Laboratoire nr. 1:

Charge:

Créer votre propre console application en C# qui doit avoir du 2 a 5 classes.

Le suivant Menu des options doit être disponible dans votre application :

* La possibilité d’ajouter un nouvel enregistrement dans une classe
* La possibilité d’ajouter nouvelles enregistrements dans une classe en donner la position d’après laquelle les enregistrements doit être introduises. Implicite, la valeur de numéro d’enregistrement qui doit être ajouté dans une classe
* La possibilité d’effacer un seul enregistrement de l’une classe choisi par utilisateur
* La possibilité d’effacer plusieurs enregistrements de l’une classe choisi par utilisateur en ayant la position initial e la position final des enregistrements qui doit être effacé.
* Le triage des enregistrements par le nom, l’année de la naissance, etc.
* Override la méthode ToString() de la classe.

NB ! Pour les personnes qui veulent avoir 8+, on doit avoir en plus les suivant options:

* Une fonction générique qui doit effacer un enregistrement ou plusieurs enregistrements de la liste ou le tableau utilisé.
* Le triage générique pour des enregistrements. Si le triage pour tout la structure ne peut pas être effectué, vous pouvez utiliser le triage d’un champ de la structure, et après cela de trier votre liste.

Ou... vous pouvez inventez de ‘s autres méthodes s’il existe... par exemple : vous pouvez utiliser la méthode ToString()...

Source Code:

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace MIDPS\_Lab1

{

class Program

{

static void Main(string[] args)

{

Program instance = new Program();

WorldHandler handler = new WorldHandler();

while (true)

{

instance.showMenu();

char x = Console.ReadKey().KeyChar;

Console.WriteLine();

switch (x)

{

case '0':

handler.AddNewOject(instance.addThing());

break;

case '1':

Console.WriteLine("At which index do you want to insert the element?");

int insertIndex = Int32.Parse(Console.ReadLine());

Console.WriteLine("How much elements do you want to add?");

int varCount = Int32.Parse(Console.ReadLine());

for (int i = 0; i < varCount; i++)

{

handler.AddNewOjectAtIndex(instance.addThing(), insertIndex+i);

}

break;

case '2':

Console.WriteLine("At which index do you want to delete the element?");

int removeIndex = Int32.Parse(Console.ReadLine());

handler.destroyObject(instance.deleteItem(), removeIndex);

break;

case '3':

Console.WriteLine("Write the start index from where to delete.");

int startRemove = Int32.Parse(Console.ReadLine());

Console.WriteLine("Write the end index until where to delete.");

int endRemove = Int32.Parse(Console.ReadLine());

handler.destroyMultipleObjects(instance.deleteItem(), startRemove, endRemove);

break;

case '4':

instance.sortItems(handler);

break;

case '5':

instance.showItems(handler);

break;

default:

return;

}

//Console.Clear();

}

}

void showMenu()

{

Console.WriteLine(" Menu:");

Console.WriteLine(" 0) Add element");

Console.WriteLine(" 1) Add elements at index");

Console.WriteLine(" 2) Remove element");

Console.WriteLine(" 3) Remove multiple elements");

Console.WriteLine(" 4) Sort elements");

Console.WriteLine(" 5) Display elements");

Console.WriteLine(" 6) Exit program");

}

object addThing()

{

Console.WriteLine(" What object do you want to create ?");

Console.WriteLine(" 0) Add a ring");

Console.WriteLine(" 1) Register a wizard");

Console.WriteLine(" 2) Register an elf");

Console.WriteLine(" 3) Create an orc");

Console.WriteLine(" 4) Visit a hobbit");

char x = Console.ReadKey().KeyChar;

Console.WriteLine();

switch (x)

{

case '0':

{

Ring temp = new Ring();

Console.WriteLine("Ring material?");

temp.Material = Console.ReadLine();

Console.WriteLine("Ring number?");

temp.id = UInt32.Parse(Console.ReadLine());

return temp;

}

case '1':

{

Wizard temp = new Wizard();

Console.WriteLine("Wizard name?");

temp.Name = Console.ReadLine();

Console.WriteLine("Wizard color?");

temp.Color = Console.ReadLine();

