show  
}
```

יצירת חריגה משלנו

- אנחנו יכולים ליצור חריגה בעצמנו ולזרוק אותה ע"י קוד.

```
try
{
    int num1 = 7;
    Console.WriteLine("insert a number in words (one\\zero): ");
    string num2 = Console.ReadLine();
    if (num2 == "zero")
    {
        DivideByZeroException dvdExcp =
            new DivideByZeroException("u should not divide by zero and respect
your teacher!");
        throw dvdExcp;
    }
    else
    {
        Console.WriteLine(num1);
    }
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```



Finally

- בלוק הfinally מגיע מיד לאחר בלוק הcatch **ותמיד** יבוצע!
 - כאשר אין חריגה כלל.
 - כאשר אין בלוק catch שמתאים לחריגה.
 - כשחריגה נוספת נזרקת בתוך בלוק הcatch.
- במקרה ואין בלוק catch מתאים לחריגה, היא תוצג על המסך לפני הרצת הקוד בfinally.
- אנו משתמשים בfinally, לדוגמה, בסגירת קבצים וחיבורים לשרתים ומסדי נתונים...

Finally

```
private static void Finally()
{
    try
    {
        int num = 7 / int.Parse(Console.ReadLine());

    }
    catch (DivideByZeroException e)
    {
        Console.WriteLine(e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        int a = int.Parse("a");
    }
    finally
    {
        Console.WriteLine("this will be shown even if there is no mathing catch
block\n"+
            "or an exception occure within the catch block!");
    }
    Console.WriteLine("END");
}
```

```
asd
Input string was not in a correct format.

Unhandled Exception: System.FormatException: Input string was
  at System.Number.StringToNumber(String str, NumberStyles o
  at System.Number.ParseInt32(String s, NumberStyles style,
  at System.Int32.Parse(String s)
  at lecture12.Program.Finally() in C:\îĖĖĖĖĖĖĖ\ĖĖĖĖĖĖĖ\
n\solutions\lecture12\Program.cs:line 54
  at lecture12.Program.Main(String[] args) in C:\îĖĖĖĖĖĖĖ\ĖĖĖ
2 Exception\solutions\lecture12\Program.cs:line 24
this will be shown even if there is no mathing catch block
or an exception occure within the catch block!
Press any key to continue . . . _
```


Finally

```
private static void Finally2()  
{  
    try  
    {  
        Finally();  
    }  
    catch (Exception)  
    {  
        Console.WriteLine("in Finally2()");  
    }  
}
```

```
asd  
Input string was not in a correct format.  
this will be shown even if there is no mathing catch block  
or an exception occure within the catch block!  
in Finally2()  
Press any key to continue . . . _
```

תרגיל 1

- כתבו פונקציה בשם `int ReadNum()` אשר תמשיך לבקש מספר שלם מהמשתמש עד שהוא יספק אחד, כמובן ללא קריסה.

פונקציית String.split()

- פונקציית הפיצול מקבלת כארגומנט מערך תווים ומפצלת את המחרוזת למערך מחרוזות לפי התווים המבוקשים.
2 דרכים להשתמש בה:

```
char[] delim = " ".ToCharArray();  
strArr = myString.Split(delim);
```

```
strArr = myString.Split(new char[] { ' ' } );
```

תרגיל 2

- כתבו תוכנית אשר מקבלת רשימה של מספרים שלמים, מופרדים ברווחים, ואשר יכולים להיות גם טווח. על המסך יוצג ממוצע המספרים. לדוגמה:

```
Please insert your numbers space seperated  
and you can use the '-' to input a range:  
1 2 3 4  
The Average is: 2.50
```

```
Please insert your numbers space seperated  
and you can use the '-' to input a range:  
7-10  
The Average is: 8.50
```

```
Please insert your numbers space seperated  
and you can use the '-' to input a range:  
1 2 3 4-9 100-200  
The Average is: 138.14
```

זכרו למנוע קריסות!!

תרגיל 2

