Projet Java IV 2022-2023
HELBElectro (Dépôt)

Introduction :

La société HELBElectro possède plusieurs entrepôts automatisés qui gèrent à la fois la fabrication et le stockage de produits électroniques. Il vous est demandé de développer le logiciel de gestion d'un entrepôt par un employé qualifié.

Afin de pouvoir réaliser le processus de fabrication des produits, l'entrepôt est approvisionné continuellement en composants électroniques de base. Ces composants permettent la fabrication de produits plus complexes. Il existe, pour l'instant, trois composants électroniques de base qui permettent d'assurer la fabrication des produits. Voici une table présentant les composants de base ainsi que les produits qui peuvent être fabriqués à partir de leur combinaison.

Combinaison	Fabrication
C1	Batterie
C2	Capteur de mouvement
C3	Moteur électrique
C1, C2	Alarme de sécurité
C1, C3	Voiture télécommandée
C2, C3	Robot suiveur
C1, C2, C3	Drone de surveillance

On peut voir que les trois composants de base sont, la batterie, le capteur de mouvement et le moteur électrique. En combinant, la batterie avec le capteur de mouvement, on peut fabriquer une alarme de sécurité.

Chaque type de produit a un temps de fabrication prédéfini. Comme l'entrepôt est automatisé, un produit d'un même type, par exemple, une alarme de sécurité, aura toujours une même durée de fabrication. Chaque type de produit a également un prix de vente, et un éco-score qui lui est propre. Voici une table qui résume les caractéristiques des différents produits :

Fabrication	Prix de vente	Éco-score	Durée de fabrication
Batterie	5	C	3
Capteur de mouvement	10	B	3
Moteur électrique	15	A	3
Alarme de sécurité	20	C	4
Voiture télécommandée	30	B	5
Robot suiveur	40	B	6
Drone de surveillance	60	E	12

Concernant le stockage, l'entrepôt dispose d'une zone de stockage pour les produits fini, ainsi qu'une zone de stockage pour les composants de fabrication. Ces

deux zones ont un nombre d'emplacements limité qui dépends de la configuration de l'entrepôt.

Au début, ces deux zones sont vides. Les composants sont alors livrés en continue à l'entrepôt et sont stockés dans la zone de stockage des composants. Quand des composants sont stockés dans cette zone, ils peuvent être combinés pour fabriquer des produits. Ces fabrications sont alors stockées dans la zone de stockage des produits, ce qui libère de l'espace dans la zone de stockage des composants.

Il peut arriver que la zone de stockage des composants soit pleine, dès lors, il faut attendre qu'un produit soit fabriqué avant de pouvoir y stocker d'autres composants. De la même façon, il se peut que la zone de stockage des produits arrive à saturation. Quand cela arrive, la totalité du stock de produits est rapatrié vers un autre entrepôt plus grand. Toute la chaine est dès lors à l'arrêt et les produits sont retirés manuellement par l'employé qualifié. Quand l'employé a terminé de vider le stock de produits, il l'indique au logiciel de gestion de l'entrepôt qui peut reprendre la fabrication et le stockage automatique des produits.

Quand des produits sont disponibles dans la zone de stockage, ceux-ci peuvent être vendu à des particuliers. L'employé peut alors libérer l'emplacement du produit acheté en le sélectionnant sur l'application, ce qui génère un ticket de caisse, sous forme d'un fichier . txt, qui peut être imprimé par l'employé et qui contient des informations plus détaillées sur les produits. Les produits ne peuvent être vendu qu'a l'unité (pas plus d'un produit par vente).

Les composants de base ont également certaines caractéristiques propres résumées dans ce tableau. Ces caractéristiques doivent être spécifiées sur le ticket de caisse lors de l'achat d'un produit.

Batterie	Les batteries ont un pourcentage de charge qui est un nombre compris entre 0 et 100.
Capteur de mouvement	Le capteur de mouvements à une portée spécifiée en mètres et une couleur qui peut être soit noir, jaune ou rouge.
Moteur électrique	Le moteur électrique une puissance, spécifiée en watts.

Les produits qui sont des combinaisons de ces composants possèdent les caractéristiques des composants ayants servi à leur fabrication.

Les emplacements de produits ont une position dans l'entrepôt (X, Y) où X et Y peuvent être, soit des numéros, soit des lettres, en fonction de l'entrepôt. Quelques exemples de positions : (0, 0), (5, 12), (A, C), (AB, E), (A, BE) etc... Afin de faire certaines statistiques sur l'entrepôt, l'employé devrait pouvoir visualiser, pour chaque emplacement de produits de l'entrepôt, le nombre total de produits qu'il a stocké et pour chaque type de produits, le nombre total de ventes réalisées à cet emplacement.

