

Mention : 171Spécialité : EigleUE : HNIN 102Sujet de M C. DONY & C. NebutDate de l'épreuve : 11 Janvier 2017N° de la Salle de Cours 16.02 ou de l'AmphiSession \* : 1 - ☒ 2 - ☐

\* Cocher la case utile

## AVIS IMPORTANT :

Tout signe de reconnaissance sur la copie, en particulier une signature sur la partie non cachée entraînera pour l'étudiant l'annulation de l'épreuve.

Ne pas écrire dans cette marge

NOTE

175  
20

Cadre réservé au correcteur

I - Framework version No 1, Bases réutilisation

- 1) a) Une affectation polymorphique est l'affectation d'un objet de type A par un objet de type B, où B est une sous classe de A.

exemple: A objet = new B(); // B est une sous-classe de A.

- b) Dans le listing 1, ligne 9 on a :

Heros h = new Packman(); // affectation polymorphique car Packman est une sous classe de heroes.

Ensuite nous avons instancié un objet de type Fantome f = new Fantome();

Replier et coller

Nom et Prénom du candidat :  
AL LAKATNuméro de la carte d'étudiant :  
Abdullahi

Signature

2036178

IMPORTANT : à remplir obligatoirement

N° de la copie :

2/4Exemple :  $\frac{1}{3} - \frac{2}{3} \dots$ 

195

0,25



au je  
g. addRencontre(f);

Cette dernière instruction ajoute l'objet  
de type Fantôme dans un "tableau"  
(Arraylist) qui est censé contenir des  
rencontres (objets de type RENCONTRE)

donc ici aussi l'objet ajouté devient  
polymorphe une fois ajouté car en java la  
résolution dynamique des arguments n'est pas faite.

L'appel setHeros(h) ne change rien car h est  
déjà déclaré de façon polymorphe.

2) Il faut adapter le patron de conception

Singleton qui limite le nombre de création  
d'objets à un seul exemplaire.

```
public class Fantôme extends Rencontre {  
    public static int compteur = 0;  
    private Fantôme () {  
    }  
    public static Fantôme creerFantôme ()  
        throws FantômeException {  
        if (compteur >= 4) {  
            throw FantômeException ();  
        } else {
```

```
        return new Fantôme ();  
    }  
}
```

3) Une méthode m1 paramétrée par composition  
a) est une méthode qui fait appel à des méthodes  
contenues dans d'autres objets qui peuvent être  
passés en argument ou qui sont des attributs  
de la classe <sup>qui</sup> définit m1.

En effet selon le type d'objet qu'elle a à sa  
disposition, le comportement de m1 varie. Donc  
m1 est bien paramétrée par composition.  
ex dans le listing 2:

rencontre(r) fait appel à r.toString()  
donc rencontre(r) se comportera (affichera)  
différemment selon la manière dont toString()  
est implémentée dans r.

Dans le cas du listing 2 par exemple, rencontre(  
Rencontre r) est paramétrée par composition,  
en fait elle est paramétrée par r, car  
selon l'objet r (Fantôme, cerise...), rencontre(r)  
va varier car elle fait appel à la  
méthode toString() contenue dans r.

b) la méthode rencontre(Rencontre) est paramétrée  
de deux façons différentes. En effet  
elle est paramétrée par composition (de son  
paramètre Rencontre), et par spécialisation

0,5

(elle fait appel à `this.toString()` qui sera redéfinie dans les sous-classes).

0,75

c) On peut étendre le framework en définissant une nouvelle classe de héros, qui sera donc une sous-classe de Héros et dans laquelle on redéfinira la méthode `toString()`.



Spécialité : Egle

UE : HTIN 102

Sujet de M C. DONY & C. NEBUT

Date de l'épreuve : 11 janvier 2017

N° de la Salle de Cours 16.02 ou de l'Amphi

Session \* : 1 - ☒ 2 - ☐

\* Cocher la case utile

AVIS IMPORTANT :

Tout signe de reconnaissance sur la copie, en particulier une signature sur la partie non cachée entraînera pour l'étudiant l'annulation de l'épreuve.

Ne pas écrire dans cette marge

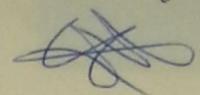
NOTE

Cadre réservé au correcteur

II - Framework version No2 - Spécialisations.

1)

Numéro de la carte d'étudiant : 20136178

Signature : 

Important : à remplir obligatoirement

N° de la copie :

2/4

Exemple :  $\frac{1}{3} - \frac{2}{3} \dots$



2) La méthode appelée est rencontrer (Fantôme) de la classe Pacman.

