­

**Département de génie logiciel et des TI**

**Rapport de laboratoire**

|  |  |
| --- | --- |
| **No de laboratoire** | Laboratoire 1 |
| **Étudiant** | Martin Grégoire |
| **Code permanent** | GREM31019408 |
| **Cours** | LOG121 |
| **Session** | A17 |
| **Groupe** | 01 |
| **Professeur** | **Vincent Lacasse** |
| **Chargé de laboratoire** | **Dany Boisvert & Samir Djeffal** |
| **Date de remise** | 23 Octobre 2017 |

|  |
| --- |
| **Sommaire :**   * *Pour la remise, effacez toutes les directives et les remarques du gabarit* * *Adapter les titres des sous-sections (exemple : « titre de la décision d'implémentation »).* * *Attention à la concision, la syntaxe et l'orthographe.* * *Parlez au présent, privilégiez les formes impersonnelles et le « nous » au « je ».* |

# 1.Introduction (max 1/2 page)

1. Amenez brièvement le sujet.
2. Introduisez les objectifs globaux du logiciel à développer.
3. Annoncez à quel point la solution satisfait aux objectifs
4. Présentez les sections consécutives de votre rapport.

|  |
| --- |
| **Remarques :**   * *Évitez les détails, synthétisez.* * *N'indiquez pas d'énumération sous forme de points (comme celle-ci).* * *Ne répétez pas l'énoncé.* * *Privilégiez les objectifs spécifiques au présent laboratoire.* |

# 2. Conception

## 2.1 Choix et responsabilités des classes

Discutez du choix des classes et de la répartition des tâches. Vous pouvez utiliser un tableau qui résume vos classes importantes.

* *Responsabilités : utilisez des phrases descriptives, ne listez pas de méthodes*
* *Dépendances : les classes de votre application nécessaires au fonctionnement de la classe en question.*

|  |  |  |
| --- | --- | --- |
| * *Classe* | * *Responsabilités* | * *Dépendances* |
| * Environnement | * Classe Maitre, * S’occupe de la logique d’un tour, * interroge les usines pour Commencer ou finir la production * Fait bouger les elements * Fait la vente selon la strategie * Lit le fichier XML pour créer les batiments et chemins. | * Usine * IVenteStrategie * XMLHelper |
| * Unite | * Definit la position et le numero d’identification d’une unite (Batiment ou composante) | * Observable |
| * Batiment | * Definit le chemin de sortie d’un batiment (Entrepot ou usine) ainsi que la quantité qu’il peut contenir dans son inventaire | * Unite |
| * Composante | * Definit une vitesse de mouvement vers une usine. | * Unite |
| * IVenteStrategie | * Interface qui permet de définir des méthodes de ventes différentes. Deux ventes sont définies : VenteAléatoire et VenteIntervalles |  |
| * Usine | * Definit la quantite de composantes en entree necessaire pour continuer la production. Définit aussi la nombre de tours nécessaire pour produire une composante. * Elle observe entrepot pour | * Batiment * Composante * Chemin |
| * Chemin | * Definit la source et la destination d’une composante. Elle est affectée à une usine car l’usine connait la destination des composantes qu’elle produit |  |
| * FenêtrePrincipale | * Fenêtre principale de la Simulation. Elle contient des menus permettant de contrôler la simulation. | * MenuFenetre |
| * PanneauPrincipal | * Fait le dessin de la simulation | * Observer |
| * XMLHelper | * Lecture d’un fichier XML, est réutilisable | * Fonctions IO de Java |

## 2.2 Diagrammes des classes

Présentez votre conception logicielle complète appuyée d'un ou plusieurs diagrammes de classes UML. Ajouter un texte justificatif au besoin.

|  |
| --- |
| **Remarques :**   * *Vous pouvez réaliser le UML avec le logiciel de votre choix ou à la main si c'est fait avec beaucoup de soin (règle et écriture très claire). Si vous utilisez des logiciels qui génèrent automatiquement les UML à partir du code, révisez attentivement l'information, dont les types de relations (dépendance, association, agrégation, composition), lesquels sont souvent inappropriés.* * *N'indiquez pas systématiquement tous les attributs et les méthodes des classes. Privilégier l’information qui démontre le fonctionnement de l’application. Par exemple, certains accesseurs/mutateurs peuvent être implicites.* * *Attention à la résolution : s'il faut zoomer pour comprendre un texte ou un détail, ce dernier doit rester net.* * *Attention aux flèches coupées : si vous séparez votre UML en plusieurs figures, et que des flèches sont coupées entre les figures, indiquez la destination ou la provenance de la flèche coupée. Un exemple est ci-dessous. Il est cependant préférable de ne pas diviser le diagramme en plusieurs parties. Ainsi, vous pouvez le mettre dans un autre fichier auquel vous faite référence dans cette section.* |

