Consignes :

*Répondez à toutes les questions sur l'énoncé ; Les documents au format papier sont autorisés ; Le sujet se compose de deux parties, la première concerne **l'élément JEE** et la deuxième concerne **l'élément .NET** ; Lisez attentivement l'énoncé ; Le sujet est composé de 10 pages ; Éteignez vos téléphones ; Le barème est sur 40 points pour JEE et sur 20 points pour .Net.*

PREMIERE PARTIE : JAVA EE (20 * 2 points)

Répondre au questionnaire à choix multiples (QCM) suivant en entourant la lettre correspondante à la bonne réponse.

Choisir une seule réponse parmi les 4 choix proposés.
Une **réponse correcte = +2 points** ; une **réponse incorrecte = - 0.5** ; **aucune réponse = 0**.
Pour chaque question du QCM, on suppose que :

- ✓ le code est un sous ensemble d'une application Web JAVA EE avec *maven* accessible via *http://localhost:8080/app_exam/*
- ✓ On suppose que toutes les dépendances nécessaires sont bien ajoutées dans *pom.xml* et que l'application est correctement configurée.
- ✓ Les classes Java sont dans un *package* nommé « *com.ensah* » sauf la class *AppConfig* dans le *package* « *com.config* »
- ✓ L'identifiant de session est échangé via des cookies
- ✓ Le code considéré ne contient pas de problèmes de compilation ni d'erreur de syntaxe
- ✓ Les dépendances entres les questions sont mentionnées explicitement dans l'énoncé. Si rien n'est indiqué alors il n y a pas de dépendance.

1) On considère le code suivant d'un filtre et sa configuration dans web.xml

Filtre :

```
public class MyFilter1 implements Filter {
        public MyFilter1() {}
        public void destroy() {}
        public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException,
        ServletException {
                System.out.println("MyFilter1 exécuté");
                chain.doFilter(request, response);
            }
        public void init(FilterConfig fConfig) throws ServletException {}
}
```

Configuration dans web.xml :

```
<filter>
        <filter-name>MyFilter1</filter-name>
        <filter-class>com.ensah.MyFilter1</filter-class>
</filter>
<filter-mapping>
        <filter-name>MyFilter1</filter-name>
        <url-pattern>/MyServlet</url-pattern>
</filter-mapping>
```

On suppose que l'application est constituée uniquement de ce filtre et d'aucune servlet. Si nous exécutons la requête http://localhost:8080/app_exam/test :

a- L'application n'affichera aucun message et elle indiquera que aucune ressource correspondante au mapping /test n'est disponible
b- l'application affichera dans la console le texte : *MyFilter1 exécuté* puis elle indiquera que aucune ressource n'est disponible via le mapping /MyServlet
c- le filtre va lever une exception car il n'existe aucune servlet ayant le nom *MyServlet*
d- l'application affichera le texte : *MyFilter1 exécuté* puis elle affichera une erreur car aucune ressource sur le serveur n'est accessible par le mapping /MyServlet

2) Dans une application Spring MVC, on considère le contrôleur suivant

```
@Controller
public class MyController {

        @Autowired
        private HttpSession session;

       @GetMapping("getStudentData")
       public String getStudentData(Model model) {
```

```
                model.addAttribute("studentModel", session.getAttribute("student"));
                return "index";
        }
}
```

A l'exécution de la requête http://localhost:8080/app_exam/getStudentData :

a- le contrôleur récupèrera un objet de la session par la clé "*student*" et il le stocke dans le modèle, mais si aucun attribut de session n'est trouvé par la clé "*student*" elle déclenchera une exception *AttibuteNotFoundException*.
b- On obtiendra une exception de type *NullPointerException* car le modèle n'est pas initialisé.
c- Il n'y aura pas d'exception et le contrôleur récupèrera un objet de la session par la clé "*student*" et il le stocke dans le modèle
d- Il n'y aura pas d'exception et le contrôleur récupèrera l'étudiant dont l'id est passé en paramètre et stocke cet étudiant dans la session.

