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**	**20143**
Professeur		*Local*	*Téléphone*
Venera Arnaoudova		**B-411**	
Jour	*Date*	*Durée*	*Heures*
Mardi	**4 Novembre 2014**	**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 **6** pages **(excluant cette page)**

La pondération de cet examen est de **5** **%**

Vous devez répondre sur : ☒ le questionnaire ☐ le cahier ☐ les deux

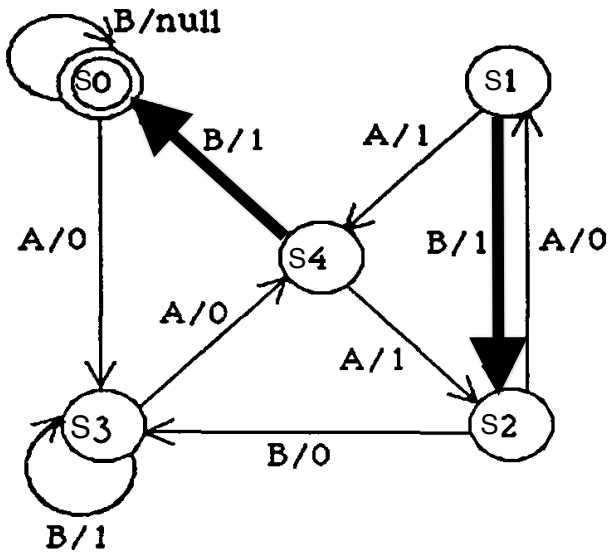
Vous devez remettre le questionnaire : ☒ oui ☐ non

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

SOLUTION

Exercice 1 – 8 points

Considérez le FSM suivant:



Proposez les séquences **les plus courtes** pour tester les transitions **S1-S2** et **S4-S0** (en gras) avec, si possible, chacune des méthodes suivantes :

1) TT (tour de transition). Ne pas se restreindre aux transitions en gras. (2 point)
2) DS (séquence distinctive). **Longueur maximale égale à 2**. (2 points)
3) UIO (entrée-sortie unique). **Longueur maximale égale à 2.** Ne pas réutiliser la séquence DS si possible. (2 points)
4) W (ensemble caractérisant). **Longueur maximale égale à 2.** Ne pas réutiliser la séquence DS si possible. (2 points)

Réponse :

	A	B	AB	BA	AA	BB
S0	0	null	01	null 0	00	null null
S1	1	1	11	10	11	10
S2	0	0	01	00	01	01
S3	0	1	01	10	01	11
S4	1	1	10	10	10	1 null

1) TT (tour de transition). (2 point)
   BABAAAAAABBAB -> null 010101101001

2) DS (séquence distinctive). (2 points)

BB est une séquence distinctive (voir le tableau ci dessus).

Tester S1-S2 :	A/0 A/0 A/1 A/0	B/1	B/0 B/1
Tester S4-S0 :	A/0 A/0	B/1	B/null B/null

3) UIO (entrée-sortie unique). Ne pas réutiliser la séquence DS si possible. (2 points)

UIO : S0 : B/null; S1 : AB/11; S2 : B/0; S3 : BB/11; S4 : AB/10

Tester S1-S2 :	A/0 A/0 A/1 A/0	B/1	B/0
Tester S4-S0 :	A/0 A/0	B/1	B/null

4) W (ensemble caractérisant). Ne pas réutiliser la séquence DS si possible. (2 points)

W = {B, AB}

S0 : {null, 01}; S1 : {1, 11}; S2 : {0, 01}; S3 : {1, 01}; S4 : {1, 10}

Tester S1-S2 :	A/0 A/0 A/1 A/0	B/1	B/0
	A/0 A/0 A/1 A/0	B/1	A/0 B/1
Tester S4-S0 :	A/0 A/0	B/1	B/null
	A/0 A/0	B/1	A/0 B/1

Exercice 2 – 12 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));
    }
}
```

SOLUTION

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 : (1 point)

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

SOLUTION

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 : (1 point)

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.

SOLUTION