TE Génie Logiciel - GEN

Nom : Chatillon Basile

Date : 10 Avril 2017

1	2	3	4	5	6	7	8	9	10	11	12	13	
1	1,5	5	1	3	4	3	1	3	2,5	5	10	4	44
4	3	5	1	4	4	3	1	3	4	6	12	4	54

5,1

EXERCICE 1 – 4 POINTS *(- 1 POINT PAR RÉPONSE FAUSSE)*

Vous avez été embauché pour créer le document des exigences d'un nouveau système – le système QUEST - de définition, d'impression et de récolte de QCM papiers, relié au système de gestion des étudiants d'un institut de formation (qui recueille les résultats) et à la base de données de questions.

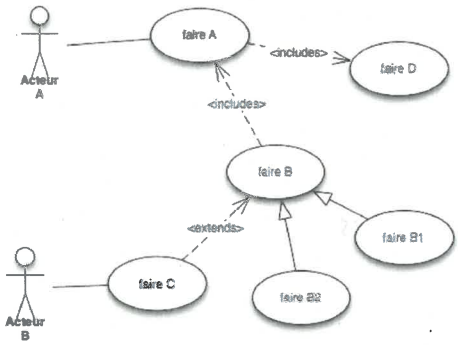
Décidez si chaque élément de la liste suivante est un *acteur externe*, un *acteur principal*, un *acteur secondaire* (auxiliaire), ou s'il n'est pas du tout un acteur.

	Acteur externe	*Acteur principal*	*Acteur secondaire*	*Pas un acteur*
L'analyseur optique de QCM (analyse des réponses) au moment où le professeur lance l'évaluation des résultats.			X ✓	
La base de données des questions qui permet de sélectionner une question au moment de la confection d'un QCM			X ✓	
Le professeur (qui définit les QCM)		X ✓		
La souris de l'ordinateur qui permet de cliquer sur les boutons (`JButton`)			X	-1
Le système informatique de gestion des étudiants (pour que le professeur puisse accéder à la liste des étudiants)				X -1
Le directeur de l'école pour lequel sont enregistrés les statistiques des résultats	X ✓			
Le responsable de l'entretien de l'imprimante		X		-1
L'imprimante utilisée pour l'impression des QCM.			X ✓	

4/4 1

1

EXERCICE 2 – 3 POINTS *(- 1.5 POINTS PAR RÉPONSE FAUSSE)*



1.5/3

Les propositions suivantes sont-elles vraies ou fausses ?

- ☑ Pour « Faire B1 », il faut « Faire D »
- ☐ Pour « Faire B1 », il faut « Faire C »
- ☑ Pour « Faire C », il faut « Faire B »

-1.5

EXERCICE 3 - 5 POINTS *(- 1.5 POINTS PAR RÉPONSE FAUSSE)*

LE VRAI ET LE FAUX DE LA RELATION EXTENDS

Dans le cadre des « cas d'utilisation », et plus spécifiquement de la relation « extends », cocher les assertions qui sont vraies.

5/5

- ☐ La relation « extends » permet de modéliser une variante au cas de base étendu.
- ☒ La relation « extends » permet d'étendre un cas d'utilisation de base, d'où le sens de la flèche.
- ☐ La rédaction du cas de base doit se référer aux différentes extensions qu'il possède
- ☐ La relation « extends » permet de compléter un cas d'utilisation de base dont la rédaction a été bloquée, en lui rajoutant une fonctionnalité considérée comme obligatoire.
- ☒ La relation « extends » permet de modéliser des activités asynchrones, pouvant interrompre le cas de base étendu

EXERCICE 4 - 1 POINTS

Quel est le principal intérêt du cycle de vie itératif ?

