FICHE UE

IDENTIFICATION
Section : Bachelier en automatisation
Intitulé de l'UE : Programmation orientée objet
Code de référence : 752521U32D3
Nombre de périodes : 120
Nombre de crédits ECTS : 9

DESCRIPTION

Prérequis ou documents de référence pour une préparation préalable au cours :

Prérequis :

mettre en oeuvre une stratégie cohérente de résolution du problème posé ;

- concevoir, de construire et de représenter l'(les) algorithme(s) correspondant(s) ;
- justifier la démarche algorithmique et les choix mis en oeuvre ;
- développer des programmes en respectant les spécificités du langage choisi ;
- mettre en oeuvre des procédures de test.
-

Attestation de réussite de l'unité de formation « Principes algorithmiques et

programmation », n° de code 7521 05 U32 D2, classée dans l'enseignement supérieur

économique de type court informatique de gestion

ou
Attestation de réussite des unités de formation initiation à la programmation **7560 41U31 D1**

FICHE UE

et **7560 42U31 D1** dans l'enseignement de type court informatique systèmes (automatisation)

-

Finalités particulières :

L'unité d'enseignement vise à permettre à l'étudiant :

♦ de développer des comportements professionnels :

o s'intégrer dans une équipe de développement de projet ou de réalisation ;

o identifier les compétences à développer pour adapter ses productions à l'évolution des langages et des besoins de la clientèle ;

♦ de mettre en œuvre, d'une manière appropriée des techniques, des méthodes spécifiques pour :

o réaliser des applications réactives et interactives en mettant en œuvre les principes de programmation événementielle ;

o assurer la maintenance du programme réalisé

Contenu du cours :

1. Introduction ..... **Erreur ! Signet non défini.**
2. Installation de Visual Studio ..... **Erreur ! Signet non défini.**
3. Installation de SQL Server et SQL Server Management Tools .... **Erreur ! Signet non défini.**
4. Création d'une application Brasserie en C# MAUI ..... **Erreur ! Signet non défini.**
5. Le modèle Objet – Classe (`Class`) et objet de classe (Object) .... **Erreur ! Signet non défini.**
6. Cahier des charges de la classe Person ..... **Erreur ! Signet non défini.**
7. Création des classes ..... **Erreur ! Signet non défini.**
8. Champs de classe ..... **Erreur ! Signet non défini.**
9. Instanciation et constructeur d'objet ..... **Erreur ! Signet non défini.**
10. Encapsulation (Wrapping) ..... **Erreur ! Signet non défini.**
    - 10.1. Propriétés, accesseurs get et set ..... **Erreur ! Signet non défini.**
    - 10.2. Exercices ..... **Erreur ! Signet non défini.**
    - 10.3. Propriété automatique (Auto implemented property) ..... **Erreur ! Signet non défini.**
11. Membres `static` ..... **Erreur ! Signet non défini.**
    - 11.1. Méthodes statiques ..... **Erreur ! Signet non défini.**
    - 11.2. Champs et propriétés statiques ..... **Erreur ! Signet non défini.**
    - 11.3. Exercices ..... **Erreur ! Signet non défini.**

