POLYTECHNIQUE MONTRÉAL

Questionnaire
Contrôle Périodique 4

LOG3430

Sigle du cours

Identification de l'étudiant(e)		
Nom :	**Prénom :**	
Signature :	**Matricule :**	**Groupe :**

Sigle et titre du cours			*Groupe*	*Trimestre*
LOG3430 - Méthodes de test et de validation du logiciel			**Tous**	**20173**
Professeur			*Local*	*Téléphone*
Soumaya Medini				
Jour	*Date*	*Durée*		*Heures*
Mercredi	**1 Novembre 2017**	**1 heure**		

Documentation	*Calculatrice*	
☐ Aucune ☒ Toute ☒ Voir directives particulières	☐ Aucune ☒ Toutes ☐ Non programmable	**Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.**

Directives particulières
Toute documentation est permise, ainsi que les calculatrices, à l'exception toutefois des téléphones cellulaires et de tout dispositif capable de connexion Internet.

Important	
	Cet examen contient **2** exercices sur un total de **5** pages **(excluant cette page)** La pondération de cet examen est de **5** **%** Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux Vous devez remettre le questionnaire : ☒ oui ☐ non

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Exercice 1 – 14 points

Considérez le programme suivant :

```
public class Account {

    private float currentAmount;
    private float minimumBalanceAllowed;

(1) public Account(float currentAmount, float allowedOverdraft) {
      this.currentAmount = currentAmount;
      this.minimumBalanceAllowed = allowedOverdraft;
    }

(2) public float getMinimumBalanceAllowed() {
      return this.minimumBalanceAllowed;
    }

(3) public void setMinimumBalanceAllowed(float minimumBalanceAllowed) {
      this.minimumBalanceAllowed = minimumBalanceAllowed;
    }

(4) public float getCurrentAmount() {
      return this.currentAmount;
    }

(5) public void setCurrentAmount(float currentAmount) {
      this.currentAmount = currentAmount;
    }


(6) public boolean tryToProcessPayment(float anAmount) {
      if(this.getCurrentAmount()-anAmount>=this.getMinimumBalanceAllowed()){
          this.makePayment(anAmount);
          return true;
      } else {
          System.out.println("Not enough money :(");
          return false;
      }
    }

(7) protected void makePayment(float anAmount) {
      this.setCurrentAmount(this.getCurrentAmount() - anAmount);
    }
}


public class AccountPlus extends Account {

    private int cumulatedPoints;

(8) public AccountPlus(float currentAmount, float allowedOverdraft) {
      super(currentAmount, allowedOverdraft);
      this.setCumulatedPoints(0);
    }
```

```
(9)  public AccountPlus(float currentAmount, float allowedOverdraft,
                        int cumulatedPoints) {
       super(currentAmount, allowedOverdraft);
       this.setCumulatedPoints(cumulatedPoints);
     }

(10) public void setCumulatedPoints(final int cumulatedPoints) {
       this.cumulatedPoints = cumulatedPoints;
     }

(11) protected void makePayment(final float anAmount) {
       this.setCurrentAmount(this.getCurrentAmount() - anAmount);
       this.setCumulatedPoints(this.cumulatedPoints + Math.round(anAmount));
     }
}
```

1) Pour la classe **`Account`** complétez la tableau suivant pour la méthode MaDUM (ajouter des reporter si nécessaire). Utilisez les numéros devant les méthodes au lieu de leurs noms ; considérez aussi les abréviations suivantes : T : transformateur, C : constructeur, O : autre, R : reporter. (3 points)

	Account	getMinimumBalanceAllowed	setMinimumBalanceAllowed	getCurrentAmount	setCurrentAmount	tryToProcessPayment	makePayment
	1	**2**	**3**	**4**	**5**	**6**	**7**
currentAmount	C			R	T	T	T
minimumBalanceAllowed	C	R	T			O	

