Projet Java IV

2023

Une image contenant logo

Description générée automatiquement

Réalisé par Popadiuc Claudiu

Supervisé par Monsieur Riggio, Jonathan

Table des matières

[Introduction 3](#_Toc134661574)

[Présentation de l’interface graphique 4](#_Toc134661575)

[Analyse et applications des Design Patterns 5](#_Toc134661576)

[Limitations 6](#_Toc134661577)

[Conclusion 7](#_Toc134661578)

# **Introduction**

Dans le cadre du cours de JAVA Q IV, nous avons été chargés de développer un logiciel de gestion d'entrepôt. Ce logiciel est conçu pour gérer plusieurs entrepôts automatisés qui s'occupent à la fois de la fabrication et du stockage de produits électroniques, le tout supervisé par un employé qualifié.

Dans ce rapport, je vais commencer par présenter l'interface graphique de manière visuelle et explicative afin de faciliter la compréhension du fonctionnement de l'application.

Ensuite, j'introduirai l'analyse et les design patterns utilisés dans l'application, en expliquant la structure de ma mise en œuvre à l'aide d'outils d'analyse. Je détaillerai également les raisons pour lesquelles j'ai choisi certains design patterns spécifiques pour mon application.

Ensuite, j'aborderai les limitations de mon application et me poserai des questions telles que dans quels cas d'utilisation l'application pourrait-elle ne pas fonctionner comme prévu ? Y a-t-il des aspects techniques qui n'ont pas été pris en compte ? Si j'avais disposé de plus de temps, qu'aurais-je pu améliorer ?

Enfin, je conclurai ce rapport en résumant les principales conclusions et en soulignant les points clés du développement du logiciel de gestion d'entrepôt.

# **Présentation de l’interface graphique**

Une image contenant capture d’écran, Rectangle, carré, Caractère coloré

Description générée automatiquement

Permettez-moi de vous présenter l'interface graphique du logiciel de gestion d'entrepôt, qui offre une vue complète et intuitive de notre système de stockage et de production. Comme vous pouvez le constater, notre entrepôt est équipé de 11 emplacements dédiés au stockage des produits. Ces emplacements sont actuellement utilisés pour la production et le stockage de divers produits et composants.

Dans la partie gauche de l'interface, vous trouverez la zone de stockage des produits. Nous avons déjà enregistré plusieurs types de produits dans notre système. Par exemple, nous avons le P7, un drone de surveillance, le P6, un robot suiveur, le P1, une batterie, le P2, un capteur de mouvement, et le P3, un moteur électrique. Il est également possible d'introduire d'autres produits, tels que le P4, une alarme de sécurité de couleur rouge foncé, ou encore le P5, une voiture télécommandée de couleur orange.

Dans la partie droite de l'interface, vous pouvez observer la zone de stockage des composants. Par défaut, nous avons prévu 8 emplacements pour les composants nécessaires à la fabrication de nos produits. Actuellement, nous disposons d'un composant disponible appelé le "C-Type-2". Ce composant polyvalent nous permet de fabriquer différents produits, ce qui explique la présence des produits déjà fabriqués dans la partie gauche de l'interface. Cependant d’autre types de composants peuvent aussi être dans cette zone de composants tel que le "C-Type-1" et le "C-Type-3" qui sont la batterie et le moteur électrique

Cette visualisation détaillée de l'interface graphique nous offre une vue d'ensemble claire et précise de l'état actuel de notre entrepôt. Elle nous permet de mieux comprendre le flux de production et de suivre facilement le stockage des produits électroniques. En examinant attentivement les différentes zones de stockage et les produits déjà présents, nous pouvons prendre des décisions éclairées concernant la gestion de l'entrepôt et l'optimisation de nos processus de production.

Une image contenant capture d’écran, Rectangle, carré, Caractère coloré

Description générée automatiquement

Vous pouvez facilement personnaliser l'affichage des colonnes et des lignes en utilisant les boutons « Lettres » et « Nombres ».

