IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE « Advanced C++ programming » - BE « Bestioles » – 20/12/19

Nom du groupe : **Date :**

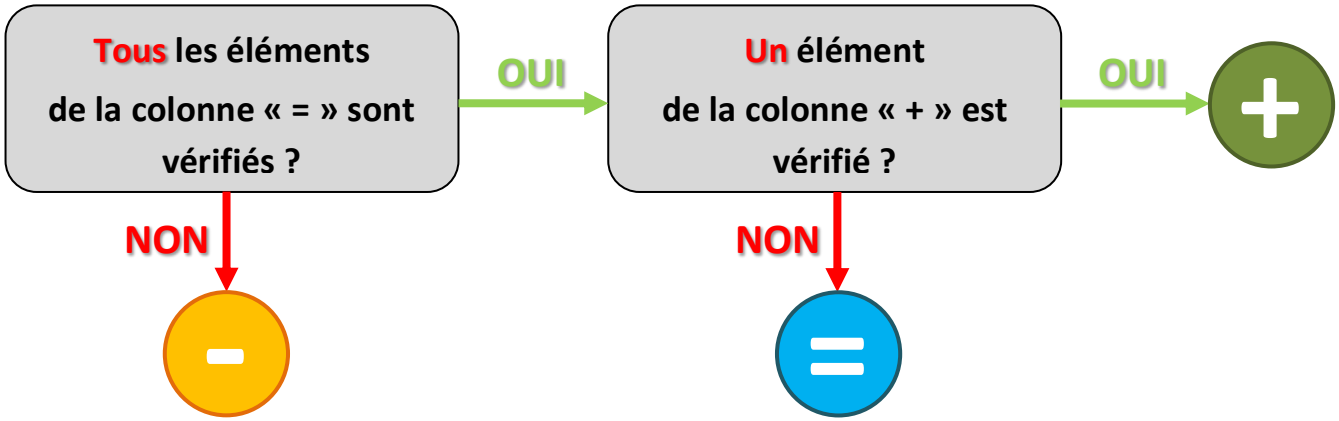
Noms des élèves :

Commentaires généraux :

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE « Advanced C++ programming » - BE « Bestioles » – 20/12/19

Algorithme d'attribution d'un niveau pour chaque critère de la grille :



Critères	Niveaux		
	-	=	+
1 – Les livrables respectent-ils les consignes données ?	Livraison partielle des livrables ou bien certains livrables ne respectent pas au moins une consigne donnée.	Tous les livrables sont présents et respectent toutes les consignes données.	Les livrables sont accompagnés par un document pertinent sur leur déploiement et/ou utilisation.
2 – La présentation donne-t-elle une vision pertinente du travail réalisé par le groupe ?	Peu ou pas de prise de recul par rapport aux problématiques traitées dans le cadre du BE. Les différentes parties de la présentation sont déséquilibrées sans que ce déséquilibre soit pertinent ou justifié.	Les travaux sont présentés de manière pertinente, synthétique et équilibrée (notamment entre **modélisation**, **implémentation** et **intégration**). La répartition des fonctionnalités retenues est explicite et nominative.	Les travaux réalisés sont recontextualisés explicitement par rapport à la demande client. Les stratégies/techniques de développement en groupe et de gestion de projet sont explicitées.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE « Advanced C++ programming » - BE « Bestioles » – 20/12/19

Critères	Niveaux		
	-	=	+
3 – Les réponses aux questions posées contribuent-elles à une meilleure compréhension des travaux ?	Les réponses aux questions posées restent partielles et peu convaincantes.	Les réponses aux questions posées sont pertinentes, complètes et **argumentées**, montrant une appropriation du sujet et des travaux réalisés.	Les réponses aux questions dépassent le cadre de la question en ouvrant explicitement des perspectives complémentaires d'usage ou de détails d'implémentation.
4 – Modélisation : Identification des fonctionnalités	Aucun ou peu d'effort pour aboutir à la décomposition du problème initial en une série de fonctionnalités. Les fonctionnalités présentées ne s'appuient pas ou partiellement sur les besoins du client.	Une ou plusieurs fonctionnalités de l'application sont clairement identifiées et décrites à l'aide d'un texte et de diagrammes UML. Chaque fonctionnalité retenue couvre l'intégralité des besoins du client s'y rapportant.	Au moins de 2 fonctionnalités ont été modélisées tant sur un plan statique que dynamique. Pour chaque fonctionnalité, la modélisation inclut du texte (explications), un diagramme de classes (architecture) et un ou plusieurs diagrammes de séquences (pour les aspects dynamiques).
5 – Modélisation : Découpage en classes	Peu ou pas d'effort de modularité en termes de modélisation objet (présence de classes fourre-tout, couteau suisse, par exemple). Le rôle de chaque classe n'est pas clairement identifié et exprimé.	La modélisation objet proposée s'appuie sur des classes aux périmètres et interfaces bien définis. Chaque classe a de ce fait, un nombre très limité de rôles qui sont explicités.	La modélisation proposée a pris en compte l'ajout potentiel de « fonctionnalités ». Cet effort est clairement expliqué et justifié. Le mode opératoire envisagé de ces ajouts est décrit.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE « Advanced C++ programming » - BE « Bestioles » – 20/12/19