2) Identifiez toutes les tranches de la classe **`Account`** dans le tableau suivant : (2 points)

currentAmount	1	4	5	6	7
minimumBalanceAllowed	1	2	3	6	

3) Pour chaque tranche, donner les séquences de tests dans les tableaux suivants : (2 points)

Réponse :

Séquences de tests pour tranche **`minimumBalanceAllowed:`**

1	2	3	2	6	2

Séquences de tests pour tranche **`currentAmount`** :

1	4	5	4	6	4	7	4
1	4	5	4	7	4	6	4
1	4	6	4	5	4	7	4
1	4	6	4	7	4	5	4
1	4	7	4	5	4	6	4
1	4	7	4	6	4	5	4

4) Pour la classe **`AccountPlus`** complétez le tableau suivant pour la méthode MaDUM : (3 points)
Réponse :

getCumulatedPoints

	6	**8**	**9**	**10**	**11**	**12**
currentAmount	T	C	C		T	
minimumBalanceAllowed	O	C	C			
cumulatedPoints	T	C	C	T	T	R

5) Combien de séquences faut-il pour tester la tranche **`cumulatedPoints`**? Justifiez votre réponse : (2 points)

Réponse : Il faut 12 séquences. La classe **`AccountPlus`** a 2 constructeurs et 3 transformateurs => 2*3! = 12.

6) Faut-il changer des colonnes de la MaDUM de la question 1)? Justifiez votre réponse : (1 point)

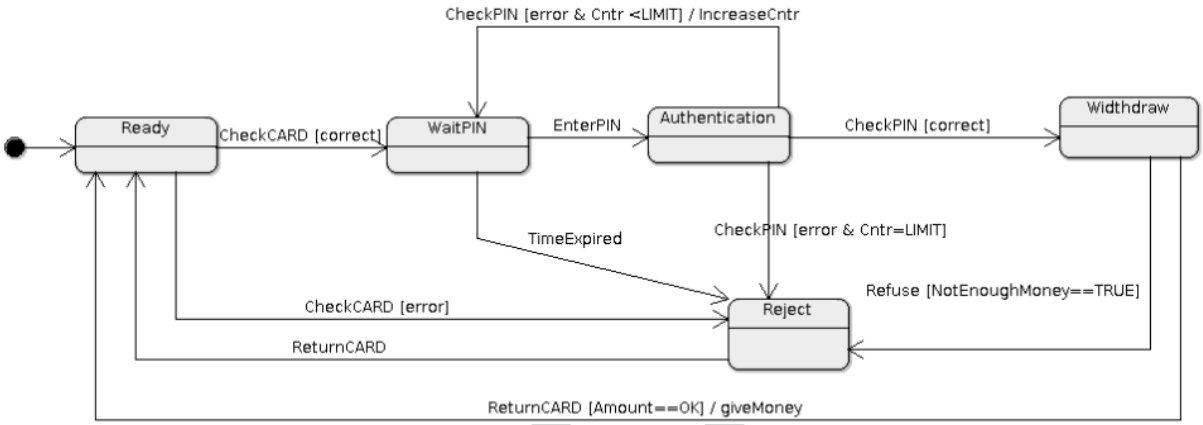
Réponse : Oui, la colonne 6 car 6 appelle 7 et 7 est redéfinie dans la classe **`AccountPlus`** – voir la méthode 11. Il faut donc ajouter T pour cumulatedPoints et 6.

7) Faut-il tester de nouveau des tranches pour la classe **`Account`**? Justifiez votre réponse : (1 point)

