**Projet Intégration : le prix, le rabais, la taxe et le paiement.**

Le projet est à réaliser par équipe de 2. L’objectif du projet est de vous faire travailler sur l’organisation de votre code objet en intégrant des modules existants.

Le problème est courant : vous avez une liste de produits en inventaire. Un consommateur vient faire ses courses. Suite à son choix, vous devez calculer le prix final pour sa transaction. Cela inclut l’application de rabais et des taxes.

**Intégration du code:**

Tout d’abord vous devez intégrer les différents composants que vous avez déjà programmés dans un unique projet :

1. Le monnayeur du TP2 incluant les tests unitaires,
2. L’interface graphique développée du TP3,
3. La couche de stockage développée dans le TP3.

Vous pouvez choisir les composants d’un membre ou l’autre de l’équipe selon ses qualités.

**Nouvelles fonctionnalités: rabais**

Vous devez intégrer les types de rabais suivants. Chaque rabais doit pouvoir être ajouté ou supprimé pendant l’exécution et on doit pouvoir fixer la valeur des paramètres indiqués.

1. 2 pour 1 pour un produit donné en paramètre. On doit pouvoir avoir plusieurs produits en 2 pour 1. Exemple : un client achète 7 bouteilles d’eau en 2 pour 1, on peut donc appliquer 3 fois le 2 pour 1   
   (1 + 1) + (1 + 1) + (1 + 1) + 1  
   Alors le client paie le prix de la bouteille 4 fois.
2. seuil au delà duquel le client ne paie pas les taxes. Ce seuil doit pouvoir être fixé en paramètre.
3. un produit gratuit pour chaque tranche de *n*$ d’achat initial. La règle doit pouvoir être paramétrée pour changer le produit (qui est automatiquement ajouté à la liste avec un prix de 0) et le seuil *n*. Exemple : on veut donner un paquet de chips pour chaque tranche de 60$, un client qui achète pour 150$ dépasse 120$ il aura droit à 2 paquets.

**Nouvelles fonctionnalités: paiement**

Vous devez initialiser une caisse à moitié pleine. Le paiement d’une transaction doit maintenant donner lieu aux étapes suivantes :

1. La liste complétée le commis déclenche le paiement en appuyant sur Payer
2. Un dialogue apparait pour entrer les objets (billets et pièces) donné par le client.
3. Le monnayeur affiche la monnaie à rendre dans une fenêtre de dialogue qui inclut le total ainsi que les objets à remettre.
4. Le texte du ticket de caisse (items, total sans taxes, taxes, rabais accordé etc.) doit s’afficher dans LogCat. Un exemple de ticket de caisse est fourni.

Afin de pouvoir déboguer, ajouter une option dans votre menu qui permet de visualiser en tout temps l’état de la caisse (nombres de billets et pièces).

**Nouvelles fonctionnalités: multilingue**

Vous devez rendre votre application multilingue. Cela ne concerne pas les données des produits mais l’ensemble de l’interface graphique de la caisse.

**Pointage:**

**Code couleur : jaune = almost done , vert = done**

La liste suivante détaille tous les éléments d’évaluation incluant les points décrits ci-dessus.

|  |  |  |
| --- | --- | --- |
| *Élément de correction* | *Fonction* | *Tests* |
| Taxes : les taux de taxes **différents selon les produits** fonctionnent **(WC)** | 2 | 3 |
| Rabais : rabais **« 2 pour 1 »** possible **produit par produit (JLC)** | 2 | 3 |
| Rabais : pas de taxes au delà de **n** dollars (avant taxes) **(WC)** | 2 | 3 |
| Rabais : un produit gratuit par tranche de **m** dollars **(JLC)** | 2 | 3 |
| Paiement : les étapes décrites sont implantées **(WC)** | 4 |  |
| Paiement : une option pour voir l’état de la caisse est au menu **(WC)** |  |  |
| Paiement : le ticket de caisse complet est affiché **(JLC)** | 3 |  |
| Architecture : l’application utilise des couches de (Repository, Service …). **(JLC ET WC)** Toute autre approche devra être justifiée. | 4 | 4 |
| Validation : tous les champs de saisie sont **validées (WC)** | 3 |  |
| Codes barre : la caisse fonctionne avec un **lecteur** de code barres (Intent) (?) | 2 |  |
| I18N : votre logiciel est **bilingue français anglais (JLC)** | 2 | 2 |