Critères	Niveaux		
	-	=	+
6 – Modélisation : Exploitation de design patterns	Aucun design pattern n'est explicitement mentionné dans la modélisation et cette absence n'est pas justifiée. L'usage éventuel de certains « design patterns » n'est pas justifié.	Certains « design patterns » sont utilisés de manière appropriée pour répondre à des problématiques clairement identifiées de modélisation. Ces choix sont présentés et justifiés de manière argumentée.	L'exploitation de design patterns est pleinement justifiée par la présence de diagrammes UML qui instancient ces design patterns dans le contexte de l'application (les diagrammes UML génériques ne sont pas suffisants). Un effort de **quantification** de la plus-value apportée par l'utilisation de design patterns apparaît dans la justification.
7 – Implémentation : Conformité avec la modélisation	Il existe des incohérences entre l'implémentation (le code fourni) et la modélisation proposée (les diagrammes UML, entre autres). L'implémentation présente des écarts par rapport à la modélisation et aucune explication convaincante n'est apportée.	L'implémentation est en accord avec la modélisation. Si des écarts existent, ils sont explicitement discutés et justifiés.	Certains choix d'implémentation tirent profit de caractéristiques proposées par le langage C++ (utilisation de la STL, de templates, de smart pointers, par exemple) ET ces choix sont explicitement présentés et argumentés.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE « Advanced C++ programming » - BE « Bestioles » – 20/12/19

Critères	Niveaux		
	-	=	+
8 - Implémentation : Les sources respectent les bonnes pratiques demandées ?	Non-respect de l'organisation standard (tout le source est dans un même fichier, pas de makefile, par exemple)	L'organisation des sources du programme est conforme aux bonnes pratiques demandées (1 classe = 1 fichier .h et 1 fichier .cpp avec le contenu approprié, programme principal dans un fichier séparé, un fichier makefile pour assurer la production de l'exécutable). Les fichiers sources sont **indentés** et **commentés** (a minima).	Les choix d'implémentation en C++ sont justifiés (encapsulation, utilisation de template, de pointeurs, …) **directement dans les commentaires** associés au code source. Les commentaires placés au fil du code permettent de comprendre très simplement la logique de l'implémentation.
9 - Implémentation : Choix d'implémentation	Aucun choix d'implémentation n'est présenté ni argumenté, ni justifié.	Certains choix d'implémentation sont indiqués et justifiés de manière pertinents	Les choix d'implémentation retenus sont argumentés d'un point quantitatif (performances, maintenabilité, …)
10 – Validation : Tests dédiés à chaque fonctionnalité	Pas de tests mentionnés explicitement. Peu ou pas de retours sur les résultats issus de l'exécution d'éventuels tests.	Des tests sont implémentés et réalisés pour vérifier le fonctionnement nominal de chaque fonctionnalité. Une synthèse des résultats est proposée.	La définition des tests mis en œuvre est justifiée et argumentée. Une réflexion sur la couverture des tests mis en œuvre par rapport aux fonctionnalités demandées est présente.
11 – Intégration : Les fichiers sources fournis permettent-ils d'obtenir un exécutable pertinent ?	Échec de la compilation, production d'un exécutable qui se termine de façon anormale, exécutable dont l'exécution ne répond pas aux exigences client.	La compilation aboutit à la production d'un exécutable fonctionnel qui permet la simulation de l'écosystème conformément aux exigences client.	L'exécutable fourni permet la configuration de plusieurs simulations ainsi que l'obtention de résultats de simulation.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UE « Advanced C++ programming » - BE « Bestioles » – 20/12/19

Critères	Niveaux		
	-	=	+
12 – Validation : Tests de l'application complète	Pas de tests mentionnés explicitement. Peu ou pas de retours sur les résultats issus de l'exécution d'éventuels tests.	Au moins un scénario a été conçu et exécuté par l'application afin de vérifier que celle-ci fonctionne correctement par rapport aux besoins du client. Le choix de ce scénario est justifié et le résultat du test est analysé.	Un ensemble de scénarios a été conçu puis exécuté par l'application afin de vérifier son fonctionnement correct dans différentes configurations correspondantes aux besoins du client. Le résultat de ces tests est analysé et valorise la couverture du besoin du client par la réponse proposée.
13– Démonstration : adéquation avec les besoins du client ?	Pas de démonstration. La démonstration ne correspond pas aux fonctionnalités visées. La démonstration échoue de manière fréquente.	La démonstration illustre concrètement la ou les réponses apportées à chaque fonctionnalité visée, fonctionnalité qui correspond à un besoin du client.	Plusieurs exécutions dans des configurations différentes et pertinentes sont présentées. Des éléments complémentaires (synthèses de plusieurs exécutions par exemple) viennent étoffer de manière quantitative la couverture de la réponse apportée aux besoins du client.