3) On suppose qu'une classe « *User* » existe déjà dans une application Spring MVC, et on considère le contrôleur suivant

```
@Controller
public class MyController {

        @Autowired
        private HttpSession session;

        private IService service;

        @GetMapping("getUserById")
        public String getUserById(@RequestParam int idUser) {
                User user  = service.getUserById(Long.valueOf(idUser));
                session.setAttribute("user", user);
                return "listStudent";
        }
}
```

A l'exécution de la requête http://localhost:8080/app_exam/getUserById?idUser=1 :

a- Il y aura une exception de type *NullPointerException* car la variable *session* est null.
b- On obtiendra une exception de type *NullPointerException* car la méthode *getUserById* n'a pas un objet modèle en paramètre
c- Il n'y aura pas d'exceptions et le contrôleur récupère l'id d'un utilisateur envoyé comme paramètre de la requête puis utilise un service métier injecté par Spring pour récupérer l'utilisateur correspondant de la base de données puis il le met dans la session.
d- On obtiendra une exception de type *NullPointerException* car l'attribut *service* est null.

4) Dans une application Spring MVC, on considère le contrôleur REST suivant

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
        @Autowired
        private IStudentService studentService;
        @GetMapping("/students/{idStudent}")
        public Student getStudentById(@PathVariable int idStudent) throws {
           //code ici . . .
        }
```

4.1. Si nous voulons appeler ce service depuis un client REST pour récupérer les données d'un étudiant ayant l'id =1. Nous allons utiliser :

a- GET http://localhost:8080/app_exam/api/students?idStudent=1
b- POST http://localhost:8080/app_exam/api/getUserById/students/1
c- GET http://localhost:8080/app_exam/api/getUserById/students/1
d- GET http://localhost:8080/app_exam/api/students/1

4.2. Après l'appel de ce service REST, Spring MVC :

a- rediriger vers une vue qui sera transformée automatiquement par Jackson en JSON.
b- retournera via ce contrôleur un objet de type Student qui sera sérialisé par JPA.
c- retournera un objet Student qui sera convertit automatiquement en JSON.
d- retournera un objet Student et qui sera transformé par le client en JSON avec Jackson.

5) On considère le code donné ci-dessous extrait d'une application Spring MVC :

MyController.java

```
package com.ensah;
@Controller
public class MyController {
    @GetMapping("/test")
    public String test() {
```

```
            return "test";
        }
    }
```

ServiceA.java

```
package com.ensah;
public interface ServiceA {
}
```

ServiceAImpl.java

```
package com.ensah;
import org.springframework.stereotype.Service;
public class ServiceAImpl implements ServiceA {
    public ServiceAImpl() {
            System.out.println("Je suis un Service A");
    }
}
```

AppConfig.java

```
package com.config;
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = { "com.ensah" })
public class AppConfig implements WebMvcConfigurer {
        @Bean
        public ViewResolver internalResourceViewResolver() {
                InternalResourceViewResolver bean = new InternalResourceViewResolver();
                bean.setViewClass(JstlView.class);
                bean.setPrefix("/WEB-INF/");
                bean.setSuffix(".jsp");
                return bean;
        }
}
```

5.1. A l'exécution de la requête http://localhost:8080/app_exam /test **, Spring MVC**

a- exécutera la méthode test() de *MyController* et redirigera vers une autre méthode annotée avec @GetMapping("/test")
b- exécutera la méthode test() de *MyController* et redirigera vers la page */WEB-INF/view/test.jsp*
c- exécutera la méthode test() de *MyController* d'une manière récursive jusqu'à obtenir une erreur
d- exécutera la méthode test() de *MyController* et redirigera vers la page */WEB-INF/test.jsp*

5.2. Dans le même code précédent on ajoute l'annotation *@Service* sur la classe ServiceAImpl et on ajoute les deux attributs suivants dans la classe MyController :

```
@Autowired
private ServiceA serviceA1;
@Autowired
private ServiceA serviceA2;
```

Choisir la bonne réponse :

a- Il n'y aura aucune erreur mais il n'y aura pas d'affichage dans la console.
b- Au démarrage de l'application, Spring MVC va créer deux instances de ServiceAImpl et il va injecter les dépendances et il y aura donc l'affichage suivant dans la console :
```
Je suis un Service A
Je suis un Service A
```
c- Au démarrage de l'application, Spring MVC va créer une instance de ServiceAImpl et il va injecter les dépendances et il y aura affichage suivant dans la console :
```
Je suis un Service A
```
d- On va obtenir une exception car on ne peut pas injecter le même service deux fois

5.3. Dans le même code précédent (celui de la question 5.2) on ajoute dans la classe ServiceAImpl après l'annotation @Service l'annotation @Scope("prototype") . Choisir la bonne réponse :

a- Il n'y aura aucune erreur mais il n'y aura pas d'affichage dans la console.
b- Au démarrage de l'application, Spring MVC va créer deux instances de ServiceAImpl et il va injecter les dépendances et il y aura donc l'affichage suivant dans la console :
```
Je suis un Service A
Je suis un Service A
```
c- Au démarrage de l'application, Spring MVC va créer une instance de ServiceAImpl et il va injecter les dépendances et il y aura affichage suivant dans la console :
```
Je suis un Service A
```

d- On va obtenir une exception car on ne peut pas injecter le même service deux fois

5.4. Dans le même code précédent (*celui de la question 5.3*), on ajoute le contrôleur suivant et on exécute depuis un même poste client et un même navigateur la requête http://localhost:8080/app_exam/test1 **suivie directement de la requête** http://localhost:8080/app_exam/test2**.**

