LOG1000 – Ingénierie logicielle

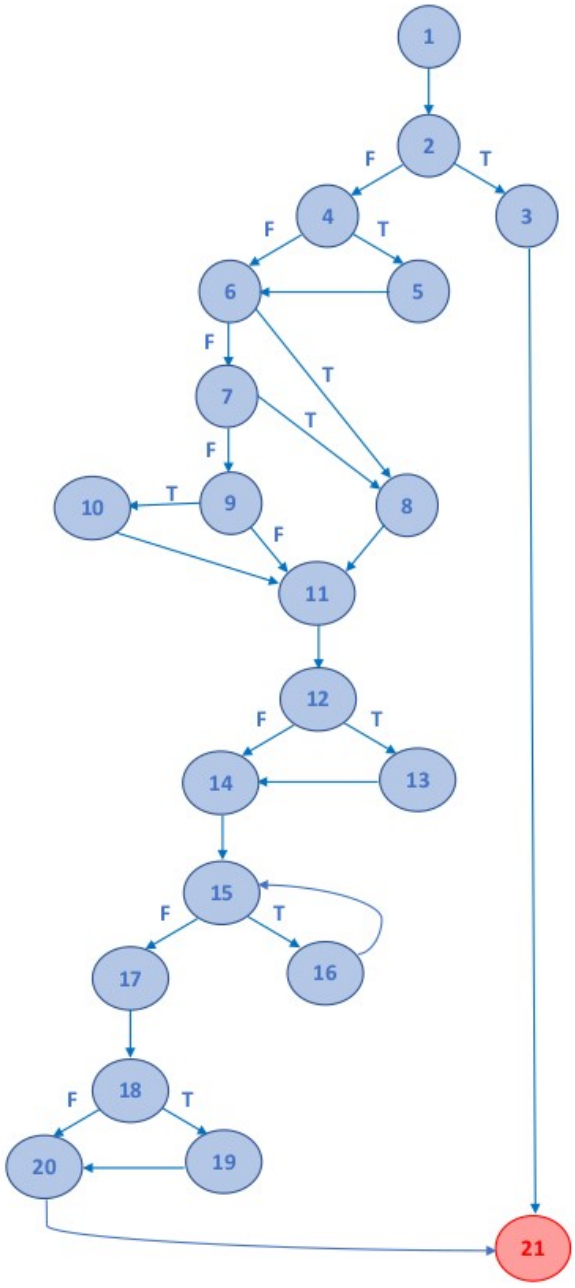
Travail pratique #4 - Introduction aux tests unitaires

Mélody Roy (1991902)
Pier-Luc Tanguay (1953707)

Groupe B1

1- Rédaction du graphe et flot de contrôle du code:

```
float Rabais::getRabais(Facture f, int code_client) {

1   float rabais = 0;
    Client* le_client = this->clients[code_client];

    // Les ID de plus de 25000 représentent les employéEs.
2   if (le_client->getID() > 25000) {
3       rabais = 0.15;
21      return rabais;
    }
    // Rabais basé sur l'âge
4   if (le_client->getAge() > 55)
5       rabais += 0.1;
    // Rabais basé sur la zone
6   if (le_client->getcodePostal().compare("H1C") == 0)
9       rabais += 0.03;
7   else if (le_client->getcodePostal().compare("H3P") == 0)
8       rabais += 0.03;
9   else if (le_client->getcodePostal().compare("J4O") == 0)
10      rabais += 0.02;

    // Rabais basé sur la date d'adhésion
    // Obtenir l'année courante.
11  std::time_t t = std::time(0);   // get time now
    std::tm* now = std::localtime(&t);
    int annee_courante = now->tm_year + 1900;
    // 1% de rabais par trois années, maximum 5%
    int rabais_adhesion = (annee_courante - le_client->getAdhesion().tm_year)/3;
12  if (rabais_adhesion > 10)
13      rabais_adhesion = 10;
14  rabais += float(rabais_adhesion) / 50;
    // Rabais basé sur le total de la facture
    float total = 0;

    for (int item=0 ; item<f.getSize() ; item++) {   15
16      total += f.getItem(item);
    }
17  int rabais_achats = total/120;
18  if (rabais_achats > 6)
19      rabais_achats = 6;
20  rabais += float(rabais_achats)/100;

    return rabais;   21
}
```



2- Rédaction des cas de tests pour la couverture des branches:

1) test_getrabais_employe:

Exigences à vérifier: SRS01, SRS02, SRS03
Raisonnement: On fournit un code client supérieur à 25 000, ce qui correspond à un employé qui devrait avoir accès à un rabais de 15%. De plus, on fournit un âge supérieur à 55 ans à cet employé pour vérifier qu'il n'a accès à aucun autre rabais.