3) Alors dans le cas où rencontrer (Fantôme) n'est pas définie dans la classe Héros, c'est la méthode rencontrer (Rencontre) de la classe Héros qui est appelée.

4) Mes réponses aux deux questions précédentes s'expliquent par l'affectation polymorphique dans le listing 4.

Heros h = new Pacman();

En effet, en java, uniquement le simple dispatch est géré. Donc la résolution dynamique des liens n'est réalisée que pour les méthodes redéfinies.

Dans le cadre de la question 2, rencontrer (Fantôme) dans Pacman est une redéfinition de rencontrer (Fantôme) de Héros, donc comme le simple dispatch est géré par java, à l'exécution, la résolution dynamique des liens permettra de choisir la méthode la plus spécialisée à invoquer.

Cependant lors du contexte de la question 3, la méthode rencontrer (Fantôme) n'est plus définie dans Héros et donc celle de Pacman n'est plus une redéfinition mais une surcharge de

rencontrer (Param), cependant dans ce cas java utilise uniquement la résolution dynamique des liens et lors de la compilation fera le lien entre la méthode rencontrer (Rencontre) dans Héros car Héros est le type statique statique de l'objet h sur lequel est invoquée la méthode. En résumé java ne reconnaît que les méthodes présentes dans le type statique, ensuite lors de l'exécution à l'aide de la résolution dynamique des liens, il exécutera la méthode la plus spécialisée.

5)



### III - Gestion des changements de comportement (pour la version No 2).

B  
0,5  
1) Le pattern state est un schéma de conception pratique dans le cas d'objet nécessitant de changer de comportement selon le contexte. Il permet de définir un comportement pour chaque état de l'objet ainsi lorsque l'objet sera dans un état ou l'autre, il aura son comportement associé.

Dans notre cas, le personnage a deux états, un état normal et un état invincible obtenu en rencontrant une creuse. Lors de ces deux états, son comportement change, dans l'un il perd une vie si il rencontre un fantôme, dans l'autre il envoie les fantômes en prison sans perdre de vie.

Ici il est donc possible et conseillé de mettre en place le pattern state pour personnage, et créer deux états :

Etat Normal

Etat Invincible

Dans les deux états on pourra implémenter différemment les agissements lors des rencontres.



Mention : M1

Spécialité : Eigle

UE : HPIN102

Sujet de M C. DONY & C. NEBUT

Date de l'épreuve : 11 Janvier 2017

N° de la Salle de Cours 16.02 ou de l'Amphi

Session \*: 1 - ☒ 2 - ☐

\* Cocher la case utile

### AVIS IMPORTANT :

Tout signe de reconnaissance sur la copie, en particulier une signature sur la partie non cachée entraînera pour l'étudiant l'annulation de l'épreuve.

Ne pas écrire dans cette marge

NOTE

Cadre réservé au correcteur

Replier et coller

Nom et Prénom du candidat :  
AL LAHAD Abdulhadi

Numéro de la carte d'étudiant :  
20136178

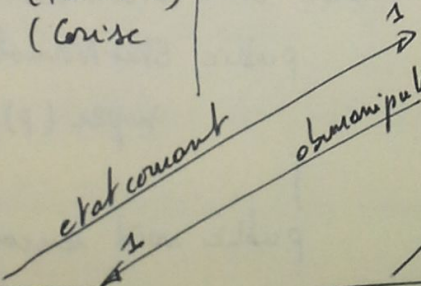
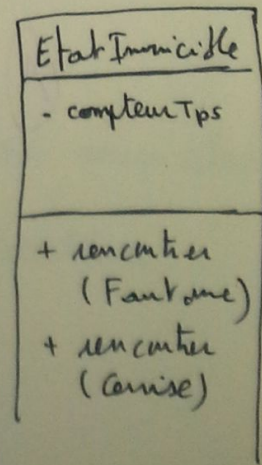
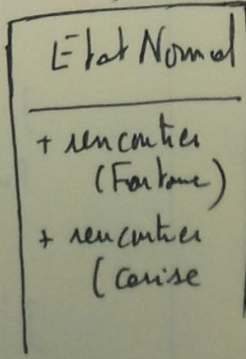
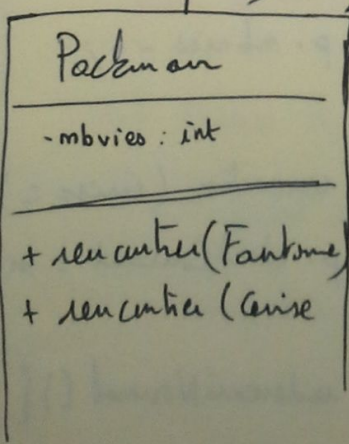
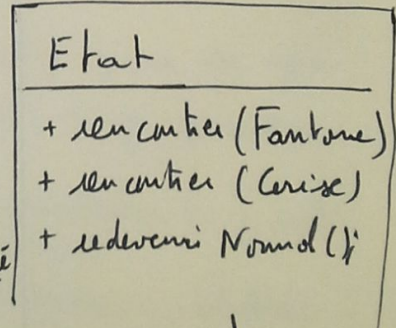
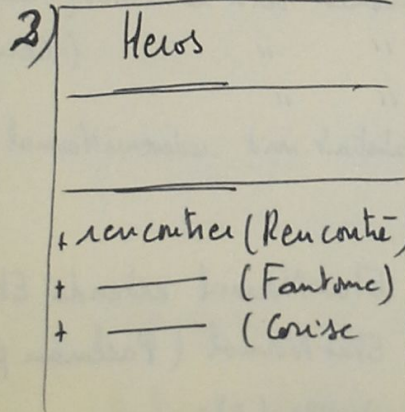
Signature

