

|  |
| --- |
| **IDENTIFICATION** |
| Intitulé de l’Unité d’Enseignement :  Programmation orientée objet | Niveau d’études : INF A ISA D  ISA C  |
| Intitulé du cours :  | Nombre de crédits ECTS :8 |
| Nombre de périodes : 120 | N°UE :1100Code de référence : **7525 21 U32 D2** |
|  |  |
| **DESCRIPTION** |
| **Prérequis :**Algorithme et programmation dans un langage proche du C# (C, Java) |
| **Documents de référence pour une préparation préalable au cours :** Néant |
| **Objectifs :** Pour atteindre le **seuil de réussite** :En disposant d’une structure informatique matérielle et logicielle opérationnelle, d’une documentation appropriée, les consignes de réalisation de l’application lui étant précisées, l’étudiant sera capable de : - concevoir, installer et utiliser des objets appropriés à la solution- concevoir et mettre en œuvre une procédure de test partiel et intégré ;- Justifier sa méthode de résolution ainsi que ses choix conceptuels et méthodologiques**Pour la détermination du degré de maîtrise**, il sera tenu compte* de la rigueur et du respect des spécificités du langage de programmation utilisé ;
* du style de programmation ;
* de l’adéquation et de la pertinence de la solution développée ;
* du respect du temps alloué ;
* de la clarté et de la précision dans l’utilisation du vocabulaire technique ;
* du degré d’autonomie atteint.
 |
| **Contenu du cours :**[1. Introduction 4](#_Toc126647429)[2. Installation de Visual Studio 4](#_Toc126647430)[3. Installation de SQL Server et SQL Server Management Tools 5](#_Toc126647431)[4. Premier aperçu d’un projet en C# WPF 5](#_Toc126647432)[5. Le modèle Objet – Classe (Class) et objet de classe (Object) 10](#_Toc126647433)[6. Cahier des charges de la classe User 12](#_Toc126647434)[7. Création d’une classe User 14](#_Toc126647435)[8. Champs de classe 15](#_Toc126647436)[9. Instanciation et constructeur d’objet 16](#_Toc126647437)[10. Encapsulation (Wrapping) 20](#_Toc126647438)[10.1. Propriétés, accesseurs get et set 21](#_Toc126647439)[10.2. Exercices 27](#_Toc126647440)[10.3. Le modificateur d’accès private 28](#_Toc126647441)[10.3.1. Lecture seule private set 28](#_Toc126647442)[10.3.2. Ecriture seule private get 32](#_Toc126647443)[10.3.3. Propriété private 35](#_Toc126647444)[10.3.4. Méthodes private 37](#_Toc126647445)[10.4. Propriété automatique (Auto implemented property) 38](#_Toc126647446)[11. Membres static 39](#_Toc126647447)[11.1. Méthode statique 39](#_Toc126647448)[11.2. Champ et propriété statiques 40](#_Toc126647449)[12. L’héritage 42](#_Toc126647450)[12.1. Etude de cas - la société Solutech 42](#_Toc126647451)[12.2. Définition et application de l’héritage 43](#_Toc126647452)[12.3. Le diagramme de classes et concepteur de classes. 46](#_Toc126647453)[12.4. Constructeurs de classes dérivées - base 50](#_Toc126647454)[12.5. Le modificateur d’accès protected 54](#_Toc126647455)[13. Les collections d’objets 56](#_Toc126647456)[13.1. Collections génériques 57](#_Toc126647457)[13.2. Utilisation de la List pour la lecture et écriture de fichier texte 58](#_Toc126647458)[13.3. Exercice - Affichage d’une collection dans une listView 59](#_Toc126647459)[13.4. Listes et lambdas 64](#_Toc126647460)[14. Polymorphisme 66](#_Toc126647461)[14.1. virtual , masquage new, remplacement override 67](#_Toc126647462)[15. Les Interfaces 72](#_Toc126647463)[15.1. Les classes abstraites – abstract class 72](#_Toc126647464)[15.2. Méthodes abstraites 73](#_Toc126647465)[15.3. Interface - interface 75](#_Toc126647466)[16. Binding 85](#_Toc126647467)[16.1. Introduction 85](#_Toc126647468)[16.2. Configuration pour le Binding 88](#_Toc126647469)[16.3. Design Pattern MVVM - Model, View, ViewModel 94](#_Toc126647470)[16.4. Exercices sur le Binding 96](#_Toc126647471)[17. Les délégués - delegate 102](#_Toc126647472)[17.1. Utilisation d’un délégué pour la société Solutech 104](#_Toc126647473)[17.2. Exercices sur les délégués 109](#_Toc126647474)[18. Les événements - event 110](#_Toc126647475)[18.1. Délégués et méthodes cibles multiples 111](#_Toc126647476)[18.2. Du délégué à méthodes cibles multiples à l’événement 113](#_Toc126647477)[18.3. L’événement : mécanisme et implémentation 117](#_Toc126647478)[18.3.1. Délégué générique EventHandler<T> 118](#_Toc126647479)[18.4. Exercices sur les événements 120](#_Toc126647480)[19. Accès à une base de données 122](#_Toc126647481)[19.1. ADO.Net 123](#_Toc126647482)[19.1.1. Création de la base de données SQL Server 123](#_Toc126647483)[19.1.2. Implémentation d’une classe utilisant ADO.Net 124](#_Toc126647484)[19.1.3. Requête sur tables avec relations 132](#_Toc126647485)[20. Tests unitaires 135](#_Toc126647486)[20.1. Exercices 139](#_Toc126647487)[21. Les threads 140](#_Toc126647488)[21.1. Utilisation des threads et communication TCP/IP, création d’une interface C# pour le contrôle d’Arduinos. 142](#_Toc126647489)[21.1.1. Exercices proposés 150](#_Toc126647490)[21.2. Supervision et communication Modbus TCP 152](#_Toc126647491)[22. Annexes 155](#_Toc126647492)[22.1. Body Expression 155](#_Toc126647493)[22.2. Délégués génériques Func<> et Action<> 155](#_Toc126647494) |
| **Bibliographie :**Syllabus de cours : Programmation orientée objet (V. Leblond)**Internet** regorge d’informations et de tutoriels sur la POO en C#- Tim Corey sur youtube **bibliographie :** Les versions de Visual Studio évoluant d’année en année, il est difficile de recommander une référence qui soit longtemps à jour.Citons les ouvrages de Jérôme Hugon, Thierry Douchet sur certaines versions de C# et Visual Studio. |
|  |
| **PERSONNEL** **ENSEIGNANT** |
| Mr LEBLOND V. |
|  |
| **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 les employés d’une société, 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 NetFramework WPF avec une orientation Windows. 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 le 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 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 le 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 (ModBus), des interactions avec des SGDB et même un peu de développement pour le mobile seront abordés. Les étudiants sont invités à développer un projet personnel ou par équipe, sur un sujet propre à leur section, réinvestiguant toutes les notions vues au cours. |
|  |
| **MODES** **D’EVALUATION** |
| * Epreuve écrite
* Projet avec dossier à déposer et présentation orale devant la classe.
 |