Console.WriteLine("Wizard id?");

temp.id = UInt32.Parse(Console.ReadLine());

return temp;

}

case '2':

{

Elf temp = new Elf();

Console.WriteLine("Elf name?");

temp.Name = Console.ReadLine();

Console.WriteLine("Elf category?");

temp.Category = Console.ReadLine();

Console.WriteLine("Elf id?");

temp.id = UInt32.Parse(Console.ReadLine());

return temp;

}

case '3':

{

Orc temp = new Orc();

Console.WriteLine("Orc master?");

temp.Master = Console.ReadLine();

Console.WriteLine("Orc id?");

temp.id = UInt32.Parse(Console.ReadLine());

Console.WriteLine("Orc power?");

temp.Power = float.Parse(Console.ReadLine());

return temp;

}

case '4':

{

Hobbit temp = new Hobbit();

Console.WriteLine("Hobbit name?");

temp.Name = Console.ReadLine();

Console.WriteLine("Hobbit family?");

temp.Family = Console.ReadLine();

Console.WriteLine("Hobbit id?");

temp.id = UInt32.Parse(Console.ReadLine());

return temp;

}

default:

return null;

}

}

Type deleteItem()

{

Console.WriteLine(" What do you want to destroy today?");

Console.WriteLine(" 0) A ring");

Console.WriteLine(" 1) A wizard");

Console.WriteLine(" 2) An elf");

Console.WriteLine(" 3) An orc");

Console.WriteLine(" 4) A hobbit");

char x = Console.ReadKey().KeyChar;

Console.WriteLine();

switch (x)

{

case '0': return typeof(Ring);

case '1': return typeof(Wizard);

case '2': return typeof(Elf);

case '3': return typeof(Orc);

case '4': return typeof(Hobbit);

default: return null;

}

}

void sortItems( WorldHandler handler)

{

Console.WriteLine(" What do you want to sort today?");

Console.WriteLine(" 0) Rings");

Console.WriteLine(" 1) Wizards");

Console.WriteLine(" 2) Elfs");

Console.WriteLine(" 3) Orcs");

Console.WriteLine(" 4) Hobbits");

char x = Console.ReadKey().KeyChar;

Console.WriteLine();

switch (x)

{

case '0': handler.SortGeneric<Ring>(); break;

case '1': handler.SortGeneric<Wizard>(); break;

case '2': handler.SortGeneric<Elf>(); break;

case '3': handler.SortGeneric<Orc>(); break;

case '4': handler.SortGeneric<Hobbit>(); break;

}

}

void showItems(WorldHandler handler)

{

Console.WriteLine(" What do you want to see today?");

Console.WriteLine(" 0) Rings");

Console.WriteLine(" 1) Wizards");

Console.WriteLine(" 2) Elfs");

Console.WriteLine(" 3) Orcs");

Console.WriteLine(" 4) Hobbits");

char x = Console.ReadKey().KeyChar;

Console.WriteLine();

switch (x)

{

case '0': handler.ShowGeneric<Ring>(); break;

case '1': handler.ShowGeneric<Wizard>(); break;

case '2': handler.ShowGeneric<Elf>(); break;

case '3': handler.ShowGeneric<Orc>(); break;

case '4': handler.ShowGeneric<Hobbit>(); break;

}

}

}

}

WorldHandler.cs

using System;

using System.Collections;

public class WorldHandler

{

private ArrayList things;

public ArrayList rings;

public ArrayList wizards;

public ArrayList elfs;

public ArrayList orcs;

public ArrayList hobbits;

public WorldHandler()

{

things = new ArrayList();

rings = new ArrayList();

things.Add(rings);

wizards = new ArrayList();

things.Add(wizards);

elfs = new ArrayList();

things.Add(elfs);

orcs = new ArrayList();

things.Add(orcs);

hobbits = new ArrayList();

things.Add(hobbits);

}

public void AddNewOject(object data)

{

if (data is Ring)

{

rings.Add(data);

}

else if (data is Wizard)

{

wizards.Add(data);

}

else if (data is Elf)

{

elfs.Add(data);

}

else if (data is Orc)

{

orcs.Add(data);

}

else if (data is Hobbit)

{

hobbits.Add(data);