Lors de la fabrication des produits, l'employé peut décider d'optimiser certains paramètres du processus de fabrication. Il peut par exemple décider de fabriquer en priorité les produits qui optimisent le temps de fabrication. Cela veut dire, que l'entrepôt fabriquera en priorité, à partir des composants disponibles dans la zone des composants, les produits qui mettent le moins de temps à être fabriqués. L'employé peut choisir d'optimiser la fabrication par rapport à la durée de fabrication, par rapport à l'éco-score, ou par rapport au prix de vente. Il peut

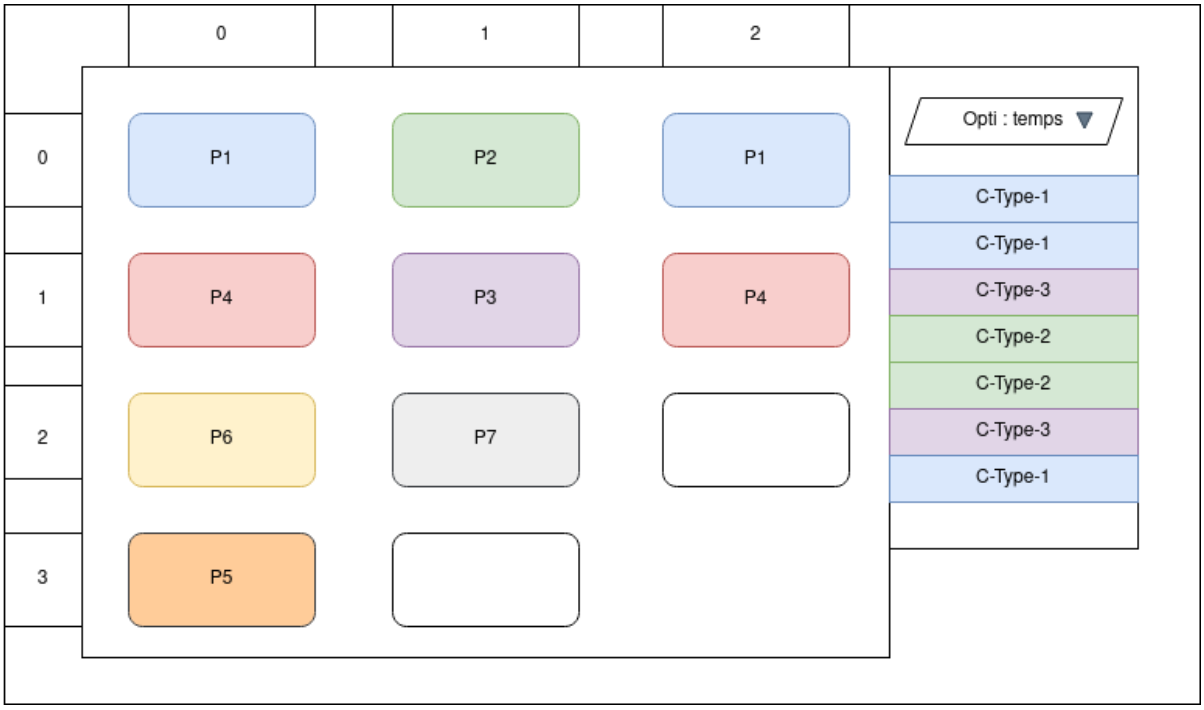
également choisir d'optimiser la fabrication, de façon à avoir le stock le plus diversifié possible en zone de stockage de produits.

Le concept de l'entreprise est encore jeune, donc il est fort possible que des choses changent à l'avenir. Par exemple il pourrait y avoir de nouveaux types de composants et de produits. Les caractéristiques des produits et composants pourraient devoir changer. D'autres possibilités d'optimisation de la chaine de fabrication pourraient apparaitre. La technologie de l'interface graphique pourrait aussi devoir changer. Etc. . .

Système visible par l'employé :

Afin de pouvoir travailler correctement, l'employé doit être capable sur un écran de monitoring de visualiser, en temps réel, les zones de stockage des produits ainsi que celles des composants.

Pour chaque emplacement l'employé peut voir, s'il est occupé ou non, et par quel type de produit ou composant. Il peut voir les caractéristiques d'un produit ou d'un composant en cliquant sur celui. Afin d'illustrer, voici un schéma de cette partie de l'application avec un formatage et une mise en forme minimale :



On peut voir sur cet exemple que la zone de stockage des produits (partie gauche) contient 11 emplacements, dont deux sont inoccupés. La zone de stockage des composants (partie droite) contient 8 emplacements, presque tous occupés, par les trois types de composants différents.

