1 Distributeur Automatique

Il s'agit de simuler un distributeur automatique type *Selecta*. Ce sera bien sûr une version simpliste.

Aucune interface utilisateur n'est demandée.

1.1 Fonctionnalités

Il sera possible de gérer plusieurs machines pendant l'exécution du programme.

Lors de la création d'une machine, il faut lui fournir la listes des articles à vendre. Chaque article est décrit par les informations suivantes:

- son nom (`name`)
- le code pour la sélection (`code`)
- la quantité initiale (`quantity`)
- le prix unitaire (`price`)

Ensuite le programmeur peut intéragir avec la machine en utilisant les actions suivantes:

- `Insert(amount)` Ajoute `amount` de crédit en vue d'une sélection.
- `Choose(code)=> message` Demande l'article correspondant à `code`, renvoie un `message` indiquant le résultat de l'opération.
- `GetChange()=> amount` Retourne le montant actuel de crédit de sélection.
- `GetBalance()=> amount` Retourne le montant total encaissé.

Chaque appel à `Insert` accumule le montant passé au crédit de sélection. Ce crédit est consultable en appellant `GetChange`. Voici les différents cas possible lors d'un appel à `Choose`:

- Il n'y a pas assez de crédit de sélection par rapport au prix de l'article choisi. Le message de retour est `"Not enough money!"`.
- L'article choisi est épuisé. Le message de retour est `"Item <name>: Out of stock!"` où `<name>` est le nom de l'article.
- Le code ne correspond à aucun article. Le message de retour est `"Invalid selection!"`.
- L'article choisi existe et le crédit de sélection est suffisant. L'article est délivré, la quantité d'article est mise à jour, le crédit de sélection également ainsi que le total encaissé. Le message de retour est `"Vending <name>"` où `<name>` est le nom de l'article.

1.2 Contraintes

Vous avez le choix du langage de programmation. Vous pouvez utilisez toutes les librairies standards de ce langage ainsi que d'autres librairies généralistes mais pas d'hypothétiques librairies gérant ce genre de problèmes.

La durée de ce travail est estimée à environ 4 heures. Evitez de passer trop de temps, vous pouvez soumettre une solution partielle. Vous présenterez votre code lors d'un entretien, vous aurez la possibilité d'en expliquer sa structure et son fonctionnement ainsi que les idées ou les pistes dans le cas où il n'est pas complet.

Vous soumettrez votre solution au moins deux jours avant la date de l'entretien soit en nous envoyant une archive ZIP (projet compilable et/ou exécutable) ou un lien vers un repository public.

1.3 Validation

Soit les `articles` suivants:

name	code	quantity	price
Smarlies	A01	10	1.60
Carampar	A02	5	0.60
Avril	A03	2	2.10
KokoKola	A04	1	2.95

Les assertions suivantes doivent être vérifiées (la machine est re-créée avec les articles ci-dessus avant chaque cas):

```
Insert(3.40)
Choose("A01") == "Vending Smarlies"
GetChange() == 1.80
```

```
Insert(2.10)
Choose("A03") == "Vending Avril"
GetChange() == 0.00
GetBalance() == 2.10
```

```
Choose("A01") == "Not enough money!"
```

```
Insert(1.00)
Choose("A01") == "Not enough money!"
GetChange() == 1.00
Choose("A02") == "Vending Carampar"
GetChange() == 0.40
```

```
Insert(1.00)
Choose("A05") == "Invalid selection!"
```

```
Insert(6.00)
Choose("A04") == "Vending KokoKola"
Choose("A04") == "Item KokoKola: Out of stock!"
GetChange() == 3.05
```

```
Insert(6.00)
Choose("A04") == "Vending KokoKola"
Insert(6.00)
Choose("A04") == "Item KokoKola: Out of stock!"
Choose("A01") == "Vending Smarlies"
Choose("A02") == "Vending Carampar"
Choose("A02") == "Vending Carampar"
GetChange() == 6.25
GetBalance() == 5.75
```

1.4 Extension

Vous produirez également l'extension suivante en impactant le moins possible la version initiale.

1.4.1 Fonctionnalité

Il s'agit d'ajouter l'horodatage des ventes, en vue d'établir des statistiques sur les ventes.

Bien que d'autres statistiques puissent être demandées dans le futur, celle qui nous intéresse ici est la suivante:
- les heures de la journée ayant le plus fort chiffre d'affaire.

Il y a donc 24 tranches d'heures possible. Le chiffre d'affaire entre 00:00 et 00:59, le chiffre d'affaire entre 01:00 et 01:50, etc…

Pour la validation, faites en sorte que la fonction `SetTime` permette de fixer la date/heure qui sera utilisée par votre code pour l'horodatage. Attention, en production, la fonction `SetTime` ne s'utilise pas, et votre code (sans modification) utilisera la vraie date/heure dans l'horodatage.

1.4.2 Validation

Afin de valider votre code, vous devez reproduire la situation décrite ci-après.

Créez une machine avec les `articles` suivants:

name	code	quantity	price
Smarlies	A01	100	1.60
Carampar	A02	50	0.60
Avril	A03	20	2.10
KokoKola	A04	10	2.95

Les opérations suivantes sont effectuées:

```
Insert(1000.00)
SetTime("2020-01-01T20:30:00"); Choose("A01")
SetTime("2020-03-01T23:30:00"); Choose("A01")
SetTime("2020-03-04T09:22:00"); Choose("A01")
SetTime("2020-04-01T23:00:00"); Choose("A01")
SetTime("2020-04-01T23:59:59"); Choose("A01")
SetTime("2020-04-04T09:12:00"); Choose("A01")
```

Votre code de validation affichera les **trois** meilleures tranches de la journée (heures) triées par ordre décroissant du chiffre d'affaire, ceci aura la forme suivante:

```
Hour 23 generated a revenue of 4.80
Hour  9 generated a revenue of 3.20
Hour 20 generated a revenue of 1.60
```