```
package com.ensah;
@Controller
public class BController {
     @Autowired
     private ServletContext appContext;

    @Autowired
    private HttpSession session;

      @GetMapping("/test1")
      public String test1() {
             appContext.setAttribute("test", "Bonjour");
             session.setAttribute("test", "Bonjour");
              return "test";
      }
      @GetMapping("/test2")
      public String test2() {
            System.out.println(appContext.getAttribute("test"));
            System.out.println( session.getAttribute("test"));
            return "test";
      }
}
```

Choisir la bonne réponse :

a- On va obtenir une exception car on ne peut pas utiliser la même clé « test » pour stocker des données dans la session et le contexte de de servlet.

b- Il y aura l'affiche suivant dans la console:
```
Bonjour
Bonjour
```

c- Il y aura l'affiche suivant dans la console:
```
Bonjour
```

d- On va obtenir une exception car on ne peut pas accéder au contexte de servlet et la session en même temps dans une application web.

5.5. Dans le même code précédent (*celui de la question 5.4*), on exécute depuis un navigateur client la requête http://localhost:8080/app_exam/test1 **on efface toutes les cookies enregistrés sur le navigateur client puis on exécute la requête** http://localhost:8080/app_exam/test2 . **Choisir la bonne réponse :**

a- Il y aura l'affiche suivant dans la console:
```
Null
Bonjour
```

b- Il y aura l'affiche suivant dans la console:
```
Bonjour
Bonjour
```

c- Il y aura l'affiche suivant dans la console:
```
null
null
```

d- Il y aura l'affiche suivant dans la console:
```
Bonjour
null
```

5.6. Dans le même code précédent (*celui de la question 5.5*), On ajoute un intercepteur et on change le code du contrôleur *BController*:

On ajoute la configuration suivante dans la classe AppConfig.java

```
@Bean
MyInterceptor myInterceptor() {
return new MyInterceptor();
}
@Override
public void addInterceptors(InterceptorRegistry registry) {
 registry.addInterceptor(myInterceptor())
         .addPathPatterns("/B/**") ;
}
```

Dans la classe MyInterceptor qui définit l'intercepteur on définit la méthode *preHandle* ainsi :

```
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
throws Exception {
              if("ensah".equals(request.getParameter("nom"))  ) {
                        request.setAttribute("test", "Bonjour ENSAH");
              }
              return true;
}
```

Le nouveau code du contrôleur :

```
package com.ensah;
@Controller
public class BController {
    @Autowired
    private HttpSession session;
        @GetMapping("/B/test1")
        public String test1(HttpServletRequest rq) {
            if(rq.getAttribute ("test") !=null){
              session.setAttribute("test", rq.getAttribute("test")) ;
             }
            else{
            session.setAttribute("test", "FSTH");
            }
            return "test";
        }
        @GetMapping("/B/test2")
        public String test2(HttpServletRequest rq) {
           System.out.println(rq.getAttribute("test"));
                return "test";
        }
        @GetMapping("/test3")
        public String test3(HttpServletRequest rq) {
           System.out.println( session. getAttribute ("test"));
          return "test";
        }
}
```

On exécute la requête http://localhost:8080/app_exam/B/test1?nom=ttt **puis** http://localhost:8080/app_exam/test3?nom=ensah **depuis le même navigateur l'une après l'autre directement. L'application affichera :**

a- Le texte "ttt"
b- Le texte " FSTH"
c- Le texte " Bonjour ENSAH"
d- null

5.7. Dans le même code précédent (*celui de la question 5.6*), on efface les cookies de session et on exécute http://localhost:8080/app_exam/B/test1?nom=ttt **puis on exécute depuis le même navigateur directement** http://localhost:8080/app_exam/B/test2?nom=ensah **L'application affichera:**

a- Le texte "ttt"
b- Le texte " FSTH"
c- Le texte " Bonjour ENSAH"
d- null

5.8. Dans le même code précédent (*celui de la question 5.7*), on efface les cookies de session on exécute la requête http://localhost:8080/app_exam/B/test1?nom=ttt **puis on ré-efface les cookies de session et on exécute** http://localhost:8080/app_exam/test3?nom=ensah **depuis le même navigateur.**

L'application affichera :

a- Le texte "ttt"
b- Le texte " FSTH"
c- Le texte " Bonjour ENSAH"
d- null

6) **On considère les deux classes suivantes annotées avec les annotations de JPA :**

On suppose que chaque attribut a son *getter* et son *setter*.