## 2.3 Faiblesses de la conception

Décrivez les faiblesses de votre conception et des solutions possibles pour y remédier. (maximum 1 page)

|  |
| --- |
| **Remarques :**   * *Essayez d'identifier les problèmes les plus importants.* * *Il faut absolument trouver des problèmes et des améliorations possibles – inutile de mentionner que votre programme est bien fait ;-)* * *Vous pouvez par exemple discuter du couplage entre vos classes (citer des exemples de mauvais couplages) et discuter de problèmes de cohésion de classes en particulier (en indiquant clairement ce qui cause le manque de cohésion dans le choix des responsabilités). Soyez précis dans les références que vous faites! Par exemple, dire simplement qu’il y a trop de couplage et pas assez de cohésion est nettement insuffisant.* |

## 2.4 Diagramme de séquence (uml)

Chaque diagramme doit être appuyé d'un texte précis qui décrit verbalement le déroulement de l'exemple.

|  |
| --- |
| **Remarques :**   1. *Les noms des méthodes, des classes et des objets doivent correspondre à ceux utilisés dans votre code.* 2. *Erreurs fréquentes :*     1. *étiquette de l'objet incorrect. On doit avoir « nom:Type » pour un objet ayant un nom, «:Type» pour un objet anonyme et «Type» pour une méta-classe.*    2. *l'objet n'apparaît pas au moment de son instanciation*    3. *interaction anonyme*    4. *présence de classe abstraite (impossible à moins qu'utilisée en tant que méta-classe)*    5. *l'interaction ne reflète pas celle du code* |

### 2.4.1. Exemple qui illustre la dynamique du patron observateur

### 2.4.2. Autre diagramme de séquence

# 3 Décisions de conception / d'implémentation

Dans cette section, vous devez présenter **deux** décisions de conception et/ou d'implémentation. Vous pouvez prendre deux décisions du même type ou une de chaque type. Une décision peut également toucher à la fois la conception et l'implémentation.

Une décision de conception concerne le choix des responsabilités des classes, de leurs hiérarchies, de leur manière de se coordonner, à travers leurs couplages, pour effectuer une tâche précise requise pour les besoins du logiciel. Une décision de conception est normalement appuyée par un ou plusieurs diagrammes UML (diagramme de classes, diagramme de séquence).

Une décision d'implémentation concerne la programmation du logiciel conçu à la section précédente. Il s'agit donc de décisions liées aux algorithmes, aux structures de données et aux librairies utilisées. Elles sont généralement appuyées par des pseudo-codes et/ou des diagrammes UML qui présentent les attributs et méthodes impliqués dans le problème d'implémentation résolu.

|  |
| --- |
| **Remarques :**   * *Choisissez des décisions originales de votre implémentation.* * *Évitez les décisions implicites, telles que celles directement liées aux contraintes de développement. Par exemple, « décider » de respecter l'énoncé est implicite. Une décision pertinente comporte plusieurs possibilités concurrentes pouvant chacune respecter les contraintes de l'énoncé.* |

## 3.1 Décision 1 : titre de la décision

Décrivez, en subdivisant bien votre texte selon chaque aspect :

* **Contexte**: le problème résolu**.**
* **Solution 1**: votre solution d'implémentation 1, appuyée d’au moins un diagramme UML et/ou un pseudo-code. Discuter des compromis de la solution (avantages et inconvénients).
* **Solution 2** : votre solution d'implémentation 2, appuyée d’un diagramme UML et/ou d’un pseudo-code. Discuter des compromis de la solution (avantages et inconvénients).
* **Choix de la solution et justification :** Votre justification doit s’appuyer sur les compromis discutés précédemment. Les compromis se présentent en termes de performance, extensibilité et/ou ergonomie du logiciel, etc.

|  |
| --- |
| **Remarques :**   * *Il est impératif de respecter les consignes ci-dessus : laissez les titres (contexte, solution 1, solution 2, choix de la solution); 2 solutions distinctes concurrentes (et non deux aspects d’une même solution); au moins un diagramme UML et/ou un pseudo-code pour chaque solution.* * *Indiquez des références complètes. Par exemple, si vous citez une méthode, citez dans quelle classe (et même dans quelle librairie selon le cas) elle se trouve.* |

## 3.2 Décision 2 : titre de la décision

# 4 Conclusion

1. Faites un rappel des objectifs du logiciel: ces objectifs ont-ils été atteints?
2. Synthétisez ce qui a été réalisé, les points forts et les points faibles de votre conception.
3. Commentez, rappelez, les améliorations possibles, la portée de votre travail pour d'autres applications et les développements futurs envisageables.

|  |
| --- |
| * *Ne discutez pas du contexte académique et de détails se rapportant au travail d'équipe (si vous éprouvez un problème dans votre équipe, contactez plutôt un chargé de laboratoire).* |

# 5 Références

* Mettez, au besoin, les références