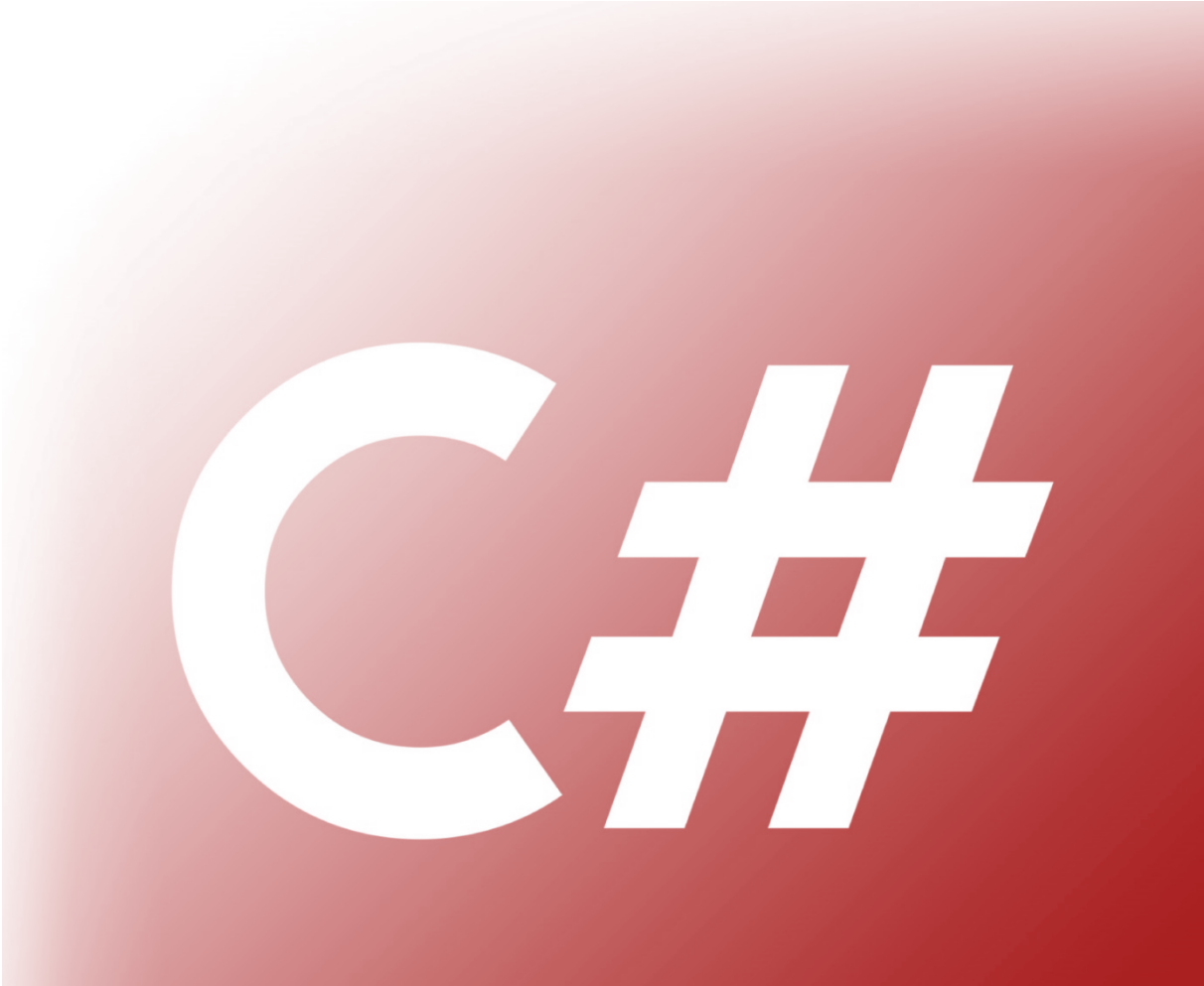
TP Chapitre 5

Concepts avancés



❖ Gestion des exceptions

- ❖ Écrire un programme dans lequel on demande à l'utilisateur de saisir un entier en gérant l'exception dans le cas où il ne saisit pas un entier correctement.
- ❖ Écrire un programme dans lequel on fait la division de 2 entiers saisis par l'utilisateur en gérant l'exception dans le cas d'une division par zéro.
- ❖ Compléter le programme suivant pour que les erreurs susceptibles de se produire soient gérées par des exceptions.

```
static void Main(string[] args)
{
    List<int> liste = new List<int>() { 17, 12, 15, 38, 29, 157, 89, -22, 0, 5 };

    Console.WriteLine("Enter l'indice de l'entier à diviser :");
    int indice = int.Parse(Console.ReadLine());

    Console.WriteLine("Entrer le diviseur :");
    int diviseur = int.Parse(Console.ReadLine());

    Console.WriteLine("Le résultat de la division est : " + liste[indice] / diviseur);

    Console.Read();
}
```

❖ Utilisation des génériques

- ❖ Ecrire un programme qui gère la présence au sein d'une classe en utilisant un dictionnaire <string, bool>.
  - ∞ L'utilisateur entre le nom de l'élève, suivi d'une virgule, suivi de la lettre « t » pour « true » (signifiant sa présence), ou la lettre « f » pour « false » (indiquant son absence).

    ```
    Luke,t
    ```
  - ∞ Le programme remplie le dictionnaire par le nom de l'élève et sa présence (true ou false).
  - ∞ L'utilisateur continue à saisir tant que qu'il ne saisit pas la valeur « x ».
  - ∞ Le programme affiche ainsi la somme des élèves présents.
  - ∞ Le programme affiche aussi la liste des élèves absents.

❖ Linq To XML

- ❖ Créer un programme qui permet de manipuler un fichier XML avec « Linq To XML ».
  - ∞ Ajouter un nouveau fichier XML « Employees.xml » au projet.
  - ∞ Y ajouter le contenu suivant :

```xml
<Employees>
   <Employee>
      <EmpId>1</EmpId>
      <Name>Sam</Name>
      <Sex>Male</Sex>