```java
@Entity(name = "ETUDIANT_TAB")
public class Etudiant {
    @Id
    @GeneratedValue(generator = "increment")
    @GenericGenerator(name = "increment", strategy = "increment")
    private Long id;
    private String nom;
    private String cin;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name="id_etudiant")
    private Set<Adresse> adresses;

    public Set<Adresse> getAdresses() {
        return adresses;
    }
```

```java
@Entity
public class Adresse {

    @Id
    @GeneratedValue(generator = "increment")
    @GenericGenerator(name = "increment", strategy = "increment")
    @Column(name = "id")
    private Long id;


    private String ville;
```

On rappelle que Hibernate utilise le chargement tardif (*Lazy loadding*) pour les collections. On suppose qu'un étudiant ayant l'id=1 existe dans la table ETUDIANT_TAB et qu'il a au moins une adresse. On suppose que la *SessionFactory* est référencée par la variable `sf`.

6.1. on exécute les instructions suivantes

```
1.      Session s = sf.openSession() ;
2.      Transaction tx = s.beginTransaction() ;
3.      //On suppose qu'un étudiant avec cet id existe en BD
4.      Etudiant e = s.get(Etudiant.class, Long.valueOf(1)) ;
5.      tx.commit() ;
6.      s.close() ;
7.      List<Adresse> adresses = e.getAdresses() ;
8.      for(Adresse a : adresses){
9.      System.out.println(a.getVille()) ;
10.     }
```

Après l'exécution il y aura :

a- L'appel de *getAdresses* provoquera une exception car *getAdresses* retourne null et nous ne pouvons pas parcourir null
b- Affichage des villes de l'étudiant ayant l'id = 1 à la console
c- Une exception liée à l'absence d'une session pour charger les adresses à la ligne 7
d- Il y aura une exception car les transactions dans Hibernate ne peuvent pas être utilisées en lecture

6.2. Après l'exécution de ces instructions

```
1.          Session s = sf.openSession() ;
2.          Transaction tx = s.beginTransaction() ;
3.          //On suppose qu'un étudiant avec cet id existe en BD
4.          Etudiant e = s.get(Etudiant.class, Long.valueOf(1)) ;
5.          List<Adresse> adresses = e.getAdresses() ;
6.          for(Adresse a : adresses){
7.          System.out.println(a.getVille()) ;
8.          }
9.          tx.commit() ;
10.         s.close() ;
```

a- L'appel de *getAdresses* provoquera une exception car *getAdresses* retourne null et nous ne pouvons pas parcourir null
b- Affichage des villes de l'étudiant ayant l'id = 1 à la console
c- Une exception liée à l'absence d'une session pour charger les adresses à la ligne 5
d- Il y aura une exception car les transactions dans Hibernate ne peuvent pas être utilisées en lecture

6.3. Après l'exécution de ces instructions

