Examen C#

Consignes : (si vous avez une question, la réponse est sûrement ici)

- La première partie contient des questions auquelles vous devrez répondre dans votre rendu en indiquant le numéro des questions
- La seconde partie concerne des travaux pratiques à réaliser. Vous copierez la totalité du code source dans votre rendu au simple format texte. Il est bien évidemment conseillé de tester son code source dans Visual Studio avant tout pour être certain qu'il fonctionne.
- Notez bien que le fait que votre code soit fonctionnel n'est pas la seule chose qui sera évalué. La méthodologie utilisée, la clarté et la propreté du code seront également des facteurs influançant la notation. Ainsi, il ne faut pas non plus hésiter à copier dans votre rendu un code qui n'est que partiellement fonctionnel pour obtenir une partie des points
- Vous disposez librement de votre cours et d'internet (à l'exception notable des logiciels de communication). Il n'est pas interdit d'utiliser Visual Studio d'une manière ou d'une autre pour vous aider à répondre aux questions de la première partie (mais attention à ne pas y perdre trop de temps)

PREMIERE PARTIE : QUESTIONS (8pts)

1) À combien est égale la variable toto après l'exécution de ses instructions ? (0,5 pt)

```
int toto = 10;

toto--;
toto *= 3;
```

2) Que vont afficher dans la console ces instructions ? (0,5 pt)

```
string toto = "tata", titi = "tutu";

Console.WriteLine(titi + " " + toto);
```

3) Pourquoi le compilateur indique une erreur dans ce code ? (0,5 pt)

```
Random r;

int rn = r.Next(0, 6);
```

4) Quelle est la différence entre une classe et un objet ? (0,5 pt)

5) Qu'est-ce qu'une classe abstraite et à quoi sert-elle ? (0,5 pt)

6) Pourquoi cette classe ne respecte pas le principe d'encapsulation ?(0,5 pt)

```
public class Client
{
    private string firstName;
    private string lastName;
    public int age;

    0 références
    public string GetFullName()
    {
        return lastName + " " + firstName;
    }
}
```

7) Quelle est la particularité de la méthode utilisée dans cette classe ? (0,5 pt)

```
public class Article
{
    1 référence
    public string Name { get; set; }
    1 référence
    public int Price { get; set; }

    0 références
    public Article(string name, int price)
    {
        Name = name ?? throw new ArgumentNullException(nameof(name));
        Price = price;
    }
}
```

8) Qu'est-ce qu'une surcharge ? (0,5 pt)

9) Que vont afficher dans la console ces instructions ? (0,5 pt)

```
static void Main(string[] args)
{
    int t0 = 5, t1 = 6, t2 = 7;
    Sum(t0, t1, ref t2);

    Console.WriteLine(t2);

    Console.Read();
}

1 référence
static void Sum(int a, int b, ref int result)
{
    result = a + b;
}
```

10) Qu'est-ce qu'une propriété ? (0,5 pt)

11) Quelle est la particularité d'un champ associé au mot-clé "static" ? (0,5 pt)

12) Que faut-il ajouter à la classe Vase pour qu'elle implémente correctement IDestroyable ? (0,5 pt)

```
interface IDestroyable
{
    0 références
    void Destroy();
}
class Vase
{
    1 référence
    public int numberOfRubies { get; set; }

    0 références
    public Vase(int numberOfRubies)
    {
        this.numberOfRubies = numberOfRubies;
    }
}
```

13) À quoi sert cette méthode ? (0,5 pt)

```
public void Method<T>(ref T a, ref T b)
{
    T temp = a;

    a = b;
    b = temp;
}
```

14) Pourquoi la méthode ci-dessus utilise un passage par référence ? (0,5 pt)

15) Qu'est-ce que le polymorphisme ? (0,5 pt)

16) Indiquez une signature possible pour une méthode qui pourrait s'abonner à l'événement ci-dessous. (0,5 pt)

```
public event EventHandler<InfoArgs> OnClick;
```

DEUXIEME PARTIE : TRAVAUX PRATIQUES (12pts)

A) **Note pour la partie A** : *La partie A nécessite l'utilisation de la classe Fourmiliere.cs qui vous a été fournie avec cet examen. Vous pouvez l'intégrer dans votre projet par simple glisser-déposer dans votre solution (et en renommant le namespace ou en ajoutant "using MNS;" comme instruction dans les fichiers où vous souhaitez l'utiliser)*

Vous venez de rejoindre un laboratoire qui analyse la prolifération des fourmis au sein d'une fourmilière. Une classe "Fourmiliere.cs" censée simuler cette prolifération vous a été fournie.

Comprenez et utilisez au mieux cette classe pour répondre à la question suivante : combien de fourmis contiendra une fourmilière si l'on y mets 500 fourmis et qu'on les laisse proliférer pendant 12 jours ? (3 pts)

Note : *Seul le résultat vous est demandé, néanmoins toute piste de recherche pourra vous fournir des points. N'hésitez donc pas à copier-coller votre fonction "main" dans votre rendu*

B) Nous souhaitons réaliser un logiciel de gestion des clients d'une agence immobilière. Pour cet exercice nous ne nous focaliserons que sur une très petite partie de ce logiciel permettant de gérer les paiements des locataires aux propriétaires de logements. Cet exercice est divisé en 5 étapes. (4 pts)

ÉTAPE 1 : Créez une classe "Client" comprenant les propriétés "Nom", "Prenom" et "Fonds" (à vous d'en définir le type, "Fonds" représentant les économies sur le compte en banque de notre client).

Créez également un constructeur pour initialiser tous ses champs.

ÉTAPE 2 : Créez une classe "Logement" comprenant les propriétés "Proprietaire", "Locataire", "TarifLocation" et "NomDuLogement", les deux premières représentant des Clients (la classe définie lors de l'Étape 1), la troisième étant un nombre décimal et la quatrième étant une chaîne de caractère.

