C# : les exceptions

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Introduction

Une exception, c'est quoi ?

- C'est une erreur qui se produit pendant l'exécution de notre programme
- Une exception dans un programme implique généralement son arrêt d'exécution

Introduction

Comment faire pour poursuivre l'exécution ?

- Repérer les blocs pouvant générer une exception
- Capturer l'exception correspondante
- Afficher un message relatif à cette exception
- Continuer l'exécution

Introduction

`Exception` : **exemple**

```
class Program {
  static void Main(string[] args)
  {
    int x = 0;
    int y = 5 / x;
    Console.WriteLine(x);
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Introduction

`Exception` **: exemple**

```
class Program {
  static void Main(string[] args)
  {
    int x = 0;
    int y = 5 / x;
    Console.WriteLine(x);
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception non gérée : System.DivideByZeroException : Tentative de division par zéro.
...Program.cs :ligne 40

Introduction

`Exception` : **exemple**

```
class Program {
  static void Main(string[] args)
  {
    int x = 0;
    int y = 5 / x;
    Console.WriteLine(x);
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception non gérée : System.DivideByZeroException : Tentative de division par zéro.
...Program.cs :ligne 40

Constatation

- Le message `Fin de calcul` n'a pas été affiché
- La division par zéro déclenche une exception `DivideByZeroException`

Capture d'exception

Comment faire pour capturer une exception ?

- Utiliser un bloc `try { ... } catch { ... }`
- Le `try { ... }` pour entourer une instruction susceptible de déclencher une exception
- Le `catch { ... }` pour capturer l'exception et afficher un message qui lui correspond

Capture d'exception

`Exception` : **exemple**

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine("Exception : Division par zero ");
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Capture d'exception

`Exception` **: exemple**

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine("Exception : Division par zero ");
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception : Division par zéro
Fin de calcul

Capture d'exception

`Exception` **: exemple**

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine("Exception : Division par zero ");
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché à l'exécution

Exception : Division par zéro
Fin de calcul

Constatation

- L'exception a été capturée
- Le message `Fin de calcul` a été affiché

Capture d'exception

Et si je ne connais pas le type d'exception

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (Exception e) {
      Console.WriteLine("Exception : Division par zero ");
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Capture d'exception

Et si je ne connais pas le type d'exception

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (Exception e) {
      Console.WriteLine("Exception : Division par zero ");
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Le même message sera affiché

Exception : Division par zéro
Fin de calcul

Capture d'exception

Et si je ne connais pas le type d'exception

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (Exception e) {
      Console.WriteLine("Exception : Division par zero ");
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Le même message sera affiché

Exception : Division par zéro
Fin de calcul

Constatation

- La classe `Exception` peut être utilisée

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine("Exception : " + e.Message);
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine("Exception : " + e.Message);
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché

Exception : Tentative de division par zéro.
Fin de calcul

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine(e.StackTrace);;
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Capture d'exception

Utiliser des méthodes de la classe `Exception`

```
class Program {
  static void Main(string[] args)
    int x = 5, y = 0;
    try {
      Console.WriteLine(x/y);
    }
    catch (DivideByZeroException e) {
      Console.WriteLine(e.StackTrace);;
    }
    Console.WriteLine("Fin de calcul");
  }
}
```

Le message affiché est :

à MyProject.Program.Main(String[] args) dans
C :/.../source/repos/MySolution/MyProject/Program.cs :ligne 43
Fin de calcul

Les exceptions personnalisées

On a utilisé (ou vu) des exceptions prédéfinies

- `Exception`
- `DivideByZeroException`
- `IndexOutOfRangeException`

Les exceptions personnalisées

On a utilisé (ou vu) des exceptions prédéfinies

- `Exception`
- `DivideByZeroException`
- `IndexOutOfRangeException`

On peut aussi définir nos exceptions personnalisées

Les exceptions personnalisées

La classe `Adresse`

```
public class Adresse {

  public string Rue { get; set; }
  public string CodePostal { get; set; }
  public string Ville { get; set; }

  public Adresse(string rue, string ville, string
     codePostal) {
    Rue = rue;
    Ville = ville;
    CodePostal = codePostal;
  }
}
```

Les exceptions personnalisées

Supposons que

- `codePostal` doit contenir exactement 5 chiffres

Les exceptions personnalisées

Supposons que

- `codePostal` doit contenir exactement 5 chiffres

Démarche à faire

- Créer notre propre exception (qui doit étendre la classe `Exception`)
- Dans le constructeur de `Adresse`, on lance une exception si `codePostal` ne contient pas 5 chiffres

Les exceptions personnalisées

Créons l'exception `IncorrectCodePostalException`

```
public class IncorrectCodePostalException :
   Exception
{
  // le constructeur de cette nouvelle exception
  public IncorrectCodePostalException() :
            base("Le code postal doit contenir
               exactement 5 chiffres")
  {
  }
}
```

Les exceptions personnalisées

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {
  public string Rue { get; set; }
  public string CodePostal { get; set; }
  public string Ville { get; set; }

  public Adresse(string rue, string ville, string
     codePostal) {
    if (codePostal.Length != 5)
      throw new IncorrectCodePostalException();
    Rue = rue;
    Ville = ville;
    CodePostal = codePostal;
  }
}
// il faut faire pareil pour le setter du codePostal
```