```
1.          Session s = sf.openSession() ;
2.          Transaction tx = s.beginTransaction() ;
3.          //On suppose qu'un étudiant avec cet id existe en BD
4.          Etudiant e = s.get(Etudiant.class, Long.valueOf(1)) ;
5.          e.setNom("Boudaa");
6.          tx.commit() ;
7.          s.close() ;
```

a- Il y aura une exception car on ne peut pas faire une transaction contenant à la fois des lectures et des écritures
b- Le nom de l'étudiant ayant l'id=1 sera changé dans la base de données
c- Le nom ne sera pas changé car il faut appeler la méthode *update* ou *merge* après la ligne 5
d- Le nom ne sera pas changé car il faut appeler la méthode *save ou saveOrUpdate* après la ligne 5

7) **Dans une application JEE à trois couches :**
   a- Il est impossible techniquement de gérer les transactions dans une autre couche autre que les DAOs
   b- Pour pouvoir réutiliser les DAOs par plusieurs services métier il vaut mieux implémenter la gestion des transactions dans les méthodes des DAOs.
   c- La gestion des transactions est une tâche à gérer dans le code SQL et non pas dans le code Java de l'application.
   d- Gérer les transactions au niveau de la couche services va permettre de réutiliser les DAOs tout en personnalisant la gestion des transactions selon les besoin de chaque service métier

8) **Lors de l'utilisation du Framework Spring pour la gestion des transactions**
   a- La gestion des transactions est une problématique gérée uniquement au niveau des bases de données et non pas au niveau du code applicatif
   b- Le code dépend fortement du Framework de persistance utilisé
   c- Spring utilise une abstraction de gestionnaire de transaction réellement utilisé pour offrir une robustesse au code métier
   d- Le code des transactions est géré au niveau DAO par le Framework de persistance et donc il n'a aucune relation avec Spring

9) **Nous avons défini un service métier en l'annotant par @Service et @Transactional(isolation = Isolation.SERIALIZABLE)**
   a- On ne peut pas ajouter l'annotation **@Transactional(isolation = Isolation.SERIALIZABLE)** sur un service métier car dans Spring cette annotation est réservée aux DAOs
   b- On ne pourra pas avoir des problèmes de lecture sale, lecture fantôme et non reproductible mais les performances de ce service peuvent être grandement impactées
   c- On peut avoir des problèmes de lecture sale car les transactions sont gérées par le SGBD
   d- Dans ce cas les données récupérées par ces transactions seront automatiquement sérialisées dans des fichiers par Spring

10) **Dans une application Java Web standard basée sur les Servlets :**
   a- le conteneur prend en charge la gestion des threads et leur synchronisation et il n'y aura aucun risque d'accès concurrent
   b- les requêtes des différents utilisateurs s'exécutent chacune dans un processus séparé ainsi il n'y a aucun risque de Thread-safty
   c- les requêtes des différents utilisateurs s'exécutent chacune dans un thread séparé et ainsi il faut prendre en compte les problèmes de concurrence.
   d- les requêtes des différents utilisateurs s'exécutent toutes dans un même thread

=== Fin de la première partie ===

DEUXIEME PARTIE : MICROSOFT .NET (8* 2 points + 4 points)

Répondre aux questions dans les zones de réponses correspondantes.

Pour toutes les questions on considère que toutes les classes sont définies dans le même espace de nom. Et pour simplicité nous avons omis les directives *using*

1) Mettre la lettre correspondante à une vraie information dans la zone de réponse (*2 points ou 0 point*) :

   a- Lors de la compilation par la CLR d'un programme C# un fichier .dll est généré contant le code machine.
   b- Lors de la compilation par la CLR d'un programme C# un fichier .exe ou .dll est généré contant le code machine.
   c- Lors de la compilation d'un programme C# par le compilateur C# un fichier .exe ou .dll est générée contant le code CIL.
   d- Les Assembly .Net contiennent le code compilé en code Assembleur.

Réponse à la question 1 :

2) On considère la classe suivante en C#

```
public class Program
 {
     public delegate double calcul(double x, double y);
     public double calculer(calcul c, double x, double y)
     {
         return c(c(x, y), y);
     }
     public static double somme(double x, double y)
     {
         return x + y;
     }
     public static double produit(double x, double y)
     {
         return 2*(x * y);
     }
```

```
        public  void changer1(double x)
        {
            x = x + 3;
        }

        public void changer2(ref double x)
        {
            x = x + 1;
        }

    }
```

2.1. Qu'affiche le programme suivant ? (2 points ou 0 point)

```
       static void Main(string[] args)
        {
            Console.WriteLine(new Program().calculer(produit, 2, 3));
        }
```

Réponse à la question 2.1 :

2.2. Qu'affiche le programme suivant ? (2 points ou 0 point)

```
static void Main(string[] args)
  {
      calcul d1 = produit;
      double a = new Program().calculer(d1, 1, 2);
      calcul d2 = somme;
      double b = new Program().calculer(d1, 2, 1);
      Console.WriteLine(new Program().calculer(produit, a, b));
  }
```

Réponse à la question 2.2 :

2.3. Quelle est l'instruction qui causera une erreur de compilation ? (2 points ou 0 point)