2) test_getrabais_55ansPlus:

Exigences à vérifier: SRS04
Raisonnement: On fournit un âge supérieur à 55 ans ce qui correspond à un client qui devrait avoir accès à un rabais de 10%.

3) test_getrabais_zoneH1C, test_getrabais_zoneH3P, test_getrabais_zoneJ4:

Exigences à vérifier: SRS05
Raisonnement: On fournit une zone géographique précise à un client qui devrait avoir accès à un rabais spécifique soit 4% pour la zone H1C, 3% pour la zone H3P et 2% pour la zone J4O.

4) test_getrabais_adhesion:

Exigences à vérifier: SRS06
Raisonnement: On fournit une année d'adhésion entre 3 et 15 ans avant l'année actuelle. Ainsi, on obtient un rabais additionnel de 2% par 3 années complètes, mais inférieur au rabais maximum de 10%.

5) test_getrabais_adhesionMax:

Exigences à vérifier: SRS07
Raisonnement: On fournit une année d'adhésion supérieur à 15 ans avant l'année actuelle. Ainsi, on obtient le rabais additionnel maximum de 10% accordé pour l'année d'adhésion.

6) test_getrabais_facture:

Exigences à vérifier: SRS08
Raisonnement: On crée une facture supérieur à 120$ mais inférieur 720$. Ainsi, on obtient un rabais additionnel de 1% par tranche de 120$.

7) test_getrabais_factureMax:

Exigences à vérifier: SRS09
Raisonnement: On crée une facture supérieur à 720$ afin d'obtenir le rabais maximal accordé selon la facture qui est de 6%.

3- Recherche de défauts:

Suite aux tests effectués, nous avons repérés deux défauts dans la fonction Rabais::GetRabais(Facture,int):

1) le test RabaisTest::test_getrabais_zoneH1C échoue:

- Expected: 0.04
- Actual : 0.0299999993294477
- Delta : 0.001

Selon l'exigence SRS05, nous voudrions avoir un rabais de 4% lorsqu'on se situe dans la zone géographique H1C. Or ici, on voit que l'on obtient un rabais de 3%. Voici ce que nous suggérons pour remédier à ce problème:

```
97    // Rabais basé sur la zone
98    if (le_client->getcodePostal().compare("H1C") == 0) rabais += 0.03;    rabais += 0.04;
99    else if (le_client->getcodePostal().compare("H3P") == 0) rabais += 0.03;
100   else if (le_client->getcodePostal().compare("J4O") == 0) rabais += 0.02;
```

2) le test RabaisTest::test_getrabais_adhesionMax échoue:

- Expected: 0.1
- Actual : 0.200000002980232
- Delta : 0.01

Selon l'exigence SRS07, le rabais additionnel basé sur la date d'adhésion ne peut pas dépasser 10%. Or ici, on voit que l'on obtient un rabais maximum de 20%. Voici ce que nous suggérons pour remédier à ce problème :

```
108   // 1% de rabais par trois années, maximum 5%
109   int rabais_adhesion = (annee_courante - le_client->getAdhesion().tm_year)/3;
110   if (rabais_adhesion > 10) rabais_adhesion = 10;
111   rabais += float(rabais_adhesion) / 50;      float(rabais_adhesion)/100;
```

4- Rétroaction:

Question 1: Combien de temps avez-vous passé sur le travail pratique, en heures-personnes, en sachant que deux personnes travaillant pendant trois heures correspondent à six heures-personnes. Est-ce que l'effort demandé pour ce travail pratique est adéquat ?

Environ 6 heures-personnes. Nous trouvons cela adéquat puis qu'il s'agit des heures fournies en période de laboratoire.

Question 2: Avez-vous des recommandations à faire afin d'améliorer ce travail pratique ?

Voir plus d'exemples de cas de tests en période de cours.