Nom et prénom :……...

Numéro d'inscription : ……

Devoir Suveillé

Matière : ……**Programmation II**
Enseignant(s) :……..…**S.BHAR**
Filière(s) :……**GL3**
Nombre de pages : ……..**4**

Semestre: ……**2**
Date: …… **Mars 2020**
Durée: ……..…**1h30min**
Documents non autorisés

Répondre sur ces mêmes feuilles SVP

Exercice 1. .Net Framework Features

Question 1 *(2 point)*

Donner le résultat de l'exécution du programme C# suivant :

```
class A {
  public virtual void M() { Console.Write("A"); }
}
class B: A {
  public override void M() { Console.Write("B"); }
}
class C: B {
  new public virtual void M() { Console.Write("C"); }
}
class D: C {
  public override void M() { Console.Write("D"); }
  static void Main( ){
     D d = new D(); C c = d; B b = c; A a = b;
     d.M(); c.M(); b.M(); a.M();
  }
}
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Question 2 *(2 points)*

Expliquez qu'est-ce qu'une expression lambda ? Expliquez comment vous pouvez affecter une expression lambda à un délégué ? Donnez un exemple concret.

……..……..

……..……..

……..……..

……..……..

……..……..

……..……..

……..……..

……..……..

Question 2 *(2 points)*

Le code ci-dessous représente une grossière faute de développement, présentez l'erreur, une solution et argumentez :

……..……..

……

……

……

```
1.  IList<Client> clients = ClientsRepository.GetAll();
2.  string commaSeparatedClientsName = string.Empty;
3.
4.  foreach (Client client in clients)
5.  {
6.      commaSeparatedClientsName += client.Name + ",";
7.  }
```

Exercice 2 - Events/Delegates

On gère dans une application les moyenne de notes des étudiants pour lesquels cette moyenne peut passer en négatif après une diminution suite à des absences répétitives. Une moyenne de notes serait représentée par un numéro d'étudiant (chaîne de caractères), une note globale (décimal) et contiendrait deux méthodes recompenser()/punir() pour respectievement la diminution et la hausse de cette note. L'objectif ici est de programmer la levée d'un évènement "**MoyenneNegative**"

Question 1 *(2 point)*

Implémenter la classe **NoteMoyenne** de manière qu'elle lève l'évènement **MoyenneNegative** dès que la note globale passe en négative.

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

Question 2 *(2 point)*

On souhaite programmer maintenant une classe **NegatifsTracker** qui surveille l'évènement **MoyenneNegative** et à chaque instance interceptée elle en affiche le numéro de l'étudiant en question. Ecrire le code de cette classe.

……

……

……

……

……

……

……

……

……

……

……

……

……

Exercice 3 – QCM :

Question 1 : Quelle est la différence entre les delegates Action et Func?

a) Action n'a pas de type de retour, alors que Func en a un.

b) Action s'exécute instantanément alors que Func s'exécute au bout d'un certain délai défini par le développeur.

c) Action est Threadsafe alors que Func ne l'est pas.

d) Action concerne les méthodes statiques alors que Func concerne les méthodes d'instance.

Question 2 : Quelle sera la valeur de la variable foobar à la fin de l'exécution de ce code?

```
1.  int? foobar = 42, dummy = null;
2.  string text = null;
3.  foobar = text != null ? text.Length : 10;
4.  foobar = dummy ?? null;
```

a) 10.

b) null.

c) 4.

d) 42.

Question 3 : Si une méthode est marquée comme protected internal, qui peut y accéder ?

a) Lesclasses qui sont à la fois dans le même assembly et qui dérivent de la classe dans laquelle cette méthode est déclarée.

b) Seules les méthodes qui sont dansla même classe que la méthode en question.

c) Les méthodes Internal ne peuvent être appelées que par reflection.

d) Les classes du même assembly et celles qui dérivent de la classe dans laquelle cette méthode est déclarée.

Question 4 : Le mot clef bool correspond à quel type .NET ?

a) System.Bool

b) System.Boolean

c) System.TrueFalse

d) Aucune de ces réponses

Exercice 4:

Terminer les bouts de code suivants :

Code 1 :

```
…… names = new List<string>() { "John", "Tom", "Peter" };
…… (…… name in names)
{
    Console.WriteLine(name);
}
```

Code 2 :

```
{
    var streamReader = new StreamReader("c:\\file.txt");
    ……
    {
        Console.Write(streamReader.ReadToEnd());
    }
    ……
    {
        if (streamReader != null)
            streamReader.Dispose();
    }
}
```

Code 3 :

```
…… (var streamReader = new StreamReader("c:\\file.txt"))
{
    Console.Write(streamReader.ReadToEnd());
}
```