Le type 1 correspond à la batterie, son code couleur est le bleu. Le type 2 correspond au capteur de mouvements, son code couleur est le vert. Le type 3 correspond au moteur électrique, son code couleur est le mauve. Dans la zone des produits, chaque type de produit a sa propre couleur et les couleurs des composants de base correspondent à leur équivalent en zone de stockage des produits.
On voit donc que dans la zone de stockage des produits, le produit à l'emplacement (0, 0) est une batterie. Le produit à l'emplacement (0, 1) est un capteur de mouvements. Chaque type de produit a sa propre couleur. On voit donc que les emplacements (1, 0) et (1, 2) sont occupés par le même type de produit (P4).

On peut voir en haut à droite un emplacement de sélection qui permet de définir les paramètres d'optimisation de la chaine de production. Ici on choisit de produire, à partir des composants, en priorité, le produit qui met le moins de temps à être fabriqué.

A partir de cet écran, on peut également vendre un produit à un particulier. Pour cela, l'employé clique sur l'emplacement occupé par le produit à vendre, ce qui ouvre une nouvelle fenêtre « emplacement produit » dont voici également un schéma avec un formatage et une mise en forme minimale :

Supposons que ce samedi 1er Avril 2023 à 09:25:37, l'employé vende le produit P5 (Voiture télécommandée) à l'emplacement (3, 0), cela génère le ticket t_092537. txt. Voici à quoi ressemble ce ticket de caisse :

Date: 01/04/2023 - 09:25:37
Type produit: Voiture télécommandée
Prix: 30 euros
Eco-Score: B
Charge: 55%
Puissance: 30W

Simulation :

Afin de tester et simuler votre code en situation, un fichier avec des arrivées de composants est disponible sur eCampus. Le fichier est composé de plusieurs lignes et chaque ligne décrit l'information suivante : [Nombre de secondes, type de composant, [caractéristiques]]. Pour la première ligne, « Nombre de secondes » est le nombre de secondes depuis le début du lancement du programme. Pour toutes les lignes suivantes, c'est le nombre de secondes depuis l'arrivée dans la zone des composants du composant précédent. Ceci permet donc de simuler le flux d'arrivée des composants.

Pour illustrer, voici un exemple de fichier de simulation :

2, Batterie, 88%
6, Capteur, 33m, rouge

Ceci signifie qu'après 2 secondes une batterie dont la charge est de 88% est déposée dans la zone de stockage des composants. Ensuite, 6 secondes après l'arrivée de

cette batterie, un capteur de mouvement dont la portée est de 33m et la couleur rouge, est à son tour déposée dans la zone de stockage des composants.

Il peut arriver que la zone de stockage des composants soit pleine, si c'est le cas, les composants ne peuvent pas être déposés dans l'entrepôt, ils doivent alors être livrés ailleurs et sont donc ignorés.

Votre programme devra obligatoirement pouvoir être lancé avec le fichier de simulation.

Notes Importantes :

Les contraintes et fonctionnalités du projet sont susceptibles d'évoluer au cours du temps. Pensez donc à adapter une stratégie de développement adéquate.

Certains points de la description ne sont pas précisés ou sont laissés volontairement vagues. Il revient à vous de faire certains choix d'interprétations. Veillez toutefois à ce que votre approche soit logique et justifiée.

Contraintes de développement :

• Votre programme devra être compilable et exécutable dans un environnement Linux Ubuntu tel qu'une des machines virtuelles utilisées en cours et disponible sur le SharePoint d'eCampus. Un script bash nommé « run.sh » devra permettre la compilation et l'exécution de l'application en utilisant seulement la commande suivante «bash run.sh » en terminale. Si le fichier n'est pas fourni, ou si la compilation et l'exécution ne fonctionnent pas dans l'environnement spécifié, le projet ne sera pas corrigé et sanctionné d'un zéro. Testez donc que tout fonctionne avant la remise !

• Votre programme devra prendre en compte le fichier de simulation.

• Votre programme devra être développé en utilisant exclusivement la librairie graphique JavaFX.

• Votre code devra respecter les principes de designs orientés objet comme vu au cours. Pensez donc à faire des choix logiques de design de classes afin de produire un code propre et maintenable.

• A l'exception des commentaires, l'entièreté de votre programme devra être codé en anglais (nom de variables/fonction/classes/etc...).

• Votre code devra présenter une structure correcte et maintenable.

