POLYTECHNIQUE MONTRÉAL

Questionnaire
Contrôle Périodique4

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**	**20171**

Professeur	*Local*	*Téléphone*
Soumaya Medini	**C-624**	

Jour	*Date*	*Durée*	*Heures*
Mardi	**14 Mars 2017**	**1 heure**	

Documentation	*Calculatrice*	
☐ Aucune	☐ Aucune	
☒ Toute	☒ Toutes	
☒ Voir directives particulières	☐ Non programmable	

Directives particulières

Toute documentation est permise, ainsi que les calculatrices, tablettes et ordinateurs à l'exception toutefois de tout dispositif connecté à Internet.

Important

Cet examen contient **2** exercices et **9** questions 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 code suivant :

```
class Car{
   String brand;
   String color;
   double price;
   int power;

Car (){ } // 1

Car(String brand,String color, double price, int power){ //2
       brand= brand;
       color= color;
       price= price;
       power=power;

   }

   public String getBrand(){       return brand;    } //3
   public String getColor(){       return color;    }  //4
   public double getPrice(){       return price;    }  //5
   public int getPower(){       return power;    }  //6
   public void setPrice(double newPrice){           price = newPrice;    }  //7
}

Class UsedCar extends Car{
int status ;
int year ;
public int getStatus(){return status;} //8
public void setStatus(int n){ status=n;} //9
public int getYear(){return year;} //10
public void setYear(int n){ year=n;} // 11
public void addUsedCar(String t, String a, int p, int s, int  y){  //12
        brand=t;
        color=a;
        power=p;
        status=s;
   }

public void changePrice(double p){setPrice(p);} //13
public void showStatus(){ // 14
        if (status == 0)
        System.out.println("Like new\n");
        elseif (status == 1)
        System.out.println("Good condition\n");
        else
        System.out.println("Damaged car\n");

   }

}
```

Pour les classes Car et UsedCar il faut concevoir les tests unitaire.

1.1 (1 point) Selon le critère de Harrold McGregor quelles sont les méthodes de la classe Car qui doivent être testées de nouveaux ?

Toutes les méthodes doivent être re-testées :

1.1.1 Car(), Car(….)
1.1.2 getBrand()
1.1.3 getColor()
1.1.4 getPrice()
1.1.5 getPower()
1.1.6 setPrice

Justification : La méthode addUsedCar change brand, color, price and power. De plus, on assume qu'il y a une erreur et le programmeur a oublié d'affecter la valeur du paramètre year à y.

1.2 (1 point) Si on utilise le critère de Harrold McGregor quelles sont les méthodes de la classe UsedCar qui doivent être testées?

Réponse : Harrold McGregor n'a actuellement rien à faire ici ; C'est la première fois qu'on teste UsedBook donc toutes les méthodes doivent être testées.

1.3 (2 points) Si on veut tester les classes avec la méthode des tranches de données, spécifiez dans le tableau suivant combien de setter et getter il faut ajouter pour chaque classe :

Classe	Setter	Getter
Car	setBrand, setColor, setPower → 3	0
UsedCar	0	0

1.4 (2 points) Pour la classe *Car* identifiez les « tranches des données », ajouter des getters si nécessaire

Attributs									
brand	**Car()**	**Car(…)**	**setBrand**	**getBrand()**					
color	**Car()**	**Car(…)**	**setColor**	**getColor**					
price	**Car()**	**Car(…)**	**setPrice**	**getPrice**					
power	**Car()**	**Car(…)**	**setPower**	**getPower**					

1.5 (2 points) Pour la classe UsedCar identifiez les « tranches des données », ajouter des getters si nécessaire

Attributs									
status	**setStatus**	**getStatus**	**showStatus**	**addUsedCar**					
year	**setYear**	**getYear**	**addUsedCar**						

1.6 (4 points) Pour la classe UsedCar donnez le MADUM dans le tableau suivant (utilisez les numéros !):

	1,2	3	4	5	6	7	8	9	10	11	12	13	14
brand		r									t		
color	C		r								t		
price	C			r		t						t	
power	C				r						t		
status							r	t			t		o
year									r	t	t !		

Selon le MADUM quelles sont les méthodes de la classe *Car* qui doivent être testées de nouveaux ?

On remarque que les méthodes 12 et 13 transforme principalement les attributs de Car et encore une fois on va avoir à tester de nouveau toutes les tranches et ainsi toutes les méthodes.

Est ce que les reposes de test selon le MADUM et le critère Harrold-McGregor sont en désaccord ? Justifiez et expliquez la réponse ?

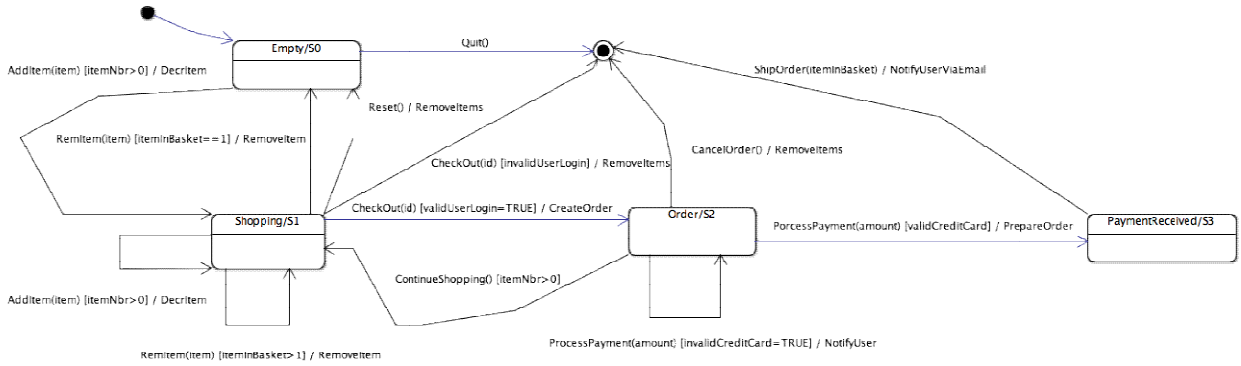
Comme prévus, la réponse est la même, on doit tester toutes les méthodes de nouveau.

