Contrôle intermédiaire

Durée : 2h – Documents interdits

Questions de cours (2,5 pts)

1. Quelle est la signification du mot clé ***final*** lorsqu'il est employé devant :
   a. Un attribut b. Une méthode c. Une classe

Corrigé : 0,5*3=1,5

- un attribut final ne peut être initialisé qu'une seule fois
- une méthode final ne peut pas être redéfinie
- une classe final ne peut être dérivée

2. Quelle est la signification du mot clé ***static*** lorsqu'il est employé devant :
   a. Un attribut b. Une méthode

Corrigé: 0,5*2=1

- un attribut static est un attribut de classe, il n'appartient à aucun objet, il est commun pour tous les objets.
- une méthode static est une méthode de classe qu'on peut appeler même sans créer un objet du type de la classe, et qui ne peut manipuler que des attributs statique, et

Exercice 1 (3 pts) Qu'affichent les codes suivants ?

a/

```
class testException extends Exception{}

class A1{
       public void methodeA(){ System.out.println("A");}
       public void methodeB() throws ArithmeticException {
              int a=3; int b=0;
              System.out.println("B"+(a/b));
         }
       public void methodeC() throws testException {  throw new testException();  }
}

public class B1{
        public static void main ( String[] args ){
          A1 appel = new A1();
          try {
            appel.methodeA();
            appel.methodeB();
            appel.methodeC();
          }
          catch ( ArithmeticException e ){
             System.out.println("Exception arithmétique");

          }
          catch ( testException e ){e.printStackTrace();
                }
          finally {
```

```
            System.out.println("Fin");
          }
        System.out.println("Suite");
      }
    }
```

Corrigé: 0,25*4=1pt

A
Exception arithmétique
Fin
Suite

b/

```
class A2 {
  protected static int x = 1;
  private int y = 2;
  public void affiche(){
       System.out.println("A: x=" + x + ", y=" + y); }
  public void set(int y){this.y = y;}
}
class B2 extends A2 {
  private int y = 3;
  private float z = 4;
  public void affiche(){
       super.affiche();
       System.out.println("B: y=" + y + ", z=" + z);
  }
  public void set(float z) { this.z = z; }
  public void set(int y){this.y = y;}
}
public class V2 {
  static void set(int i) { i = 5 * A2.x; }
  static void set(A2 a, int i) { a.set(i); }
  static void set(float f, B2 a) { a.set(f); }
  public static void main(String[] args) {
       int i = 5;
       float f = 7;
       A2 a = new A2();
       B2 b = new B2();
       b.affiche();
       A2.x = i;
       a.affiche();
       set(i);
       set(a, 4*i); a.affiche();
       set(b, 3*i); b.affiche();
       set(f, b); b.affiche();
 }
}
```

Corrigé:

0,25pt*8= 2pts

A: x=1, y=2
B: y=3, z=4.0
A: x=5, y=2
A: x=5, y=20
A: x=5, y=2
B: y=15, z=4.0
A: x=5, y=2
B: y=15, z=7.0

Exercice 2 (4.5 pts)

1. Si les classes Camion et Bus dérivent de la classe Vehicule et la classe MiniBus dérive de la classe Bus alors, peut-on écrire :

a) Vehicule [] tab = new MiniBus [4] ;
b) MiniBus [] tab = new Camion [4] ;
c) Vehicule [] tab = new Vehicule [4] ;
d) MiniBus [] tab = new Bus [4] ;
e) Vehicule [] tab = new Camion [4] ;

Corrigé: 0,25*5=1,25
a. ok, tout minibus est un véhicule
b. non car y a pas de lien entre minibus et Camion
c. ok, même type
d. non, tout minibus est un bus, mais pas l'inverse
e. oui, car tout camion est un véhicule

2. Parmi les quatre propositions précédentes, laquelle permet de créer un tableau pouvant contenir des Camions, des Bus et des MiniBus (a, b, c, d ou e)? Pourquoi ?
Corrigé: la c car ils sont tous des véhicules et peuvent donc être référencé par un objet de ce type general (0,5)

3. Soit la méthode *afficherType()* définie dans la classe Véhicule, qui affiche (« je suis un objet de type Véhicule »), qui est redéfinie dans les classes MiniBus (« je suis un objet de type MiniBus ») et Camion (« je suis un objet de type Camion »). En supposant que le tableau « tab » a été correctement créé dans la question 2, que produit l'exécution des instructions suivantes :

```
tab[0]=new MiniBus () ;tab [1]= new Vehicule() ;tab[2]= new Camion () ;tab[3]= new Bus () ;
for(int i = 0; i < tab.length; i++) tab[i]. afficherType ();
```

Corrigé: 0,25*4=1
je suis un objet de type MiniBus
je suis un objet de type Véhicule
je suis un objet de type Camion
je suis un objet de type Véhicule

3. En prenant en considération les affectations de la question 3, nous effectuons les instructions suivantes : (chaque ligne d'instructions est indépendante des autres, elles ne sont pas exécutées en séquence !)

```
MiniBus m= tab[0] ;
tab [1]= tab[0] ; Bus b= tab [1] ;
```