      <Phone Type="Home">423-555-0124</Phone>
      <Phone Type="Work">424-555-0545</Phone>
      <Address>
         <Street>7A Cox Street</Street>
         <City>Acampo</City>
         <State>CA</State>
         <Zip>95220</Zip>
         <Country>USA</Country>
      </Address>
   </Employee>
   <Employee>
      <EmpId>2</EmpId>
      <Name>Lucy</Name>
      <Sex>Female</Sex>
      <Phone Type="Home">143-555-0763</Phone>
      <Phone Type="Work">434-555-0567</Phone>
      <Address>
         <Street>Jess Bay</Street>
         <City>Alta</City>
         <State>CA</State>
         <Zip>95701</Zip>
         <Country>USA</Country>
      </Address>
   </Employee>
   <Employee>
      <EmpId>3</EmpId>
      <Name>Kate</Name>
      <Sex>Female</Sex>
      <Phone Type="Home">166-555-0231</Phone>
      <Phone Type="Work">233-555-0442</Phone>
      <Address>
         <Street>23 Boxen Street</Street>
         <City>Milford</City>
         <State>CA</State>
         <Zip>96121</Zip>
         <Country>USA</Country>
      </Address>
   </Employee>
   <Employee>
      <EmpId>4</EmpId>
      <Name>Chris</Name>
      <Sex>Male</Sex>
      <Phone Type="Home">564-555-0122</Phone>
      <Phone Type="Work">442-555-0154</Phone>
      <Address>
         <Street>124 Kutbay</Street>
         <City>Alta</City>
         <State>CA</State>
         <Zip>94037</Zip>
         <Country>USA</Country>
      </Address>
   </Employee>
</Employees>
```

- Ecrire les requêtes « Linq To XML » suivantes :
  1. Charger le document XML.
  2. Afficher tous éléments du document.
  3. Afficher tous les éléments « Name » et « EmpId ».
  4. Afficher tous les employés dont le sexe est « Female ».
  5. Afficher tous les employés qui habite à la ville « Alta ».
  6. Afficher tous les numéros de téléphone de type « Home ».
  7. Afficher tous les codes zip du document.
  8. Afficher et trier en ordre ascendant tous les codes zip.
  9. Afficher l'employer à la 2ème position.
  10. Afficher les 2 premiers employés.