IMPORTANT : à remplir obligatoirement

N° de la copie :

3/4

Exemple :  $\frac{1}{3} - \frac{2}{3} \dots$





```

3) Public class Packman extends Hero {
    protected Etat etatCourant;
    public Packman() {
        this.etatCourant = new EtatNormal(this);
    }
    public void rencontrer (Fantome F) {
        this.etatCourant = rencontrer(F);
    }
    public void rencontrer (Cuisse c) {
        this.etat = rencontrer(c);
    }
}

```

```

Public abstract class Etat {
    Packman p;
    Public Etat (Packman p) {
        this.p = p;
    }
    Public abstract void rencontrer (Fantome f);
    " " " " (Cuisse c);
    " " "
    Public abstract void redevenirNormal();
}

```

```

Public class EtatNormal extends Etat {
    public EtatNormal (Packman p) {
        super(p);
    }
    public void rencontrer (Fantome F) {
        this.p.mbrues--;
    }
    Public void rencontrer (Cuisse c) {
        this.p.etatCourant = new EtatInvincible(p);
    }
    Public void redevenirNormal() {

```

// ne fait rien, elle peut renvoyer une exception  
mais pas fait par économie de temps

```

}
}

Public class EtatInvincible extends Etat {
    public EtatInvincible (Packman p) {
        super(p);
    }
    public void rencontrer (Fantome f) {
        f. envoyerEnIsms();
    }
    public void rencontrer (Cuisse c) {
        // on change prolonge le laps de temps?
    }
    // le compteur de temps, une fois le laps de temps
    écoute, fait appel à cette méthode pour changer
    d'Etat.
    public void redevenirNormal() {
        this.p.etatCourant = new EtatNormal();
    }
}

```



**UFR des Sciences  
MASTER SCIENCES ET TECHNOLOGIES**

Mention : R1Spécialité : EgleUE : NRIN102Sujet de M C.DONY Y C.NEBUTDate de l'épreuve : 11 janvier 2017N° de la Salle de Cours 16.02 ou de l'AmphiSession \* : 1 - ☒ 2 - ☐

\* Cocher la case utile

**AVIS IMPORTANT :**

Tout signe de reconnaissance sur la copie, en particulier une signature sur la partie non cachée entraînera pour l'étudiant l'annulation de l'épreuve.

Ne pas écrire dans cette marge

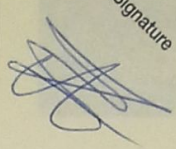
NOTE

Cadre réservé au correcteur

**IMPORTANT : à remplir obligatoirement**

Nom et Prénom du candidat : AL LAHAN Abdullahi

Numéro de la carte d'étudiant : 20136178

Signature : 

N° de la copie :

Exemple :  $\frac{1}{3} - \frac{2}{3} \dots$ Exercice 4

1) Comme dit précédemment, Java ne gère pas le multiple dispatch, du coup quand  $r$  est passé en paramètre à la rencontre ( $r$ ), seul (Rencontre) le type statique de  $r$  est reconnu, et donc rencontre(Rencontre) de Heros est appelée : celle de Pacman n'est pas une redéfinition

2) Dans la classe Rencontre et ses sous-classes il faut définir et redéfinir la méthode rencontre(Heros).

(0,5)



```
public class Rencontre {
```

```
    :
```

```
    public void rencontrer(Heros h) {
```

```
        h.rencontrer(this);
```

```
    }
```

```
    :
```

```
}
```

```
public class Fantome extends Rencontre {
```

```
    :
```

```
    :
```

```
    public void Rencontrer(Heros h) {
```

```
        h.rencontrer(this);
```

```
    }
```

```
    :
```

```
}
```

②

Ceci nous avons implémenté le patron  
Double-dispatch.