Vehicule v= tab[2] ;

Camion c = tab[3] ;

Pour chaque affectation, dites si elle fonctionne sans erreur, si elle provoque une erreur à la compilation ou si elle provoque une erreur à l'exécution. S'il y a une erreur expliquez pourquoi et dites si une correction est possible. Si oui, expliquez ce qu'il faut faire pour que l'affectation soit valide (à la compilation et à l'exécution)

Corrigé (1.75)

Ligne d'instructions	Erreur à la compilation ? (Oui/Non)	Correction possible ? (Oui/Non)	Correction	Erreur à l'execution ? (Oui/Non)
MiniBus m= tab[0] ; 0.25 (si 3 premières cases justes)+0.25 (dernière case)	OUI, car tab est de type Véhicule qui n'est pas un minibus	Oui,	MiniBus m= (MiniBus) tab[0]	NON car tab[0] fait reference à un objet de type minibus
tab [1]= tab[0] ; Bus b= tab [1] ; 0.25 (si 3 premières cases justes)+0.25 (dernière case)	OUI car tab[1] est de type Vehicule et ne peut pas pas être affecté à une reference de type Bus	Oui	Bus b= (Bus) tab [1] ;	NON car tab[1] fait reference à un objet de type Minibus qui est un sous-type de Bus
Vehicule v= tab[2] ; : 0,25	NON, les deux sont de type Vehicule			
Camion c = tab[3] 0.25 (si 3 premières cases justes)+0.25 (dernière case)	OUI, car tab est de type Véhicule qui n'est pas un Camion	Oui,	faire un cast vers Camion Camion c= (Camion) tab[3]	Oui, car tab[3] référence un objet de type Bus qui n'est pas compatible avec Camion

Exercice 3 (10 pts)

On désire réaliser une application informatique destinée à un hôtel qui loue des chambres et des bungalows. Chaque bungalow est caractérisé par un numéro, un nombre de pièces et un nombre de lits et dispose d'une cuisine. Quant aux chambres elles sont caractérisées par un numéro et un nombre de lits et peuvent offrir une vue sur la mer ou sur la piscine de l'hôtel. L'hôtel possède également des suites qui sont des chambres plus grandes équipées d'un salon.

Tous les logements, quel que soit leur type, offrent un confort de base mis à la disposition des clients (Téléviseur, réfrigérateur, climatiseur, connexion wifi), mais les clients qui réservent un bungalow ou une suite peuvent demander des équipements supplémentaires : fer à repasser et sèche-cheveux pour les suites et les bungalows et matériel de cuisine pour les bungalows…etc. Ce service engendre

un coût supplémentaire qui sera ajouté au prix de la location.

L'hôtel fixe un tarif par nuitée pour chaque logement selon les critères suivants :

- *Le type du logement* : pour les bungalows un tarif est fixé par pièce et pour les chambres et les suites un tarif est fixé par lit.
- *Le nombre de pièces dans le bungalow et le nombre de lits dans la chambre ou la suite* : Le tarif par pièce (resp. lit) est multiplié par le nombre de pièces (resp. lits)
- *Le type de la vue dans les chambres et les suites* : Une vue sur la mer engendre un coût supplémentaire égal à 20%
- *L'ajout d'équipements supplémentaires* : pour une suite cela engendre un coût égal à 10% du tarif par nuitée alors que pour le bungalow le coût de ce service est égal à 15% du tarif/nuitée multiplié par le nombre de pièces dans le bungalow.

Exemple : soient les tarifs donnés dans le tableau suivant

	Bungalow Tarif/pièce/nuitée	Chambre Tarif/lit/nuitée	Suite Tarif/lit/nuitée
Tarif	5000 da	2500 da	3000 da

Voici à titre d'exemples les montants des tarifs de quelques logements :

- Tarif/nuitée d'une chambre de deux lits avec vue sur la piscine = 2500 * 2 = 5000 DA/Nuitée
- Tarif/nuitée d'une suite de quatre lits avec vue sur la mer = (3000+3000*0.2) * 4 = 14400 DA/Nuitée
- Tarif/nuitée d'une suite de quatre lits avec vue sur la piscine et équipements supplémentaires = (3000 * 4) + ((3000*4)*0.1)= 13200 DA/Nuitée
- Tarif/nuitée d'un bungalow de 3 pièces = 5000 * 3 = 15000 DA/Nuitée
- Tarif/nuitée d'un bungalow de 3 pièces avec équipement supplémentaire = (5000*3) + ((5000*0.15)*3)= 17250

Questions :