}

}

public void AddNewOjectAtIndex(object data, int index)

{

if (data is Ring)

{

rings.Insert(index, data);

}

else if (data is Wizard)

{

wizards.Insert(index, data);

}

else if (data is Elf)

{

elfs.Insert(index, data);

}

else if (data is Orc)

{

orcs.Insert(index, data);

}

else if (data is Hobbit)

{

hobbits.Insert(index, data);

}

}

public void destroyObject(Type type, int index)

{

if (type == typeof(Ring))

{

rings.RemoveAt(index);

}

if (type == typeof(Wizard))

{

wizards.RemoveAt(index);

}

if (type == typeof(Elf))

{

elfs.RemoveAt(index);

}

if (type == typeof(Orc))

{

orcs.RemoveAt(index);

}

if (type == typeof(Hobbit))

{

hobbits.RemoveAt(index);

}

}

public void destroyMultipleObjects(Type type, int from, int to)

{

if (type == typeof(Ring))

{

rings.RemoveRange(from, from - to);

}

if (type == typeof(Wizard))

{

wizards.RemoveRange(from, from - to);

}

if (type == typeof(Elf))

{

elfs.RemoveRange(from, from - to);

}

if (type == typeof(Orc))

{

orcs.RemoveRange(from, from - to);

}

if (type == typeof(Hobbit))

{

hobbits.RemoveRange(from, from - to);

}

}

public void SortGeneric<T>()

{

if (typeof(T) == typeof(Ring))

{

rings.Sort();

}

if (typeof(T) == typeof(Wizard))

{

wizards.Sort();

}

if (typeof(T) == typeof(Elf))

{

elfs.Sort();

}

if (typeof(T) == typeof(Orc))

{

orcs.Sort();

}

if (typeof(T) == typeof(Hobbit))

{

hobbits.Sort();

}

}

public void ShowGeneric<T>()

{

ArrayList current = new ArrayList();

if (typeof(T) == typeof(Ring))

{

current = rings;

}

if (typeof(T) == typeof(Wizard))

{

current = wizards;

}

if (typeof(T) == typeof(Elf))

{

current = elfs;

}

if (typeof(T) == typeof(Orc))

{

current = orcs;

}

if (typeof(T) == typeof(Hobbit))

{

current = hobbits;

}

foreach (T obj in current)

{

Console.WriteLine(obj.ToString());

}

}

}

Elf.cs

using System;

public class Elf : IComparable

{

public string Name { get; set; }

public uint id { get; set; }

public string Category { get; set; }

public Elf()

{

}

public override string ToString()

{

return String.Format("{0} nr. {1} with Name = {2}", "Elf", id, Name);

}

public int CompareTo(object obj)

{

return this.ToString().CompareTo(obj.ToString());

}

}

Hobbit.cs

using System;

public class Hobbit: IComparable

{

public string Name { get; set; }

public uint id { get; set; }

public string Family { get; set; }

public string Region { get; set; }

public Hobbit()

{

}

public override string ToString()

{

return String.Format("{0} nr. {1} with Name = {2}","Hobbit", id, Name);

}

public int CompareTo(object obj)

{

return this.ToString().CompareTo(obj.ToString());

}

}

Orc.cs

using System;

public class Orc : IComparable

{

public uint id { get; set; }

public float Power { get; set; }

public string Master { get; set; }

public Orc()

{

}

public override string ToString()

{

return String.Format("{0} nr. {1} with Master = {2}", "Orc", id, Master);

}

public int CompareTo(object obj)

{

return this.ToString().CompareTo(obj.ToString());

}

}

Ring.cs

using System;

public class Ring : IComparable

{

public uint id { get; set; }

public string Name { get; set; }

public object Owner { get; set; }

public string Material { get; set; }

public string Rune { get; set; }

public Ring()

{

}

public Ring(uint id, string material)

{

this.id = id;

this.Material = material;

}

public override string ToString()

{

return String.Form at("{0} nr. {1} with Name = {2}", "Ring", id, Name);

}

public int CompareTo(object obj)

{

return this.ToString().CompareTo(obj.ToString());