1.7 (2 points) Pour la « tranches des données » **price**, donner les séquences de tests dans le tableau suivant :

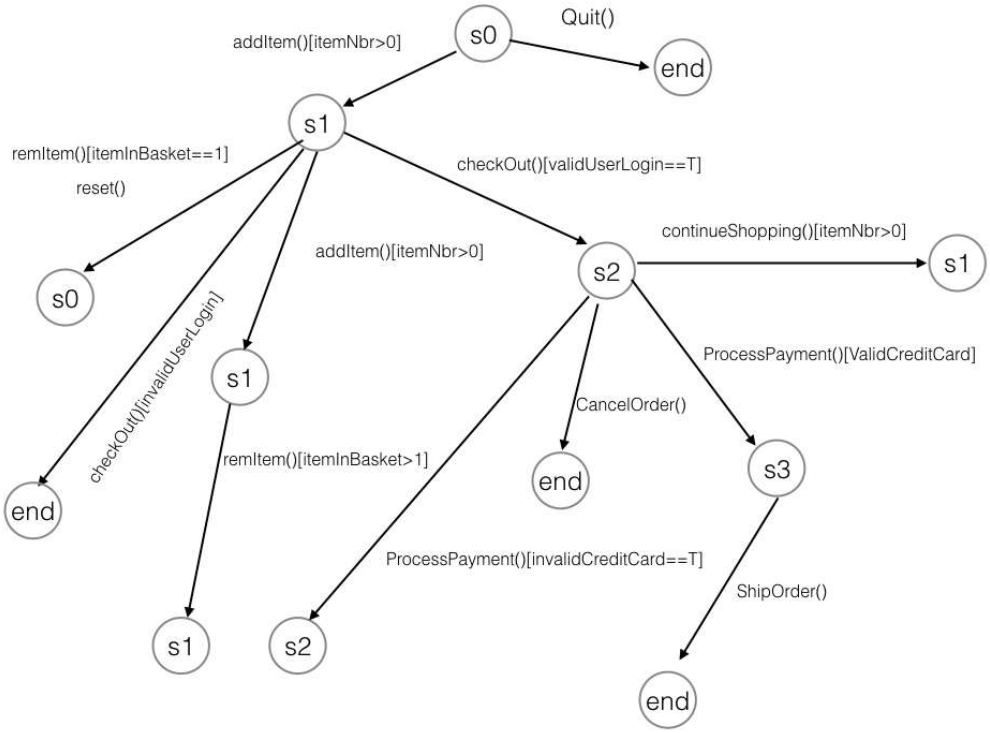
Séquences de test pour la Tranche **price**							
Car()	getPrice	setPrice	getPrice	changePrice	getPrice	showStatus	getPrice
Car(…)	getPrice	setPrice	getPrice	changePrice	getPrice		
Car()	getPrice	changePrice	getPrice	setPrice	getPrice		
Car(…)	getPrice	changePrice	getPrice	setPrice	getPrice		

Exercice 2 - 6 points

Pour le diagramme suivant **:**



Q2.1) Dérivez l'arbre de transition selon la convention S0=Empty, S1=Shopping, etc. (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)

S0 Quit () END

S0 addItem[ItemNBR>0] S1 reset S0

S0 addItem[ItemNBR>0] S1 remItem [itemInBasket==1] S0

S0 addItem[ItemNBR>0] S1 checkOut [invalidUserLogin] END

S0 addItem[ItemNBR>0] S1 addItem S1

S0 addItem[ItemNBR>0] S1 addItem [ItemNBR>0] S1remItem [itemInBasket>1] S1

S0 addItem[ItemNBR>0] S1 checkOut[validUserLogin==T] S2
ProcessPayment()[invalidCreditCard==T] S2

S0 addItem[ItemNBR>0] S1 checkOut[validUserLogin==T] S2 CancelOrder END

S0 addItem[ItemNBR>0] S1 checkOut[validUserLogin==T] S2 ProcessPayment()[validCreditCard==T] S3 ShipOrder END

S0 addItem[ItemNBR>0] S1 checkOut[validUserLogin==T] S2 ContinueShopping S1

On remarque que les conditions essentielles de garde sont signalées. Il est également important de noter que le 5ème cas de test peut être écrit comme suit :

S0 addItem [ItemNBR>0] S1 (addItem [ItemNBR>0] S1)+ remItem [itemInBasket>1] S1

Où nous ajoutons un nouvel élément au moins une fois. Les séquences ci-dessus peuvent être combinées pour obtenir un nombre plus faible de cas de test. Par exemple

S0 addItem [ItemNBR>0] S1 (addItem [ItemNBR>0] S1)+ remItem [itemInBasket>1]
S1vcheckOut[validUserLogin==T] S2 ContinueShopping S1 checkOut[validUserLogin==T] S2
ProcessPayment()[validCreditCard==T] S3 ShipOrder END

Ce cas de test associe 3 séquences de test simples à une seule séquence plus longue.