TD de *Génie Logiciel Avancé* n° 1
(Correction)

Types abstraits de données

Exercice 1 Un magasin grossiste de jouets commercialise 3 types de jouets : des billes, des puzzles et des poupées. Le prix d'un puzzle dépend de son nombre de pièces (0,5`EUR` la pièce), le prix d'une bille est de 1`EUR` l'unité, mais il y a une réduction de 2`EUR` par douzaine. Les poupées sont classées en trois catégories, petite taille, moyenne taille, grande taille, et leur prix est fixé en fonction de leur taille : 20`EUR` , 80`EUR` , 170`EUR` respectivement. Chaque client possède un nom et une catégorie. Il y a trois catégories différentes : A, B, C. Les clients de la première catégorie bénéficient d'une réduction de 5% sur les billes et de 3% sur les poupées. Les client de la catégorie B bénéficient d'une réduction de 2% sur les billes et de 7% sur les puzzles, enfin les clients de la troisième catégorie bénéficient d'une réduction de 10% sur les poupées.

1. Définir un type `taille` qui modélise les trois types de tailles de poupées.
   Correction :

```
{domaine}
taille
{constantes}
petite, moyenne, grande
```

2. Définir un type `jouet` qui modélise les types de jouets.
   Correction :

```
{domaine}
jouet
{constantes}
bille
{operations de construction}
cons-puzzle: {1<=n} -> jouet
cons-poupee: taille -> jouet
{operations de test}
est-bille: jouet  -> bool
est-puzzle: jouet -> bool
est-poupee: jouet -> bool
{operations d'acces}
nb-pieces: {n in jouet | est-puzzle(n))} ->  {1<=n}
taile-poupe: {n in jouet | est-poupee(n)} ->  taille
```

3. Définir un type `achat-modele` qui modélise l'achat d'une quantité déterminée d'un type de jouet.
   Correction :

```
{domaine}
achat-modele
{operations de construction}
cons-achat : jouet * { n in int | n >=1} -> achat-modele
{operations d'acces}
jouet-achat: achat-modele -> jouet
quantite-achat: achat-modele -> { n in int | n >=1}
```

4. Définir un type `categorie` qui modélise les trois différentes catégories d'un client.
   Correction :

```
{domaine}
categorie
{constantes}
a,b,c
```

5. Définir un type `client` qui modélise un client.
   Correction :

```
{domaine}
client
{operations de construction}
cons-client: chainecaracteres * categorie -> client
{operations d'acces}
nom-client: client  -> chainecaracteres
categorie-client: client  -> categorie
```

6. Définir un type `achat-client` qui modélise un client et tous ses achats.
   Correction :

```
{domaine}
liste-achats
{constante}
liste-vide
{operations de construction}
cons-liste: achat-modele * liste-achats -> liste-achats
{operations de test}
est-liste-vide: liste-achats -> bool
{operations d'acces}
premier-liste-achats: {n in liste-achats | non est-liste-vide(n)} -> achat-modele
reste-liste-achats: {n in liste-achats | non est-liste-vide(n)} -> liste-achats

{domaine}
achat-client
{operations de construction}
cons-achat-client: client * liste-achats -> achat-client
{operations d'acces}
client-achat: achat-client -> client
liste-achat: achat-client -> liste-achats
```

7. Spécifier une fonction `prix-billes` qui renvoie le prix d'une quantité de billes sans considérer les réductions dues aux différentes catégories de clients.
   Correction :

```
prix-billes(e) := soit d = e / 12 dans
                    soit rest = e - 12*d  dans
                       10 * d + rest;;
```

8. Spécifier une fonction `reduction` qui prend un jouet et une catégorie de client en entrée et calcule le coefficient qu'il faut appliquer aux prix de ce jouet en fonction de la catégorie donnée.

   Correction :

```
reduction(c,j) :=
    si est-bille(j)
       alors si c=a
             alors 0.95
             sinon si c=b
                   alors 0.98
                   sinon 1
       sinon si est-puzzle(j)
             alors si c=a
                   alors 1
                   sinon si c=b
                         alors 0.93
                         sinon 1
             sinon si c=a
                   alors 0.97
                   sinon si c=b
                         alors 1
                         sino 0.9
```

9. Spécifier une fonction `prix-modele` qui prend un objet de type `achat-modele` et une catégorie de client et renvoie un prix.

   Correction :

```
prix-modele(a,c) := soit j=jouet-achat(a) et e = quantite-achat(a) dans
    si est-billet(j)
    alors reduction(c,j) * prix-billes(e)
    sinon si est-puzzle(j)
          alors reduction(c,j) * 0.5 * n * e
          sinon soit t = taille-poupee(j) dans
                soit p = (si t=petite
                            alors 20
                            sinon si t=moyenne
                                  alors 80
                                  sinon 170) dans
                reduction(c,j) * p * e
```

10. Spécifier une fonction `prix-total` qui renvoie le prix de la liste de tous les achats d'un client.

    Correction :

```
rec prix-total(c,l) :=
    si est-liste-vide(l)
       alors 0
       sinon prix-modele(p, categorie-client(c)) + prix-total(c,r)
```