Créez également un constructeur pour initialiser tous ses champs.

ÉTAPE 3 : Dans la classe "Logement", créez une méthode "PayerLoyer" qui enlèvera des "Fonds" du "Locataire", d'une valeur égale à "TarifLocation" et qui ajoutera aux "Fonds" du "Propriétaire" cette même valeur.

ÉTAPE 4 : Créez une classe "Calendrier" comportant un événement "PremierDuMois" (il symbolise les actions qui devront être effectuées le premier jour de chaque mois). Vous pouvez utiliser le délégué Action ou votre propre délégué, dans chaque cas cet événement ne comportera que des signatures de méthodes sans paramètre et ne renvoyant rien.

Créez également une méthode "AppelerPremierDuMois" qui ne prends aucun paramètre, ne renvoie rien et va simplement déclencher l'événement "PremierDuMois".

ÉTAPE 5 : Modifiez le constructeur de la classe "Logement" pour qu'il prenne un "Calendrier" comme paramètre en plus. Dans le corps de ce constructeur, faîtes en sorte que lorsque que l'événement "PremierDuMois" est déclenché, la méthode "PayerLoyer" du "Logement" en train d'être créé par ce constructeur soit appelée.

Note pour le rendu : *Ne mettez que les classes "Client", "Logement" et "Calendrier" dans votre rendu. Vous pouvez bien entendu utiliser une autre classe contenant un Main pour tester votre code mais il n'est pas nécessaire de la fournir dans le rendu.*

C) League of legends est un jeu vidéo en ligne où des joueurs·euses s'affrontent en équipe dans un univers fantasy et où l'objectif est de détruire la base adverse. Dans ce jeu, vaincre un joueur adverse rapporte des pièces d'ors (ou PO) qui permettront aux joueurs·euses d'acheter des objets pour rendre le personnage qu'ils ou elles incarnent plus puissant lors de la partie. Lors de cet exercice nous nous focaliserons sur cet aspect du jeu : la gestion des objets. Cet exercice est divisé en 4 étapes (5 pt + 0,5 pt BONUS).

ETAPE 1 : Créez une classe "ObjetMagique" contenant des propriétés "Nom" représentant le nom de l'objet et "Valeur" représentant un prix en pièces d'or pour acheter l'objet. Le nom sera une chaîne de caractère et la valeur d'un objet est forcément un nombre entier.

Créez pour la classe un constructeur par défaut et une surcharge de ce constructeur permettant d'initialiser chaque attribut (avec des valeurs par défaut pour le constructeur par défaut ou avec des valeurs passées en paramètre pour la surcharge).

ETAPE 2 : Créez une classe "Joueur" contenant un attribut "inventaire" qui est un tableau permettant de stocker 6 ObjetMagiques (classe créée lors de l'ETAPE 1) ainsi

qu'une propriété "PO" représentant le nombre de pièces d'or que possède le joueur (c'est toujours un nombre entier).
Il n'y a pas de getter et setter pour l'inventaire, c'est un attribut privé.

Créez simplement un constructeur par défaut initialisant la valeur de "PO" à 0 et l'inventaire avec un tableau **de taille 6**.

ETAPE 3 : Créez une méthode dans la classe "Joueur" nommée "AcheterObjet". Cette méthode ne renvoie rien et prends en paramètre un ObjetMagique (classe créée dans l'ETAPE 1) représentant l'objet que l'on souhaite acheter et un nombre entier représentant l'emplacement dans lequel on souhaite stocker l'objet.

Cette méthode vérifie si l'objet peut être stocké (c'est à dire si l'emplacement passé en paramètre est bien dans notre tableau qui ne contient que 6 cases et qu'il n'y a pas déjà un objet à cet emplacement) et si le joueur possède bien un nombre de pièces d'ors égal ou supérieur au prix de l'objet qu'il souhaite acheter.

Si toutes ces conditions sont vérifiées alors le Joueur paye le nombre de pièces d'or nécessaire (donc son attribut "PO" est décrémenté d'autant que la valeur du prix de l'objet) et l'objet est stocké à l'emplacement passé en paramètre dans son inventaire.
Sinon la fonction affiche dans la console "je ne peux pas acheter cet objet".

ETAPE 4 : Créez une méthode dans la classe "Joueur" nommée "AfficherInventaire" qui affiche dans la console le nom des objets dans l'inventaire du joueur.

Attention : si l'emplacement est vide (donc égal à null) alors on affichera plutôt dans la console "vide".

Note : *Pour votre rendu, ne mettez que le code source de vos classes Joueur et ObjetMagique. Vous pouvez bien entendu utiliser une autre classe contenant un main pour tester votre code mais il n'est pas nécessaire de fournir cela dans votre rendu*

BONUS (pour 0,5 pt) : Créez dans la classe "Joueur" des propriétés "PV", "PM", "Attaque" et "Vitesse" représentants les caractéristiques du joueur (PV = Points de Vie et PM = Points de Mana).

Créez dans la classe "ObjetMagique" des propriétés "BonusPV", "BonusPM", "BonusAttaque" et "BonusVitesse".

Modifiez les constructeurs en conséquences.

Puis, faîtes en sorte que lorsqu'un joueur achète un objet, ses attributs augmentent selon les bonus qu'offre l'objet (par exemple, un BonusPV de 10 augmente de 10 la valeur PV du joueur)

PARTIE BONUS (0,5pts de Bonus)

BONUS) Que fait cette fonction ? (0,5 pt)

```
public bool FonctionMystere(bool b1, bool b2)
{
    return b1 ? (b2 ? false : true) : (b2 ? true : false);
}
```