12. L'héritage ..... **Erreur ! Signet non défini.**
12.1. Introduction, étude du cas de la brasserie ..... **Erreur ! Signet non défini.**
12.2. Définition et application de l'héritage en C# ..... **Erreur ! Signet non défini.**
12.3. Le diagramme de classes et concepteur de classes. ..... **Erreur ! Signet non défini.**
12.4. Constructeurs de classes dérivées - base ..... **Erreur ! Signet non défini.**
12.4.1. Exercices ..... **Erreur ! Signet non défini.**
12.5. Le modificateur d'accès protected ..... **Erreur ! Signet non défini.**
13. Les collections d'objets ..... **Erreur ! Signet non défini.**
13.1. Collections génériques ..... **Erreur ! Signet non défini.**
13.2. Exemple d'utilisation dans le projet brasserie ..... **Erreur ! Signet non défini.**
13.3. Collections et lambdas ..... **Erreur ! Signet non défini.**
13.3.1. Exercices ..... **Erreur ! Signet non défini.**
13.4. Classe de type Collection d'objets ..... **Erreur ! Signet non défini.**
13.4.1. Exercices ..... **Erreur ! Signet non défini.**
13.5. Utilisation de collections pour la lecture et écriture de fichiers texte .... **Erreur ! Signet non défini.**
14. Polymorphisme ..... **Erreur ! Signet non défini.**
14.1. virtual , masquage new, remplacement override ..... **Erreur ! Signet non défini.**
14.2. Exercices ..... **Erreur ! Signet non défini.**
15. Les Interfaces ..... **Erreur ! Signet non défini.**
15.1. Les classes abstraites – `abstract class` ..... **Erreur ! Signet non défini.**
15.2. Membres abstraits ..... **Erreur ! Signet non défini.**
15.3. Interface - `interface` ..... **Erreur ! Signet non défini.**
15.4. Implémentation d'interface pour l'accès aux données Csv **Erreur ! Signet non défini.**
15.4.1. Exercices ..... **Erreur ! Signet non défini.**
15.5. Implémentation d'interface pour l'accès aux données json **Erreur ! Signet non défini.**
16. Pop up d'interaction avec l'utilisateur - DisplayAlert ..... **Erreur ! Signet non défini.**
17. Singleton et injection de dépendance ..... **Erreur ! Signet non défini.**
18. Desgin MVVM - ViewModel - Binding : première approche .. **Erreur ! Signet non défini.**
18.1. Exercices ..... **Erreur ! Signet non défini.**
19. Binding ..... **Erreur ! Signet non défini.**
19.1. Introduction ..... **Erreur ! Signet non défini.**

19.1. Configuration pour le Binding ..... **Erreur ! Signet non défini.**
19.1.1. INotifyPropertyChanged ..... **Erreur ! Signet non défini.**
19.1.2. Exercices ..... **Erreur ! Signet non défini.**
20. Navigation - Popup ..... **Erreur ! Signet non défini.**
20.1. Exercices ..... **Erreur ! Signet non défini.**
21. Accès à une base de données ..... **Erreur ! Signet non défini.**
21.1. Configuration SQL Server ..... **Erreur ! Signet non défini.**
21.2. DataAccessSql C# ..... **Erreur ! Signet non défini.**
21.3. Exercices ..... **Erreur ! Signet non défini.**
22. Les délégués - delegate ..... **Erreur ! Signet non défini.**
22.1. Utilisation d'un délégué pour la société Solutech ..... **Erreur ! Signet non défini.**
22.2. Exercices sur les délégués ..... **Erreur ! Signet non défini.**
23. Les événements - event ..... **Erreur ! Signet non défini.**
23.1. Délégués et méthodes cibles multiples ..... **Erreur ! Signet non défini.**
23.2. Du délégué à méthodes cibles multiples à l'événement .... **Erreur ! Signet non défini.**
23.3. L'événement : mécanisme et implémentation ..... **Erreur ! Signet non défini.**
23.3.1. Délégué générique EventHandler<T> ..... **Erreur ! Signet non défini.**
23.4. Exercices sur les événements ..... **Erreur ! Signet non défini.**
24. Accès à une base de données ..... **Erreur ! Signet non défini.**
24.1. ADO.Net ..... **Erreur ! Signet non défini.**
24.1.1. Création de la base de données SQL Server ..... **Erreur ! Signet non défini.**
24.1.2. Implémentation d'une classe utilisant ADO.Net ..... **Erreur ! Signet non défini.**
24.1.3. Requête sur tables avec relations ..... **Erreur ! Signet non défini.**
25. Tests unitaires ..... **Erreur ! Signet non défini.**
25.1. Exercices ..... **Erreur ! Signet non défini.**
26. Les threads ..... **Erreur ! Signet non défini.**
26.1. Utilisation des threads et communication TCP/IP, création d'une interface C# pour le contrôle d'Arduinos. ..... **Erreur ! Signet non défini.**
26.1.1. Exercices proposés ..... **Erreur ! Signet non défini.**
26.2. Supervision et communication Modbus TCP ..... **Erreur ! Signet non défini.**
27. Annexes ..... **Erreur ! Signet non défini.**
27.1. Body Expression ..... **Erreur ! Signet non défini.**

