Université Libre de Bruxelles Mardi 20 août 2013

Faculté des Sciences Appliquées

INFO-H-200 : Programmation Orientée Objet

Examen

Remarques préliminaires

- N'oubliez pas d'inscrire nom, prénom et numéro de matricule sur chaque feuille.
- Vous disposez de trois heures et vous ne pouvez pas utiliser de notes.
- Pour les questions où l'on vous demande de justifier ou d'expliquer, la justification ne doit pas faire plus de 4 ou 5 lignes.
- Choisissez bien l'ordre dans lequel vous répondez aux questions : elles n'ont pas toutes le même rapport entre valeur en points et temps nécessaire.
- L'examen est sur 12 points, sauf si le projet diminue la note finale (et dans ce cas l'examen sera comptabilisé sur 20), les 8 autres points sont alloués au projet.

Question 1 – QCM (/2)

Pour chaque question, soulignez l'unique bonne réponse.

Dans les langages de programmation compilés (java, C#, C++), laquelle de ces trois assertions est vraie ?

- Le compilateur vérifie la bonne syntaxe des classes
- Le compilateur vérifie le bon comportement des objets
- Le compilateur vérifie la bonne syntaxe des classes et le comportement des objets conforme à leur classe

Laquelle de ces assertions est correcte ?

- La classe permet de découper les données du programme en des blocs plus simples
- La classe est comme un petit programme à part entière avec ses propres données et ses propres fonctions
- La classe est la partie plus petite d'un algorithme

Laquelle de ces assertions est correcte ?

- Les méthodes d'une classe ne portent que sur les attributs de cette même classe
- Les méthodes d'une classe donnée peuvent agir sur les attributs de toutes les classes ayant un rapport avec celle-ci.
- Les méthodes d'une classe n'agissent que sur les attributs déclarés public de cette même classe.

Laquelle de ces assertions est correcte ?

- Une méthode statique n'agit qu'au démarrage du code
- Une méthode statique peut s'exécuter sans créer d'objets
- Une méthode statique n'agit que sur les attributs privés d'une classe

Le constructeur d'une classe a pour rôle :

- De créer les objets de cette classe
- D'initialiser les attributs des objets issus de cette classe
- De vérifier que l'objet est toujours bien présent dans la mémoire vive de l'ordinateur

Un objet peut :

- N'être référé qu'une et une seule fois
- Partager son référent avec d'autres objets
- Etre référé de multiples fois

L'association entre deux classes signifie :

- Qu'une classe peut envoyer des messages à une autre
- Qu'une classe possède comme attribut un référent vers l'autre qu'elle utilise pour lui envoyer des messages
- Qu'une classe s'occupe de la vie et la mort des objets de l'autre classe

L'encapsulation a pour rôle premier

- De favoriser la stabilité des codes et limiter la portée des changements
- D'empêcher une classe d'interagir avec les autres
- De rendre une classe invisible aux autres

Si A est une super-classe de B, laquelle de ces 4 instructions ne compile pas ?

- A a = new A()
- B b = new B()
- A b = new B()
- B a = new A()

Si A est une super-classe de B, « a » un objet issu de la classe A et « b » un objet issu de la classe B (A a = new A() et B b = new B()), laquelle de ces trois instructions ne compile pas ?

- a = b
- b = a
- b = (B) a

Si A est une super-classe de B, laquelle de ces trois assertions est correcte ?

- Le constructeur de B ne s'occupe d'initialiser que ses propres attributs
- Le constructeur de B s'occupe de l'initialisation de ses attributs et ceux de A

- Le constructeur de B fait d'abord appel au constructeur de A puis s'occupe de l'initialisation de ses propres attributs

Dans l'exemple du match de foot, le polymorphisme provient du fait que :

- L'entraîneur envoie un message unique aux joueurs en ne se préoccupant pas de leur nature ultime
- L'entraîneur envoie un massage unique aux joueurs en sachant qu'ils l'exécuteront tous de la même manière
- L'entraîneur envoie un message unique aux joueurs en sachant qu'au moment de la réception chaque joueur l'exécutera selon sa nature ultime

Quelle est la bonne définition d'une classe abstraite ?

- Une classe contenant une ou plusieurs méthodes abstraites
- Une super-classe
- Une classe qui ne peut donner naissance à des objets

Quel est le rôle premier des interfaces ?