1/1

Le client ~~peut~~ a accès a une executable à chaque itération → il peut valider au fur et à mesur le projet

meilleur gestion des risques ✓

7,5

EXERCICE 5 - 4 POINTS

- 1POINT PAR RÉPONSE FAUSSE

Méthode UP. Indiquer par une croix dans quelle phase l'intensité de chaque discipline/activité est maximum (choix exclusif !)
Init : Initialisation – **Elab** : Elaboration – **Constr** : Construction – **Trans** : Transition

	Init	Elab	Constr	Trans
Réalisation de l'architecture centrale	o	X	X	o
Livraison finale	o	o	o	X
Réalisation d'un module	o	o	X	o
Etude de faisabilité	X	o	o	o
Modélisation de domaine	o	o	X	o
Rédaction des cas d'utilisation	o	X	o	o
Planification des itérations	o	X	o	o
Génération d'un sous-ensemble exécutable	o	o	X	o

EXERCICE 6 - 4 POINTS

- 1.5 POINT PAR RÉPONSE FAUSSE

La crise du logiciel des années 70 a eu pour conséquence - *dans le courant des deux ou trois années qui ont suivi ce constat* - d'opérer une remise en question du processus de développement logiciel et des outils utilisés. Notamment :

- ☐ L'arrêt de la programmation procédurale au profit d'une programmation orientée objet.
- ☒ La promotion de la phase d'analyse des besoins
- ☐ La promotion d'un cycle de vie de type itératif
- ☐ La réduction systématique de la taille des équipes de développement

EXERCICE 7 - 3 POINTS

- 1POINT PAR RÉPONSE FAUSSE

Parmi les différents critères de qualités énumérés ci-dessous, indiquez s'il s'agit d'une qualité « externe », « interne » ou alors ayant trait au processus de développement. Une seule réponse possible par critère de qualité.

	Externe	Interne	Processus
Réutilisation	o	X	o
Robustesse	X	o	o
Correction	X	X	o
Maintenabilité	o	X	o
Evolutivité	o	X	o
Respect des délais	o	o	X

EXERCICE 8 - 1 POINT

Les gens du Génie Logiciel sont parfois tordus. Ils opèrent une distinction assez nette entre le fait de *vérifier* le logiciel et le fait de le *valider. Qu'est-ce que cela signifie ?*

do the rigth things and do the things right
→ valider → vérifier
→ le programme doit faire ce que le client veut → le programme ne doit pas avoir de bugs / être correcte

EXERCICE 9 - 3 POINTS

Qualité d'un logiciel. Citez :

- Trois critères de qualité externe robustesse / ergonomie / performance
- Trois critères de qualité interne portabilité / ~~réutilisabilité~~ maintenabilité / évolutivité
- Deux critères ayant trait au processus de développement -> respect des délais / communication

3/3

EXERCICE 10 - 4 POINTS

-1.5 points par réponse fausse

Cochez, parmi les assertions suivantes concernant UP, celle ou celles qui vous paraissent exactes :

- [x] Il arrive que les itérations de UP soient de durée différente.
- [x] UP prend en compte la gestion des risques.
- [x] La méthode de gestion du processus UP s'applique à une modélisation procédurale et/ou objet.
- [ ] La planification globale du projet ainsi que l'estimation réaliste des coûts est réalisée dans la phase d'initialisation

-1.5

2,5/4

EXERCICE 11 – 6 POINTS

Dresser le diagramme de cas d'utilisation décrivant les informations manipulées par une agence de voyage qui répond à des demandes de clients en leur proposant des hôtels susceptibles de répondre aux critères de la demande. Trois acteurs : le conseiller de l'agence, le client, le gestionnaire.

Pour opérer une demande, un client a deux façons d'opérer, soit il remplit un formulaire par internet soit il choisit d'avoir un entretien téléphonique avec un conseiller de l'agence.

Au cas où la demande est faite par internet, cette dernière est automatiquement enregistrée dans le système. Sinon, c'est au conseiller de l'agence d'enregistrer la demande dans le système.