}

}

Wizard.cs

using System;

public class Wizard : IComparable

{

public string Name { get; set; }

public uint id { get; set; }

public string Color { get; set; }

public Wizard()

{

}

public override string ToString()

{

return String.Format("{0} nr. {1} with Name = {2}", "Wizard", id, Name);

}

public int CompareTo(object obj)

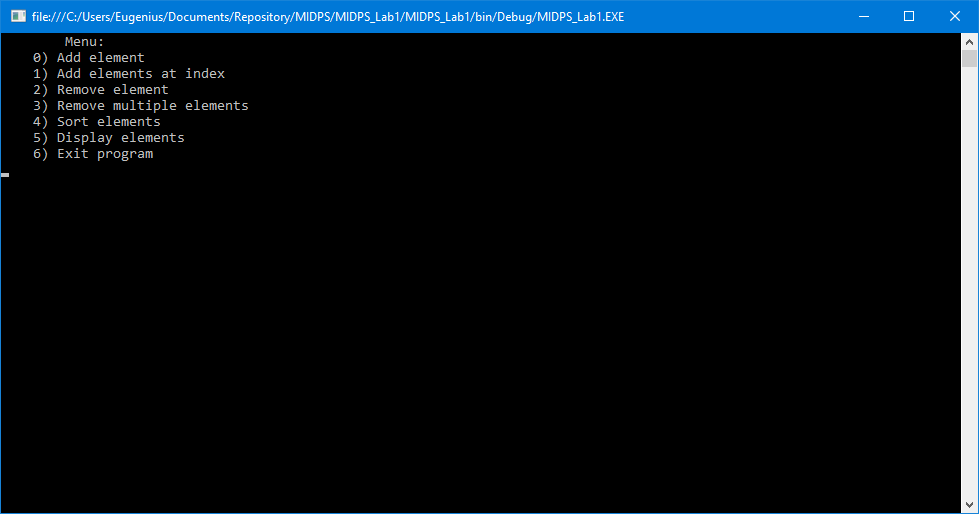
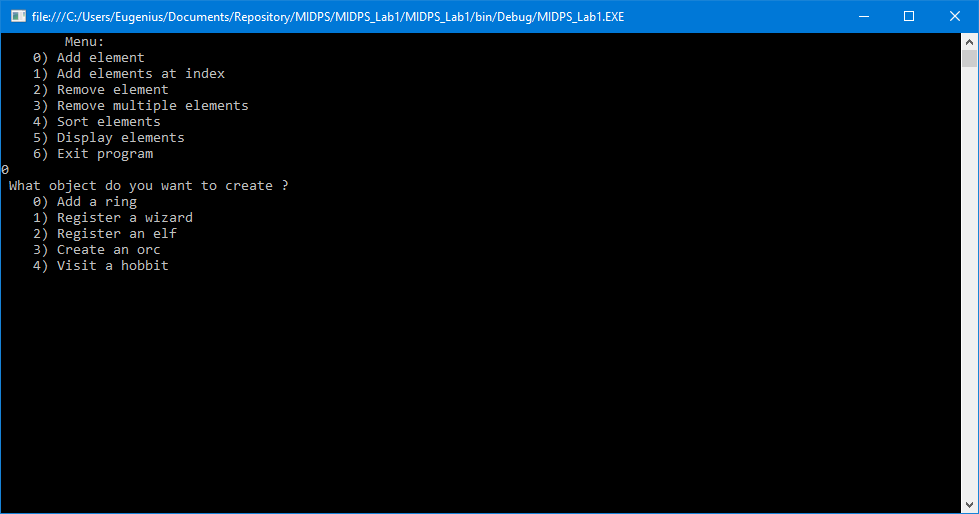
{

return this.ToString().CompareTo(obj.ToString());

}

}

Screenshots:



Conclusion:

En effectuant ce travail de laboratoire, j’ai perdu beaucoup de temps pour trouver des noms aux mes classes. J’ai choisi l’univers de Tolkien, d’où j’ai extrait 5 objets / caractères. Au présent, je regrette mon choix, parce que c’est difficile de faire les autres laboratoires avec ces classes, mais on ne peut pas tourner le temps, et c’est pourquoi je finis cette conclusion et je commence à faire le deuxième laboratoire.