```
static void Main(string[] args)
  {
      calcul d1 = produit * somme;
      double a = new Program().calculer(d1, 1, 2);
      calcul d2 = somme;
      double b = new Program().calculer(d1, 2, 1);
      Console.WriteLine(new Program().calculer(produit, a, b));
  }
```

Réponse à la question 2.3 :

2.4. Qu'affiche le programme suivant ? (2 points ou 0 point)

```
 static void Main(string[] args)
  {

      Program p = new Program();
      double a = p.calculer(produit, 1, 2);
      Console.WriteLine(a);
      p.changer1(a);
      Console.WriteLine(a);


  }
```

Réponse à la question 2.4 :

2.5. Qu'affiche le programme suivant ? (2 points ou 0 point)

```
   static void Main(string[] args)
    {
      Program p = new Program();
      double a = 1;
      p.changer1(a);
      double b = p.calculer(produit, a, 2);
      Console.WriteLine(b);
      p.changer2(ref b);
      Console.WriteLine(b);

 }
```

Réponse à la question 2.5 :

3) On considère la classe ci-dessous

```
public  class Program
 {
    public void executer(Action<double[,]> c, double[,] tab)
     {
         c(tab);
     }

     public void traitement1(double[,] tab)
     {
         for (int i = 0; i < tab.GetLength(1); i++)
         {
             tab[0, i] = -tab[0, i];
         }
     }
 }
```

3.1. Qu'affiche le programme suivant ? (*2 points ou 0 point*)

Réponse à la question 3.1 :

```
static void Main(string[] args)
{
    Program p = new Program();
    Action<double[,]> t = (tab)=>
    {
        for (int i = 0; i < tab.GetLength(0); i++)
        {
            for (int j = 0; j < tab.GetLength(1); j++)
            {
                Console.Write(" {0} ", tab[i, j]);
            }
            Console.Write("\n");
        }
    };
    double[,] tab = { { 1, 2, -1 }, { 2, 3, -2 }, { 2, 3, 3 } };
    t(tab);
}
```

3.2. Qu'affiche le programme suivant : (2 points ou 0 point)

Réponse à la question 3.2 :

```
static void Main(string[] args)
  {
      Program p = new Program();
      Action<double[,]> t = p.traitement1;
      t += (tab) =>
      {
          for (int i = 0; i < tab.GetLength(0); i++)
          {
              for (int j = 0; j < tab.GetLength(1); j++)
              {
                  Console.Write(" {0} ", tab[i, j]);
              }
              Console.Write("\n");
          }
      };

      double[,] tab = { { 1, 2, -1 }, { 2, 3, -2 }, { 2, 3, 3 } };
      t(tab);

  }
```

4) On considère les classes C# suivantes

Classe Ballon

```
public class Ballon : Point
{
    public delegate void delegatePoint(Point p);
    public event delegatePoint eventChange;
    public Ballon(Point position) : base (position.x, position.y, position.z)
    {

    }
```

```
    public void bouger()
    {
        x=x+1; y=y+1; z=z+2;
        if (eventChange != null)
        {
            eventChange(this);
        }
    }
}
```

Classe GardienBut

```
  public  class GardienBut
    {
        public Point position { get; set; }
        public Ballon bal;

        public GardienBut(Ballon pBal, Point p) {
            this.bal = pBal;
            this.position = p;
            bal.eventChange += FaireUnSaut;
        }

        public void FaireUnSaut(Point p)
        {
            position.x = p.x; position.y = p.y; position.z = p.z;
            Console.WriteLine("Sauter vers " + position);
        }
    }
```

Classe Point

```
 public   class Point
    {
        public double x { get; set; }
        public double y { get; set; }
        public double z { get; set; }

        public Point(double pX, double pY, double pZ) { this.x = pX; this.y = pY; this.z = pZ; }
        public override String ToString(){return "(" +x+","+y+","+z+ ")"; }

  }
```

Qu'affiche le programme suivant ? (*4 points ou 0 point*)

```
  static void Main(string[] args)
    {
        //Création d'un ballon
        Ballon b = new Ballon(new Point(0, 0, 0));
        //Créer un gardien de but
        GardienBut g = new GardienBut(b, new Point(0, 0, 0));

        while(b.x < 3)
        {
            b.bouger();
        }

    }
```

Réponse à la question 4 :

=== Fin de l'énoncé ===