Le conseiller de l'agence répond à une demande en recherchant les hôtels susceptibles de répondre aux critères demandés puis en envoyant une proposition au client par e-mail après avoir enregistré dans le système la liste des hôtels proposés au client.

Cette réponse peut être opérée par le conseillé lorsque ce dernier consulte l'état des demandes de client. Elle peut également être opérée aussitôt, au moment où le client opère sa demande par téléphone.

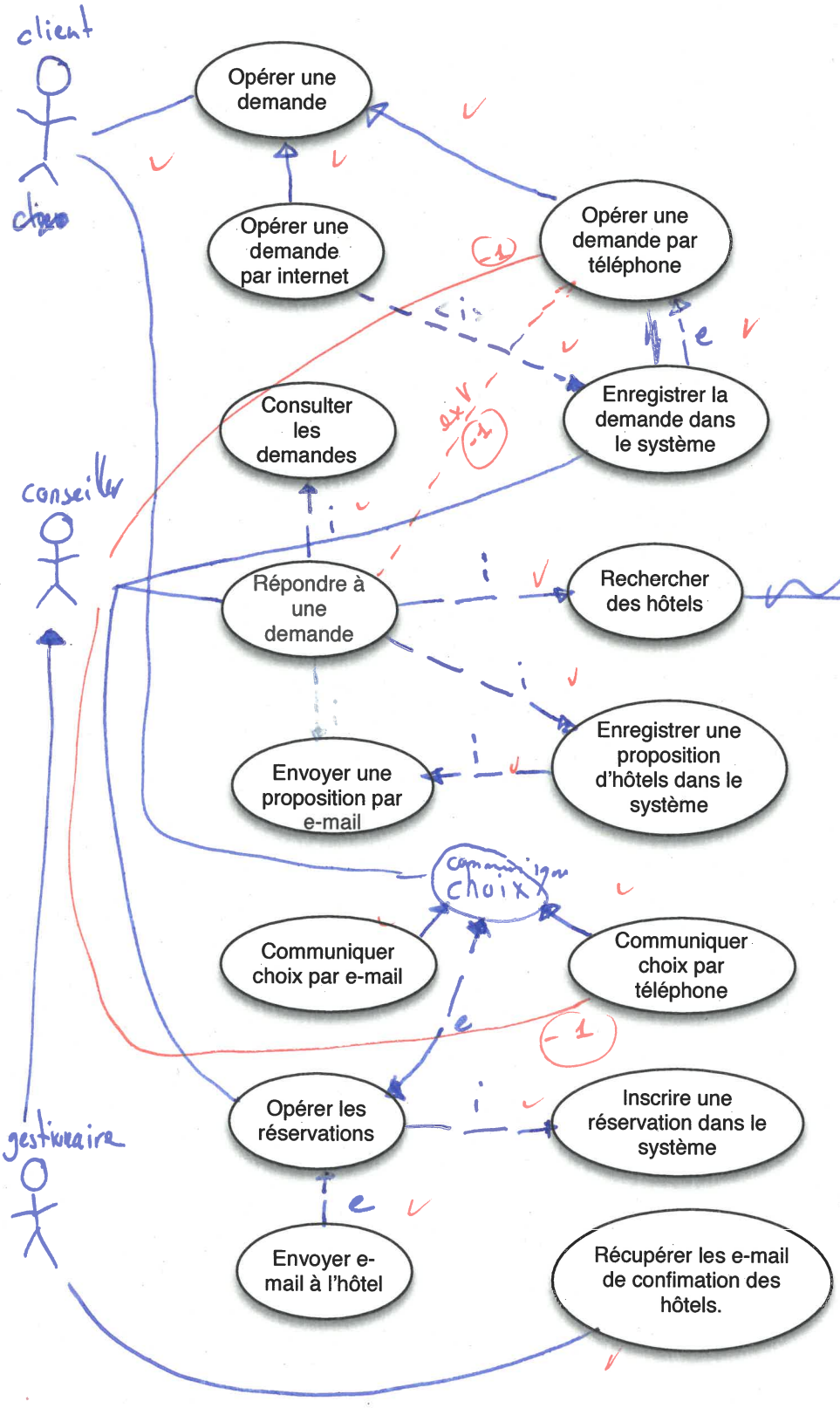
Pour répondre à une proposition, le client peut communiquer son choix par email ou par téléphone.