Les exceptions personnalisées

Testons tout cela dans le `Main()`

```
static void Main(String[] args) {
  Adresse a = null;
  try {
    a = new Adresse ("rue de paradis", "Marseille",
       "1300");
  }
  catch(IncorrectCodePostalException icpe) {
    Console.WriteLine(icpe.Message);
  }
}
```

Les exceptions personnalisées

Testons tout cela dans le `Main()`

```
static void Main(String[] args) {
  Adresse a = null;
  try {
    a = new Adresse ("rue de paradis", "Marseille",
       "1300");
  }
  catch(IncorrectCodePostalException icpe) {
    Console.WriteLine(icpe.Message);
  }
}
```

Le message affiché est :

Le code postal doit contenir exactement 5 chiffres

Les instructions multi-catch

On peut rajouter une deuxième condition

- `codePostal` doit contenir exactement 5 chiffres
- `rue` doit être une chaîne en majuscule

Les instructions multi-catch

Créons une deuxième exception

`IncorrectStreetNameException`

```
public class IncorrectStreetNameException :
   Exception
{
  public IncorrectStreetNameException(): base("Le
     nom de la rue doit être en majuscule")
  {
  }
}
```

Les instructions multi-catch

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {

  public string Rue { get; set; }
  public string CodePostal { get; set; }
  public string Ville { get; set; }

  public Adresse(string rue, string ville, string codePostal) {
    if (codePostal.Length != 5)
      throw new IncorrectCodePostalException();
    if (!rue.Equals(rue.ToUpper()))
      throw new IncorrectStreetNameException();
    Rue = rue;
    Ville = ville;
    CodePostal = codePostal;
  }
}
```

Les instructions multi-catch

Re-testons tout cela dans le `Main()`

```
static void Main(String[] args)
{
  try
  {
    Adresse a = new Adresse ("paradis", "Marseille", "
       13000");
  }
  catch(IncorrectCodePostalException icpe)
  {
    Console.WriteLine(icpe.Message);
  }
  catch(IncorrectStreetNameException isne)
  {
    Console.WriteLine(isne.Message);
  }
}
```

Les instructions multi-catch

On peut aussi fusionner les `catch`

```
static void Main(String[] args)
{
  try
  {
    Adresse a = new Adresse ("paradis", "Marseille",
        "1300");
  }
  catch (Exception e) when
    (e is IncorrectCodePostalException ||
     e is IncorrectStreetNameException)
  {
    Console.WriteLine(e.Message);
  }
}
```

Les exceptions paramétrées

Hypothèse

- Si on voudrait afficher les valeurs qui ont déclenché l'exception dans le message

Les exceptions paramétrées

Modifions la première exception

`IncorrectCodePostalException`

```
public class IncorrectCodePostalException :
   Exception
{
  // le constructeur de cette nouvelle exception
  public IncorrectCodePostalException(string cp) :
            base($"Le code postal {cp} doit contenir
                exactement 5 chiffres")
  {
  }
}
```

Les exceptions paramétrées

Modifions la deuxième exception

`IncorrectStreetNameException`

```
public class IncorrectStreetNameException :
   Exception {
  public IncorrectStreetNameException(String rue) :
    base ("Le nom de la rue '" + rue + "' doit être
       en majuscule");
  }
}
```

Les exceptions paramétrées

Modifions le constructeur de la classe `Adresse`

```
public class Adresse {

  public string Rue { get; set; }
  public string CodePostal { get; set; }
  public string Ville { get; set; }

  public Adresse(string rue, string ville, string codePostal) {
    if (codePostal.Length != 5)
      throw new IncorrectCodePostalException(codePostal);
    if (!rue.Equals(rue.ToUpper()))
      throw new IncorrectStreetNameException(rue);
    Rue = rue;
    Ville = ville;
    CodePostal = codePostal;
  }
}
```

Les exceptions paramétrées

Pour tester

```
static void Main(String[] args)
{
  try
  {
    Adresse a = new Adresse ("paradis", "Marseille", "1300");
  }
  catch (Exception e) when
    (e is IncorrectCodePostalException ||
     e is IncorrectStreetNameException)
  {
    Console.WriteLine(e.Message);
  }
}
```

Les exceptions paramétrées

Pour tester

```
static void Main(String[] args)
{
  try
  {
    Adresse a = new Adresse ("paradis", "Marseille", "1300");
  }
  catch (Exception e) when
    (e is IncorrectCodePostalException ||
     e is IncorrectStreetNameException)
  {
    Console.WriteLine(e.Message);
  }
}
```

Le message affiché est :

Le code postal ’1300’ doit contenir exactement 5 chiffres

Les exceptions paramétrées

Exercice

- Créer une nouvelle classe d'exception `AdresseException` pour fusionner et remplacer les deux exceptions `IncorrectCodePostalException` et `IncorrectStreetNameException`

Le bloc `finally`

- À utiliser quand on veut exécuter une instruction qu'une exception soit levée ou non

Le bloc finally

Exemple

```
public class Program {
    static void Main(String[] args) {
     int x = 5, y = 0;
     try
     {
       Console.WriteLine(x/y);
     }
     catch (Exception e)
     {
       Console.WriteLine("Division par zero");
     }
     finally
     {
       Console.WriteLine("Instruction exécutée systématiquement"
          );
     }
   }
}
```

Le bloc `finally`

Remarque

- Le bloc `finally` peut s'avérer intéressant si le `catch` contiens un `return` qui forcera l'arrêt de l'exécution du code. Malgré cela, ce bloc (`finally`) sera exécuté.