- Jouer le rôle de mode d'emploi d'une classe pour les autres classes
- Favoriser le multihéritage
- Récupérer les seules méthodes publiques d'une classe

Question 2 - Code Java (/3)

Découvrez ce que ce code Java parfaitement correct écrira à l'écran lors de son exécution.

```
abstract class Etudiant {
 protected boolean sexe;
 protected int QI;
 private Groupe monGroupe;

 public Etudiant(boolean sexe, int QI) {
  this.sexe = sexe;
  this.QI = QI;
 }

 public int getQI() {
  return QI;
 }

 public void setGroupe(Groupe monGroupe) {
  this.monGroupe = monGroupe;
 }

 abstract public int passeExamen();

 public int passeProjet(){
  return monGroupe.passeProjet();
 }
```

```
  public int calculNote() {
    return (passeExamen() + passeProjet());
  }
}

class Crack extends Etudiant {
  public Crack(boolean sexe, int QI) {
    super(sexe,QI);
  }

  public int passeExamen() {
    if ((sexe) && QI > 80) {
      return 10;
    }
    else {
      return QI/8;
    }
  }
}

class Moyen extends Etudiant {
  public Moyen(boolean sexe, int QI) {
   super(sexe,QI);
  }

  public int passeExamen() {
    if ((sexe) && QI > 60) {
      return 8;
    }
    else {
      return QI/9;
    }
  }
}

class Nullard extends Etudiant {
  public Nullard(boolean sexe, int QI) {
    super(sexe,QI);
  }

  public int passeExamen() {
   if ((sexe) && QI > 40) {
     return 6;
   }
   else {
     return QI/10;
   }
  }
}

class Groupe {
  private Etudiant[] lesEtudiants = new Etudiant[3];
  Projet p;
```

```
  Groupe(Etudiant e1, Etudiant e2, Etudiant e3, Projet p) {
    lesEtudiants[0] = e1;
    lesEtudiants[1] = e2;
    lesEtudiants[2] = e3;
    for (int i=0; i<lesEtudiants.length;i++) {
      lesEtudiants[i].setGroupe(this);
    }
    this.p = p;
  }
  public int passeProjet() {
    return p.evalue(lesEtudiants);
  }
}

abstract class Projet {
  public int totalQI(Etudiant[] lesEtudiants) {
    int total = 0;
    for (int i=0; i<lesEtudiants.length; i++) {
     total = total + lesEtudiants[i].getQI();
    }
    return total;
  }
  abstract public int evalue(Etudiant[] lesEtudiants);
}

class JeuExpert extends Projet {
  public int evalue(Etudiant[] lesEtudiants) {
    if (totalQI(lesEtudiants) > 150) {
      return 10;
    }
    else {
      return 8;
    }
  }
}

class JeuMoyen extends Projet {
  public int evalue(Etudiant[] lesEtudiants) {
    if (totalQI(lesEtudiants) > 100) {
      return 8;
    }
    else {
      return 6;
    }
  }
}

class JeuNovice extends Projet {
  public int evalue(Etudiant[] lesEtudiants) {
    if (totalQI(lesEtudiants) > 50) {
      return 6;
    }
    else {
      return 4;
    }
  }
}
```