Une fois que le choix opéré par le client est connu, le conseiller peut alors opérer les réservations qui s'accompagneront d'un enregistrement de la réservation dans le système et en envoyant le cas échéant un e-mail à l'hôtel (si ce dernier est réservé par un opérateur de type e-booking).

Enfin, parmi les conseillers de l'agence, ceux qui jouent également le rôle de gestionnaire peuvent avoir à récupérer les e-mails de confirmation des hôtels.

Vous pouvez formuler votre réponse en complétant le diagramme présenté ci-après :

5,5



inc = include
ext = extends

Ok qu'on a pas le droit de rajouter, il faut relier le client au 2. ✓

-3 ⇒ 5/6

(-0,4 par faute)

EXERCICE 12 – 12 POINTS

On désire réaliser en Java une horloge qui affichera son heure sous la forme `hh:mm:ss`

Adepte inconditionnel du modèle Observable-Observé, vous décidez de réaliser votre programme en vous appuyant sur 4 classes :

- Une classe `Compteur`, qui sera instanciée 3 fois, une fois pour le compteur des heures, une fois pour le compteur des minutes et une fois pour le compteur des secondes.

  Le constructeur de la classe `Compteur` aura un constructeur qui acceptera au moins 2 paramètres : la valeur initiale et la valeur maximale avant de recommencer à zéro (par exemple 59 pour le compteur des secondes). Un compteur n'est pas un objet actif : il incrémente sa valeur uniquement lorsqu'il reçoit une notification de mise à jour. Responsabilité du compteur : notifier chaque changement de valeur à la vue associée, notifier également chaque remise à zéro pour donner une impulsion au compteur suivant dans la chaîne.

- Une classe `VueHorloge`, observant les 3 compteurs, mettant à jour sa valeur interne à chaque notification, sa valeur interne constituée de 3 strings de 2 caractères : les heures, les minutes et les secondes. A chaque notification, la vue de l'horloge affiche l'état de l'horloge `hh:mm:ss` dans la console, au moyen d'un `System.out.println`.

- Une classe `Controleur` dont le constructeur va créer les 3 compteurs, la vue de l'horloge et différentes opérations d'initialisation. Le contrôleur sera un objet actif : il donnera toutes les secondes une « impulsion» au compteur des secondes en le notifiant. Il s'agira du seul objet actif du système.

Que faire ??

- Ecrivez le code des 3 classes `Compteur, VueHorloge & Controleur`
- Contraintes :
  - Toutes les notifications sont opérées par le biais du modèle Observable/Observé
  - Par soucis d'efficacité, les objets observeurs ne doivent pas être notifiés par des événements qui ne les concernent pas.

EXERCICE 13 - 4 POINTS

Soit le code source suivant :

```
public class MultiCompteur {
    private int registre = 0;
    private int iteration;
    public void effectue() {
        Runnable r1 = new Runnable() {
            public void run() {
                for(int i=0; i<iteration; i++) {
                    int tmp = registre;
                    tmp = tmp + 1;
                    registre = tmp;
                }
            }
        };
        Runnable r2 = new Runnable() {
            public void run() {
                for(int i=0; i<iteration; i++) {
                    int tmp = registre;
                    tmp = tmp - 1;
                    registre = tmp;
                }
            }
        };
        Thread t1 = new Thread(r1);
        Thread t2 = new Thread(r2);

        t1.start();
        t2.start();
        System.out.println("Nous avons au final: "+registre);
    }
    public MultiCompteur(int iteration) { this.iteration = iteration; }
    public static void main(String args[]) {
        MultiCompteur mc = new MultiCompteur(100000000);
        mc.effectue();
    }
}
```

private Object objet = new Object(); ← permet de faire la synchro. soit voir si l'objet est utilisé ou non

objet

synchronized (registre) {

synchronized (objet) {

4/4

Répondre aux deux questions suivantes :