Notamment :

- o Evitez la duplication de code.
- o Evitez les constantes magiques.
- o Evitez le code mort.
- o Commentez intelligemment et suffisamment votre code
- o Commentez correctement vos variables et méthodes (ex : nom pertinents, pas de majuscules comme première lettre).

Un code dont la qualité sera jugée insuffisante sera sanctionné d'un zéro.
De manière générale, l'application devra faire un usage suffisant et correcte des concepts vus au cours.

Rapport :

Il vous est demandé de rédiger un rapport décrivant votre projet. Votre rapport devra contenir au minimum les sections suivantes :

Introduction : Cette section devra introduire votre projet. Décrire ce qui a été réalisé et présenter brièvement la structure de votre rapport.

Présentation de l'interface graphique : Cette section devra illustrer et expliquer la partie de votre application qui concerne l'interface graphique.

Analyse et Application des Design Patterns : Cette section devra expliquer la structure de votre implémentation en utilisant les outils d'analyses déjà vus durant votre parcours. Attention tous les diagrammes doivent être commentés. Vous devrez également dans cette section décrire les patterns utilisés et expliquer comment et pourquoi vous les avez utilisés. Les diagrammes d'analyse pourront vous aider à illustrer vos propos.

Limitations : Les limites de votre application, par exemple : dans quels cas d'utilisation votre application pourrait ne pas fonctionner comme prévu ? Y a-t-il des aspects techniques qui n'ont pas été traité ? Si vous aviez plus de temps pour le projet, qu'auriez-vous amélioré ? Plusieurs points de vue sont possibles, il revient à l'étudiant de choisir les points qu'il considère les plus pertinents pour réaliser son autocritique.

Conclusion : Votre conclusion sur le projet. Ce que vous avez réussi à faire ou non durant le projet et les apprentissages que vous en tirez.

Le rapport sera notamment évalué sur la qualité écrite et l'effort de présentation ainsi que la pertinence et la complétude des points abordés.

Deadline et remise :

La date limite pour la remise du projet est le **Mardi 30 Mai à 23h59**. Le projet devra être déposé sur eCampus à l'intérieur d'un fichier **. zip** (**et pas .rar**) contenant toutes les sources de votre projet ainsi que le rapport au format **P**DF (**et pas word**).

Développement et Triche :

• Tout acte de triche sera sanctionné par une **note de fraude au bulletin et sera notifié à la direction qui pourra possiblement décider de sanctions supplémentaires.** Des parties de code réutilisés d'un projet existant (d'un autre étudiant ou disponible sur le net) sans références dans votre rapport et sans mention de l'utilité du code utilisé est considéré comme une fraude.

• Pour ce projet, **vous ne pouvez pas reprendre des parties du code d'un autre étudiant** que ce soit de cette année ou d'une des années précédentes.

• Pour ce projet **vous pouvez vous inspirer/servir d'un code disponible sur internet**. N'oubliez pas toutefois d'en mentionner la source.

• Pour ce projet v**ous ne pouvez pas vous inspirer/servir d'un code généré par des outils de génération de code** tels que, par exemple, ChatGPT.

• Si vous avez un doute, contactez l'enseignant le plus tôt possible afin d'éviter du refactoring inutile, **un zero, ou pire, une note de fraude**.

Conseils pratiques :

Voici quelques conseils qui j'espère pourront vous aider.

- Veillez à ce que votre code ne contienne pas de constantes magique et/ou de duplication qui serait facilement évitable avec l'utilisation de méthodes.
- Veillez à effectivement implémenter les designs patterns quand cela est pertinent.
- Réfléchissez en terme d'« attribution de responsabilités » en accord avec les principes vu au cours (segmentation logique en classes).
- Ne négligez pas la théorie du cours (vous serez interrogés dessus).
- Ne négligez pas votre rapport. Tachez d'y expliquer/justifier explicitement vos choix d'implémentation. (Exemple : pourquoi un héritage ici ? pourquoi avoir choisi (ou non) d'implémenter ce design pattern ? quel avantage en termes de structure ? ...)
- Prenez le temps de bien comprendre tout l'énoncé avant de vous lancer (et lisez entièrement la FAQ).
- Vérifiez que votre code compile et run effectivement via le script bash prévu sur la machine Ubuntu présente dans le SharePoint du cours.

F**AQ :**

- **Puis je ajouter d'autres sections ou sous-sections dans le rapport ?**

Oui. La partie rapport de ce document donne seulement la structure minimum.

- **Puis je coder ou rendre mon rapport en anglais ?**

Oui. Pour ce qui est du code vous devez obligatoirement coder en anglais.