FICHE UE

27.2. Délégués génériques `Func<>` et `Action<>` **Erreur ! Signet non défini.**

Bibliographie :

Olivier Dahan. .Net MAUI Le Guide Complet

PERSONNEL(S) ENSEIGNANT(S)

LEBLOND Vincent

METHODOLOGIE

Ce cours de programmation orientée objet (POO) est destiné au débutant en programmation objet avec, comme prérequis, l'algorithmique et des bases de C# ou Java. Un fil conducteur : une application gérant une brasserie, sert de prétexte pour aborder les différents piliers de la POO. Ceux-ci sont donc initiés dans ce contexte précis, puis généralisés, et enfin contextualisés dans d'autres domaines. A chaque chapitre, des exemples de code sont fournis de façon à pouvoir tester et apprendre, en autonomie, les notions élémentaires de POO. Le langage utilisé est le C# et principalement pour des applications MAUI. Ce genre de projet permet de déployer l'application vers des applications de bureau Windows, des applications mobiles Android, iOS, …
Le plus important étant d'acquérir les fondamentaux de la POO, l'essentiel n'est pas d'investiguer trop longuement des techniques spécifiques au C#. Cependant, les cours précédents étant concentrés exclusivement sur la production de résultats à la console, on ne peut continuer ainsi sans utiliser les interfaces graphiques sous peine de décourager les développeurs en herbe, le but étant, au contraire, de déclencher des passions et susciter des vocations pour l'univers du codage. Il est à noter que les solutions proposées dans ce cours ne sont pas forcément celles utilisées par le codeur professionnel, celles qui sont les plus efficaces, mais bien celles qui se trouvent à un niveau compréhensible par le débutant en POO et dans ses zones proximales de développement. Le langage balisé XAML propre aux projets C# WPF, …, MAUI sera utilisé pour coder nos interfaces et les lier à nos données. Même s'il est spécifique au C#, il permet d'introduire le codage en langage balisé et XML. La structuration des projets se fera en UML par un diagramme de classes réutilisant ainsi le cours d'analyse informatique. Des fonctionnalités spécifiques comme des communications entre appareils (Arduino, API) en TCP/IP (ethernet, ModBus), des interactions avec des SGDB seront abordées. Les étudiants sont invités à développer un projet personnel (dans certains cas par équipe), sur un sujet propre à leur section, réinvestissant toutes

les notions vues au cours.

SUPPORTS

- Slides présentés mis à disposition au format pdf
- Document de cours POO MAUI CSharp.pdf (V.Leblond)

MODES D'ÉVALUATION ET ACQUIS D'APPRENTISSAGE

1° Test à cours ouvert en fin d'unité.

2° Projet personnel à présenter et défendre devant la classe.

Les deux éléments doivent être réussis pour obtenir la réussite.

Pour atteindre le seuil de réussite, l'étudiant devra prouver qu'il est capable,

En disposant d'une structure informatique matérielle et logicielle opérationnelle, d'une documentation appropriée, les consignes de réalisation de l'application lui étant précisées,

- de concevoir, d'installer et d'utiliser des objets appropriés à la solution ;
- de concevoir et mettre en oeuvre une procédure de test partiel et intégré ;

de justifier sa méthode de résolution ainsi que ses choix conceptuels et méthodologiques.

UTILISATION DE L'IA

L'IA est conseillée pour illustrer les concepts fondamentaux de la POO dans d'autres contextes que ceux présentés au cours.

L'IA peut se révéler très performante dans la génération de code C# MAUI. A ce stade initial de la formation en POO, le danger réside dans le fait de créer des parties de projet par l'IA et sans

l'encoder manuellement, de questionner l'IA à la moindre difficulté ou pire de copier-coller des parties de codes sans en comprendre les tenants et aboutissants.

C'est dans cette optique qu'une défense à lieu avec réponse aux questions portant sur le code fourni. Celui-ci doit pouvoir être expliqué complètement sans quoi il ne répond pas au règlement décrit dans le ROI IRAM.

Le test d'évaluation finale, à cours ouvert, se fait sans aucun recours à l'IA.