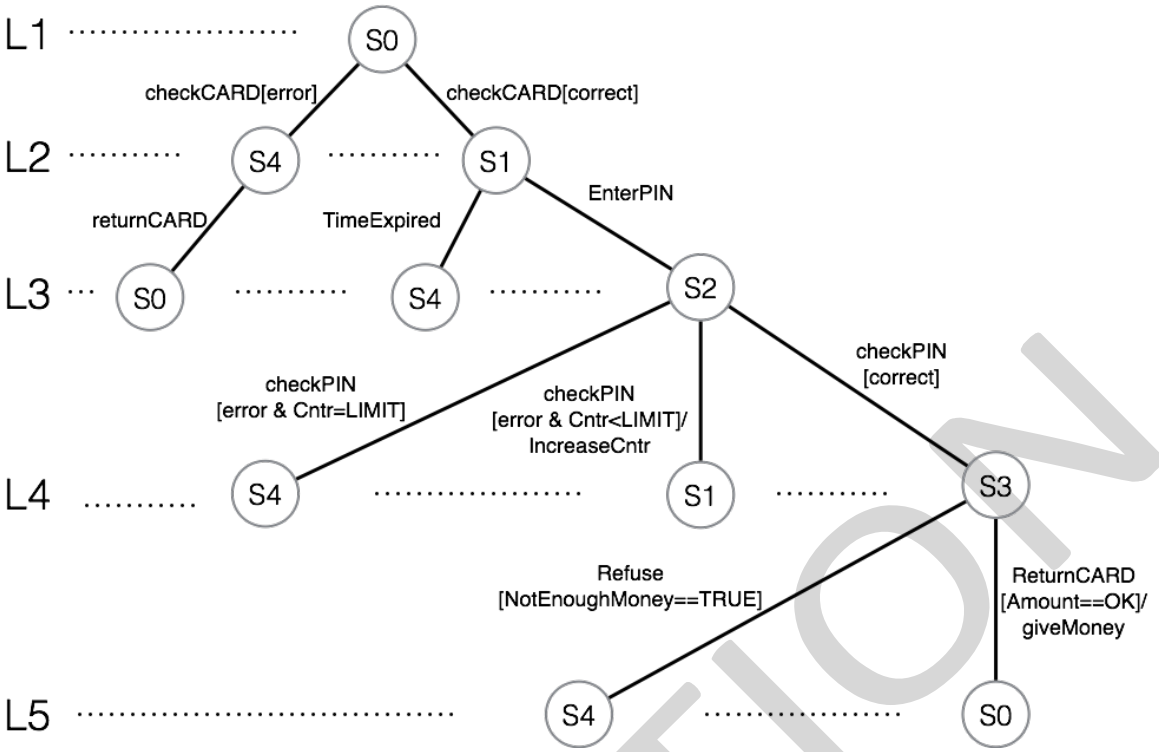
Réponse : Oui, il faut tester toutes les tranches dans le contexte de **`AccountPlus`** car toutes les tranches sont modifiées.

Exercice 2 - 6 points

Pour le diagramme suivant :



Q2.1) Dérivez l'arbre de transition selon la convention S0=Ready, S1=WaitPIN, S2=Authentication, S3=Widthdraw et S4=Reject : (3 points)



Q2.2) Dérivez les séquences de test. Pour chaque cas de test, il faut justifier et expliquer si des conditions spéciales sont nécessaires. (3 points)

Les cas de tests sont les séquences de la racine de l'arbre aux feuilles. Nous avons 6 séquences comme suit:

1) checkCARD, ReturnCARD
   condition de garde: [$error_{checkCARD}$]
2) checkCARD, TimeExpired
   condition de garde: [$correct_{checkCARD}$]
3) checkCARD, enterPIN, checkPIN (vers S4)
   condition de garde: [$correct_{checkCARD}$ & $error_{checkPIN}$ & Cntr==LIMIT]
4) checkCARD, enterPIN, checkPIN (vers S1)
   condition de garde: [$correct_{checkCARD}$ & $error_{checkPIN}$ & Cntr<LIMIT]
5) checkCARD, enterPIN, checkPIN, Refuse
   condition de garde: [$correct_{checkCARD}$ & $correct_{checkPIN}$ & NotEnoughMoney==TRUE]
6) checkCARD, enterPIN, checkPIN, ReturnCARD
   condition de garde: [$correct_{checkCARD}$ & $correct_{checkPIN}$ & Amount==OK]

Chaque cas de test dépend des conditions de gardes, notez en particulier que les séquences 3 et 4 sont identiques et seules les conditions de garde diffèrent.