- **Puis je programmer sur Windows avec Eclipse ?**

Oui vous pouvez programmer comme vous le désirez mais vous devez respecter les contraintes de ce document, notamment : votre code doit être exécutable sur un environnement Linux Ubuntu via un script run. sh que vous devez fournir en même temps que vos sources (voir les contraintes de développement). Une machine préconfigurée sera disponible sur le SharePoint, celle-ci pourrait être utilisée pour la démo de votre application lors de la défense du projet.

- **Le rapport est-il important ?**

Oui. Le rapport est une pièce centrale de votre projet et c'est le premier outil de communication qui me servira à juger de la bonne réalisation du projet, pas seulement du point de vue du code mais également de la méthodologie utilisée. Il convient pour le rapport d'être pertinent et complet quant aux informations contenues.

- **Que voulez-vous dire par « tous les diagrammes doivent être commentés ».**

Les diagrammes doivent servir à illustrer et appuyer vos explications sur la structure de votre implémentation. Ils ne remplacent aucunement un texte explicatif revenant sur les points d'attention.

- **Je n'ai pas réussi à tout réaliser. Est-ce que ça vaut la peine de vous rendre le projet ?**

Oui, veillez toutefois à être claire sur les parties non implémentées. Il est très déconseiller de dissimuler ou d'« oublier » de mentionner qu'une partie n'a pas été réalisée. Veillez toutefois à bien respecter les consignes. Par exemple, votre code doit pouvoir compiler avec le script bash demandé, la qualité du code doit être suffisante, etc...

- **Puis je réaliser le projet en groupe ?**

Non, le projet doit être réalisé individuellement.

- **Que voulez-vous dire par « Votre code devra respecter les principes de design orienté objet comme vu au cours. »**

L'orienté objet fait intervenir certains principes comme l'héritage ou les méthodes statiques. Il revient à vous de décider quand les mettre en œuvre ou non. Votre approche devra toutefois être logique et justifiée. Cela implique notamment, de faire apparaitre de l'héritage quand cela a du sens, de rendre une classe abstraite quand cela a du sens, d'utiliser intelligemment l'encapsulation etc...

- **Dois-je vraiment faire de l'orienté objet ? Mon programme peut fonctionner sans.**

Vous devez absolument mettre en œuvre l'orienté objet pour ce projet. Cela fait partie des contraintes du projet. Un non-respect de ces contraintes sera pénalisé. L 'utilisation de l'orienté objet n'est pas une contrainte faible. Veillez donc à la respecter.

- **Doit-on faire de l'encapsulation ou de l'héritage et du polymorphisme ?**

Vous devez toujours en faire usage quand cela a du sens et ne pas le faire est généralement un signe de mauvaise conception.

- **Dois-je vraiment utiliser des design patterns ? Mon programme peut fonctionner sans.**

Vous devez absolument mettre en œuvre les design patterns pour ce projet. Cela fait partie des contraintes du projet. Un non-respect de ces contraintes sera pénalisé. L'utilisation des design patterns n'est pas une contrainte faible. Veillez donc à la respecter. Il ne s'agit toutefois pas d'utiliser les patterns sans réflexion quant à leur utilité dans le projet. Leur utilisation et choix ou non choix doit être pertinente et justifiée. Une partie non négligeable des points sera donnée pour l'implémentation adéquate des designs patterns comme vu au cours

- **L'aspect graphique et l'ergonomie sont-ils des critères importants ?**

Ces critères seront susceptibles d'être pris en compte dans l'évaluation mais sont nettement moins importants que l'implémentation des fonctionnalités et le respect des contraintes.
Dis grossièrement : Mieux vaut un programme moche mais avec toutes les fonctionnalités implémentées qu'un programme magnifique mais avec des fonctionnalités manquantes et un design logique (architecture du code) désastreuse.

- **Puis je utiliser un outil comme Scene Builder pour la création d'interfaces graphique ?**

Oui tant que le code généré fasse partie de la bibliothèque d'interfaces utilisateur JavaFX.

- **Puis je utiliser des designs patterns qui n'ont pas été vu au cours ?**

Oui mais vous devez les définir, justifier leur utilisation et fournir des sources. Vous devez dans tous les cas justifier leur utilisation.

- **L'interface graphique doit-elle être exactement celle présentée dans l'énoncé ?**

Les informations et fonctionnalités présentes dans les maquettes doivent apparaitre dans l'application mais vous êtes libres d'ajouter des informations ou fonctionnalités ainsi que de parfaire l'esthétique et l'ergonomie tant que cela ne rentre pas en contradiction avec l'énoncé de ce projet. .