- Le concepteur du programme aurait voulu qu'à la fin de l'exécution des deux méthodes `run`, le registre ait la valeur 0. Or ce n'est probablement pas le cas. Pourquoi ? Comment y remédier ? (proposer des modifications du code). car les 2 threads r1 et r2 accède à la même ressource registre. ils peuvent donc écrire en même temps → problème
- A la fin de la méthode `effectue()`, nous avons un affichage avec l'appel à `System.out.println()`. Cette instruction produira de toute façon un résultat aberrant (indépendamment du point précédent). Pourquoi ? on a aucune garentie que les 2 threads soient terminés. car il peut y avoir des préemptions. Comment y remédier ? (Proposer des modifications du code).

il faut attendre la fin des 2 threads en faisant t1.join(); et t2.join();

il faut entourer la section critique avec un synchronized (objet...) {

Basile Chatillon 10.04.2017

Exercice 12

```
public class Compteur extends Observable implements Observer {
    private final int vMin, vMax;
    private final Compteur nextCompteur
    private Controleur cont;
    private int vCourant
    public Compteur (Controleur c, int min, int max, Compteur nc) {
        nextCompteur = nc;
        cont = c;
        vMin = min;
        vMax = max;
        vCourant = min;
    }

    private void incrementer () {
        vCourant++;
        if (vCourant >= vMax) { vCourant = vMin; }
    }

    public void update (Observable c, Object arg) {
        // récuperation de la nouvelle valeur.
        incrementer ();
        setChanged ();
        notify (vCourant);
    }

    public void incrementer () {
        vCourant ++;
        if (vCourant > vMax) {
            vCourant = vMin;
            if (nextCompteur != null) { nextCompteur.update(this, null); }
        }
    }

    public Compteur () { vMin=0; vMax=0; vCourant=0; cont=null; nextCompteur=null; }
}
```

1 seul paramètre ! (-1)

Observers (comme dans le contrôleur)

(-2) c'est de la triche !
c.f énoncé
⇒ utiliser le mécanisme Observable/Observer

```
public class Controleur extends Observable {
    Timer t;
    Compteur s, m, h;
    VueHorloge v1;

public Controleur ( ) {
    h = new Compteur (this, 0, 23, null);
    m = new Compteur (this, 0, 59, h);
    s = new Compteur (this, 0, 59, s);

    this.addObservable (s);

    v1 = new VueHorloge(s, m, h)

    h.addObserver (v1);
    m.addObserver (v1)
    s.addObserver (v1);

    t = new Timer ( ) {

        void run ( ) {
            notifyObservers ( );
            sleep (1000);
        }
    }
}
```

j'aurais pu faire ça dans le constructeur de v1.

Basile Chatillon 10.04.2017

```
public class VueHorloge implements Observer {
  private Compteur csec; cmin, cheure;
  private String sec, min, heure;
  public Horloge (Compteur c1, Compteur c2, Compteur c3) {
        csec = c1;
        cmin = c2;
        cheure = c3;
        sec = new String("00");
        min = new String("00");
        heure = new String("00");
  }

  void update (Observable o, Object arg) {
      if (o == csec) {
          sec = arg[0];
      } else if (o == cmin) {
          min = arg[0];
      } else if (o == chour) {
          heure = arg[0];
      }
      system.out.println(heure + ":" + min + ":" + seconde);
  }
}
```

sec = (String) arg

j'ai ~~eu~~ pas assez de place dans la class ~~Compteur~~ compteur pour ajouter la méthode :

```
public void setNextCompteur ( Compteur nc ){
        nextCompteur = nc ;
}
```

j'en ai pas besoin en fait