❖ Linq To Object

- ❖ Créer un programme qui permet de manipuler une liste des objets avec « Linq To Object ».
  - ∞ Ajouter les 2 classes « Employee » et « Address » qui ont les mêmes attributs que le fichier XML « Employees.xml ».
  - ∞ Créer une nouvelle liste des employés comme suit :

```
List<Employee> listEmployees = new List<Employee>()
{
    new Employee() { EmpId = 1, Name = "Sam", Sex = "Male", Phone = "423-555-0124", Address =
        new Address() { Street = "7A Cox Street", City = "Acampo", State = "CA", Zip = "95220", Country = "USA" }},
    new Employee() { EmpId = 2, Name = "Lucy", Sex = "Female", Phone = "143-555-0763", Address =
        new Address() { Street = "Jess Bay", City = "Alta", State = "CA", Zip = "95701", Country = "USA" }},
    new Employee() { EmpId = 3, Name = "Kate", Sex = "Female", Phone = "166-555-0231", Address =
        new Address() { Street = "23 Boxen Street", City = "Milford", State = "CA", Zip = "96121", Country = "USA" }},
    new Employee() { EmpId = 4, Name = "Chris", Sex = "Male", Phone = "564-555-0122", Address =
        new Address() { Street = "124 Kutbay", City = "Montara", State = "CA", Zip = "94037", Country = "USA" }},
};
```

  - ∞ Ecrire les requêtes « Linq To Objects » suivantes :
    1. Afficher la liste de tous les employés dont le sexe est Male et habitent à USA.
    2. Afficher la liste de tous les employés dont le numéro de téléphone commence par 1 ordonnés en descendant par le numéro de téléphone et après en ascendant par la ville.
    3. Afficher l'employé à la position 1 de la liste de tous les employés dont le numéro de téléphone commence par 1 ordonnés en ascendant.
    4. Afficher la liste des noms de tous les employés.
    5. Afficher l'employé à la 3ème position.
    6. Afficher l'employé dont l'identifiant est 2.
    7. Afficher si au moins un employé habite à la ville d'Alta.
    8. Afficher la longueur du plus grand nom de voie.
    9. Faire la multiplication par 10 des identifiants de tous les employés.
    10. Afficher la somme des identifiants de tous les employés dont le sexe est "Male".

❖ Delegates d'événements

- ❖ Nous Allons créer un mini simulateur de tir à l'arc, qui va être utilisé par un joueur afin d'en faire des statistiques.
  - ∞ Nous devons créer un simulateur de tir à l'arc. Lorsque celui-ci est démarré, il génère autant de nombre aléatoire que demandé, des nombres entre 0 et 100, qui indiquent la valeur du tir dans la cible.
  - ∞ Si le nombre aléatoire est inférieur à 30, alors cela veut dire que le tir est mauvais. S'il est supérieur ou égal à 30 et inférieur à 60, alors nous aurons un tir passable. S'il est supérieur ou égal à 60 et inférieur à 90, alors c'est bien. Sinon, nous aurons un tir excellent.
  - ∞ Un événement sera levé à chaque changement de cible. Le but de notre joueur est de s'abonner aux événements du simulateur afin de compter le nombre de fois où le tir est excellent, et le nombre de fois où le tir a changé. Il affichera ensuite son rapport en indiquant ces deux résultats et le pourcentage de fois où le tir est excellent.