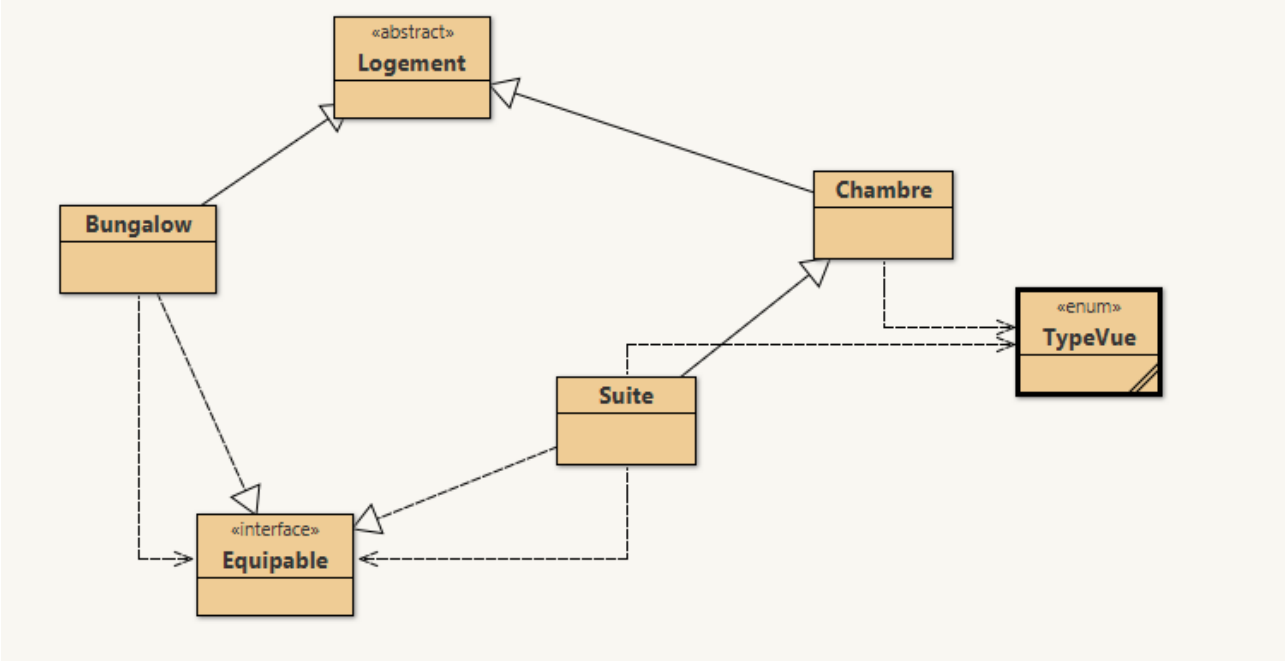
A/ proposer une conception orientée objet en traçant un diagramme des classes qui illustre les liens entre les différentes classes et en décrivant chacune d'elle à travers le tableau suivant.

Nom de la classe
Liste des attributs Modificateur d'accès Type de l'attribut nom de l'attribut (donnez un nom significatif)
Liste des méthodes Modificateur d'accès Valeur de retour Signature de la méthode description

Solution Type

Remarque : ceci est un corrigé type, toute autre solution correcte est prise en considération

Diagramme des classes



A/ 8.25

Remarque : les contructeurs, getters et setters sont volontairement omis.

Element	note
Présence des classes Logement Bungalow, Chambre Suite	(0.5*4)
Classe Logement abstraite	(0.25)
Présence d'une interface Equipable ou bien une classe abstraite Equipable	(0.5)
Lien d'héritage entre Bungalow et Logement et entre Chambre et logement	(0.5*2)
Si Equipable a été conçue comme étant une interface alors lien d'implémentation entre suite et Equipable et entre Bungalow et Equipable sinon si Equipable a été conçue comme étant une classe abstraite alors lien d'héritage entre suite et Equipable et entre Bungalow et Equipable	(0.5*2)
Logement protected int numero; protected float tarifType; protected int nbLits; public abstract double calculerTarif();	0.25*4

Bungalow private int nbPieces; private boolean equipementSupp; public double calculerSupplement() public double calculerTarif()	0.25*4

Chambre
protected TypeVue vue; // peut être une enum ou boolean ou String
protected final double suppVue = 0.2;// facultatif
public double calculerTarif()
	0.25*2

Suite private boolean equipementSupp; public double calculerSupplement() public double calculerTarif()	0.25*3

Equipable
public final double suppSuite = 0.1; // facultatif
public final double suppBungalow = 0.15; // facultatif
public double calculerSupplement()
	0.25

B/ Donner le code source java de la classe Bungalow. (1.75)

```
class Bungalow extends Logement implements Equipable //0.25*2 si déclaration correcte
{
// Ne pas noter les attributs car notés précédemment
    private int nbPieces;
    private boolean equipementSupp;

public Bungalow(int num, float tarifType, int nbLits int nbPieces, boolean supp)
    {
      // Appel au constructeur de la classe supérieure correctement 0.25
       super (num, tarifType, nbLits);
```

```
        this.nbPieces = nbPieces;
        equipementSupp = supp;
        // initialisation des champs propres à la classe 0.25
    }

public double calculerSupplement(){
    return tarifType*Equipable.suppBungalow*nbPieces;

    // remarque : Equipable.suppBungalow peut être remplacé par 0.15
// Redéfinition de la méthode claculerSupplément 0.25

    }

public double calculerTarif(){
       double tarif = tarifType*nbPieces;
       if (equipementSupp) tarif = tarif+ calculerSupplement();
       return tarif ;
    }
// Redéfinition de la méthode claculerTarif 0.5

}
```