LOG121 - Conception orientée objet : Laboratoires

Objectif des laboratoires

1. Mettre en pratique la conception orientée-objet dans le développement de logiciels pratiques
2. Mettre en pratique la documentation d'un code (avec Javadoc)
3. Développer sa capacité à communiquer et justifier son design
4. Respecter des échéanciers
5. Développer sa capacité à travailler en équipe (laboratoires 2 à 3).

Quelques points

- Vérifiez fréquemment vos courriels: interagissez, et rapidement.
- Évitez les échéanciers trop serrés: prévoyez de terminer le travail au moins deux jours avant la date limite.
- Nommer-vous un "chef" pour gérer la division et l'intégration du travail, faire l'échéancier, signaler les problèmes, etc.
- Signalez-nous les problèmes rapidement.

Remise des laboratoires

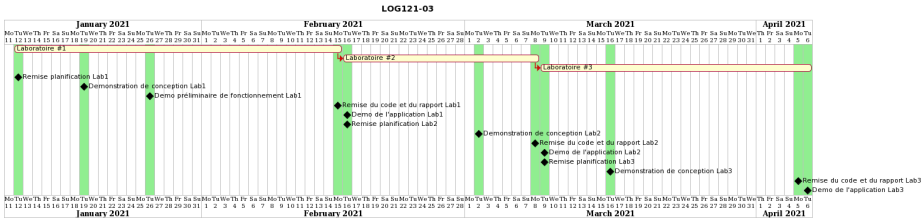

Figure 1: LOG121-03

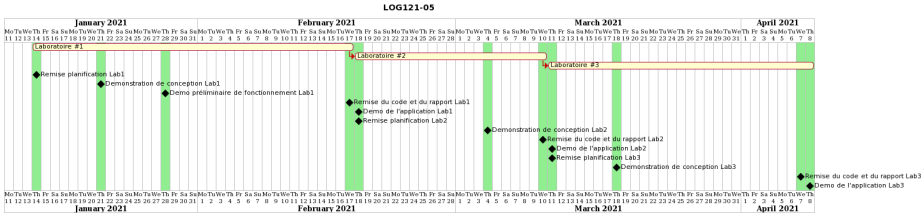

Figure 2: LOG121-05

Les laboratoires remis en retard ne seront pas corrigés et se verront automatiquement attribuer la note zéro.

Toutes les remises se font directement dans Github Classroom. Tout les rapports

doivent être au format markdown (.md) avec une version PDF généré. Tout les diagrammes doivent être réalisé à l'aide de PlanUML. Aucun ficher ne doit être remise par Moodle ou courriel.

Convertissez vos fichiers markdown en PDF avant la remise.

Évaluations

- Rapport (50 %) : le contenu exigé pour chaque laboratoire est indiqué dans son gabarit
- Code et fonctionnement (50 %) : attention documenter toutes les classes et les fonctions avec JavaDoc. N'oubliez pas les entêtes de fichiers. L
- Le fonctionnement est vérifié lors de la démo.

Différentes pénalités s'appliquent...

- Absence à une démo : 0 pour la partie fonctionnelle / membre absent Justification d'absence : contactez l'agente de gestion des études (lucie.caron@etsmtl.ca).
- Fautes d'ortographe: 0.5% / faute
- Retard dans la remise : automatiquement la note de 0.
- Professionnalisme: mauvaise nomenclature dans la remise, retard à une démo...

Planification du travail d'équipe

Les principales tâches (issues) doivent être réalisées dans Github project la première semaine de chaque laboratoire. Vous devez par la suite vous assurer que l'état de chaque tâche correspond à sont état dans le monde réel. Vous devez ajouter/gérer les sous-tâches que vous identifierez tout au long de votre laboratoire.

Énoncé des laboratoires

Lab	Objectifs	Durée	Pondération
1	Concevoir et implémenter une application en Java en appliquant les principes orientés objet et les patrons « observateur » et « stratégie »..Documenter des décisions de conception à l'aide d'UML.	15 heures	16%
2	Concevoir, implémenter et tester un cadriciel en Java. Appliquer les patrons « méthode template » et « stratégie ». Développer des tests unitaires.	9 heures	10%
3	Concevoir et implémenter selon le modèle MVC (modèle-vue-contrôleur) une application en Java.Application des patrons « commande », « mémento », « observateur », et d'autres.	12 heures	14%

Démonstration de conception

Une démonstration de conception est exigée lors de la deuxième séance de chaque laboratoire. Les documents de conception sont présentés directement au chargé de laboratoire, qui évaluera sur place équipe par équipe. Ainsi, aucun document n'est à remettre.

La présence à la démonstration est obligatoire : un membre absent ou en retard perdra ainsi tous les points relatifs à la démonstration, peu importe si cette dernière est réalisée par ses coéquipiers.

Le tableau ci-dessous résume ce qui est exigé pour chaque laboratoire.

Lab	Exigences
1	1. Choix et responsabilités des classes2. Diagramme de classe de l'application3. Diagramme de séquence : Par exemple un diagramme qui illustre la dynamique du patron observateur
2	1. Choix et responsabilités des classes2. Diagramme de classe de l'application3. Diagramme de séquence : Par exemple un diagramme qui illustre la dynamique du patron stratégie
3	1. Tableau d'application des patrons de conception2. Diagramme de classe de l'application3. Diagramme de séquence : Par exemple un diagramme qui illuste l'appel à l'exécution d'une commande4. Diagramme de séquence : Par exemple un diagramme qui illuste l'appel à défaire une commande (undo)

Documentation

- Java in Visual Studio Code
- PlantUml
- Mastering Markdown
- java
- Java Look and Feel Guidelines
- Tutoriel, programmation graphique avec Swing
- eclipse

Visual Studio Code plugin

- markdown all in One
- Plantuml
- Live Share Extension Pack

Commandes Utilitaires

pandoc -s -o S20211/laboratoires.pdf S20211/laboratoires.md