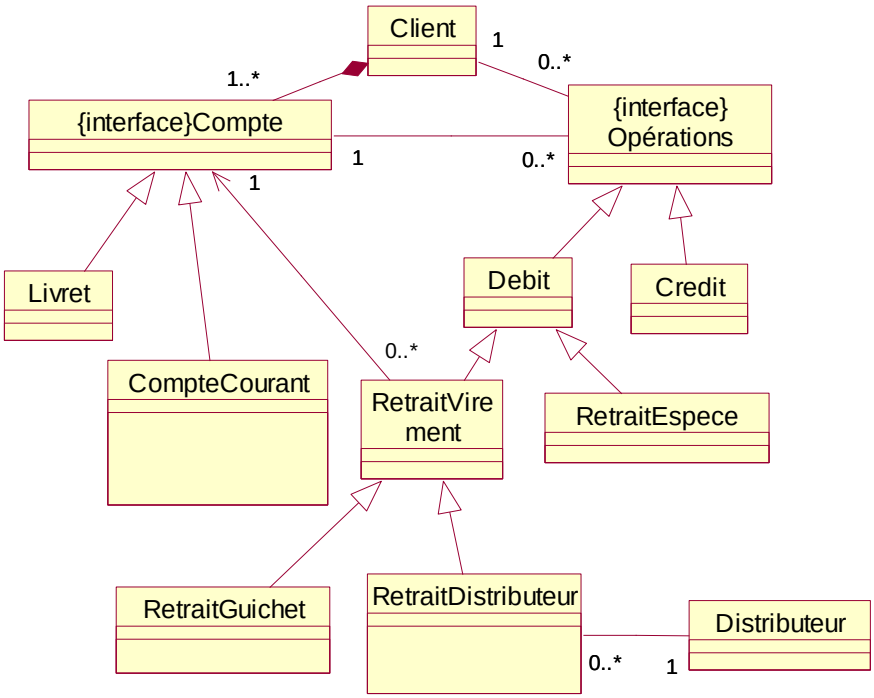
```
public class Examen{
  Etudiant[] lesEtudiants= new Etudiant[9];
  Projet[] lesProjets= new Projet[3];
  Groupe[] lesGroupes = new Groupe[3];

 public Examen () {
   lesEtudiants [0] = new Crack(true, 70);
   lesEtudiants [1] = new Moyen(false, 50);
   lesEtudiants [2] = new Moyen(true, 30);
   lesEtudiants [3] = new Crack(false, 60);
   lesEtudiants [4] = new Crack(false, 50);
   lesEtudiants [5] = new Nullard(true, 10);
   lesEtudiants [6] = new Moyen(true, 60);
   lesEtudiants [7] = new Moyen(false, 70);
   lesEtudiants [8] = new Nullard(true, 10);
   lesProjets[0] = new JeuExpert();
   lesProjets[1] = new JeuMoyen();
   lesProjets[2] = new JeuNovice();

   for (int i=0; i<3; i++) {
     int ii= i*3;
     lesGroupes[i] = new Groupe(lesEtudiants[ii], lesEtudiants[ii+1],
                        lesEtudiants[ii+2], lesProjets[i]);
   }

   for (int i=0; i<lesEtudiants.length; i++) {
     System.out.println(lesEtudiants[i].calculNote());
   }
 }

 public static void main(String[] args) {
   new Examen ();
 }
}
```

Question 3 – UML (/2)

L'université désire se doter d'un système informatisé d'accès par carte magnétique à ses différents bâtiments et locaux. Tout accès par un de ses membres universitaires à un moment précis sera contrôlé et enregistré. Parmi les différents lieux accessibles, on distinguera les bâtiments, les auditoires, les salles de TP et les bureaux. Sont repris parmi les membres universitaires, le personnel technique, catégorisé en personnel de sécurité et d'entretien, le personnel académique, catégorisé en professeurs et assistants, et finalement les étudiants. Le personnel de sécurité a un droit d'accès à tous les lieux. Le personnel d'entretien a un accès limité aux bâtiments, auditoires et aux salles de TP. Les professeurs et les assistants ont également un droit d'accès aux bâtiments, auditoires, salles de TP. Les professeurs peuvent également accéder à leur bureau particulier (chaque professeur a son propre bureau). Les étudiants ne peuvent accéder qu'aux bâtiments. Représentez le diagramme de classe UML de cette application réalisée dans un pur esprit orienté objet, en prenant bien soin d'indiquer les classes (on se passera à ce stade des méthodes et attributs) et les liens existant entre celles-ci (avec la cardinalité associée).

Question 4 : UML → Java (/2)

Traduisez le diagramme de classe UML suivant dans le squelette de code Java (en ajoutant également les constructeurs) le plus fidèle qu'il serait possible de générer automatiquement à partir de ce diagramme.



Question 5 : Théorie (/1)

Dans la plupart des processeurs Intel aujourd'hui, il existe deux niveaux de mémoire cache. Pendant que le processeur travaille, entre la cache de premier niveau et la cache de deuxième niveau des transferts d'information se produisent, sans pour cela interrompre le processeur. Expliquez le pourquoi de ces transferts et la raison pour laquelle ces transferts s'effectuent dans les deux sens, du premier niveau vers le deuxième tout autant que du deuxième vers le premier.

Question 6 – Théorie (/1)

Tentez d'écrire dans le langage des instructions élémentaires du processeur ce que donnerait la compilation du petit code Python ci-dessous :

```
>>> compteur=10
>>> i=0
>>> while i<compteur:
        print i
        i=i+1
```

Question 7 - Théorie (/1)

Soit un ensemble de 6 routeurs Internet labélisés (A,B,C,D,E,F). On connaît le parcours effectué par 5 paquets circulant sur ce réseau :

1) A C E F
2) C D E
3) B F E D
4) A F E D
5) A C E D F

Retrouvez à partir de ces informations le contenu des tables de routage associées à chacun de ces 6 routeurs.