Lorsque vous appuyez sur le bouton « Nombres », les lignes et les colonnes de l'interface se transforment en numéros, facilitant ainsi l'identification et la localisation des emplacements de stockage, le nom du bouton correspondant se transforme alors en « Lettres ». Cependant, si vous appuyez à nouveau sur le bouton, les lignes et les colonnes redeviennent des lettres, offrant une représentation plus conventionnelle de l'emplacement des produits dans l'entrepôt. Le bouton retrouve alors son libellé original, soit « Nombres ».

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Si la partie de gauche, la zone de stockage des produits, se remplit, nous recevons une alerte. Il suffit ensuite d'appuyer sur le bouton "OK" pour vider l'entrepôt en une seule fois.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une action similaire ce produit lorsque la zone des composants est pleine, à ce moment-là, en revanche la zone n’est pas vidée, mais les produits qui arrive sont tout simplement ignoré.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Si j’appuie sur l’optimisation, une liste d’optimisation se mets à disposition, je peux ensuite choisir l’optimisation de production que je souhaite, ainsi que mettre la production de produits en pause.

Une image contenant texte, capture d’écran

Description générée automatiquement

Voici comment l’affichage des informations se présente lorsque l’on appuie sur l’un des produits dans la zone de stockage. Si le produit est vide, l’alerte affichera simplement "Statut inoccupé ".

Une image contenant texte, capture d’écran, Police, Page web

Description générée automatiquement

Si un produit est présent, l'affichage de l'alerte sera différent. On y trouvera le statut "occupé", ainsi que tous les attributs nécessaires à sa création. De plus, deux actions seront disponibles : la possibilité de consulter les statistiques de l'emplacement du produit et la possibilité de vendre le produit.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant capture d’écran, Rectangle, carré, Caractère coloré

Description générée automatiquement

# **Analyse et applications des Design Patterns**

* **Factory** : Permet de centraliser la création de mes objets. J’utilise ce Design Pattern pour créer mes composants et mes produis.
* **Singleton** : Permet de créer une seule instance de la classe via la methode getInstance (). Je l’utilise dans la factory car.
* **MVC** : J’ai utilisé la classe **« HELBElectroView »** qui représente la vue, la classe « HELBElectroController » qui sert de Controller et les autres classes servent pour le modèle.
* **Créateur** : Qui détient la responsabilité de créer la vue de mon interface graphique avec tout ce qui se tient de la Grid, Vbox, Boutton, etc... La classe **« HELBElectroView »** a la responsabilité de création de tout cela car c’est cette classe qui s’occupe de l’affichage.
* Créateur : Qui détient la responsabilité de créer les boutons ? La classe « InterfaceGraphic » a la responsabilité de création des boutons car c’est cette classe qui s’occupe de l’affichage.
* Expert en information : principe d’attribuer la responsabilité aux objets. Le fait que les emplacements remplissent le parking. Le parking contient toutes les informations des emplacements donc c’est logique qu’elles soient créées dans celui-ci (ControllerParking).
* Principe de faible couplage : si on effectue une modification, cela ne devrait pas impacter d’autres classes. Il faut réduire au minimum le nombre de modification à effectuer.
* MVC : j’ai utilisé la classe « IntrefaceGraphic » qui représente la vue, la classe « ControllerParking » qui sert de Controller et les autres classes servent pour le modèle.
* Contrôleur : la classe « ControllerParking ». Elle représente le système dans sa globalité. Par exemple, le fait que lorsqu’on ajoute un véhicule et que l’emplacement change de couleur, cela crée des changements dans les données.
* Singleton : permet de créer une seule instance de classe.
* Fabrique : permet de centraliser la création de mes objets. J’utilise ce pattern pour créer mes véhicules. S’il y a d’autres objets à créer, c’est dans cette classe que ça se passe.
* Stratégie : le comportement des prix varie selon les jours. Ils évoluent donc dynamiquement à l’exécution.

# **Limitations**

# **Conclusion**