

JAVA – Programmation Objet
Examen de T.P.
IUT R&T 2ème année, 2005/2006

Consignes :

3 Heures.

Documents autorisés.

Vous commenterez votre code.

Vous rendrez la première page du sujet.

Nom :

Prénom :

Introduction : Concepts objets, syntaxe JAVA

```
public class monProgramme{
    public static void main(String s[])
    {
        Adresse monAdresse ;
        System.out.println(monAdresse.toString());
    }
}
```

A l'exécution du programme ci-dessus, une exception de type **NullPointerException** est lancée. Expliquez pourquoi.

Utilisation des tableaux

```
public class monProgramme{
    public static void main(String s[])
    {
        personne p[][] = new personne[10][10];
    }
}
```

Cocher la bonne Réponse. Le code ci-dessus :

- Crée un tableau (10x10) de références vers des personnes, mais ne crée pas les objets personnes
- Crée un tableau (10x10) de personnes (les objets personnes sont effectivement créés)

```
public class monProgramme{
    public static void main(String s[])
    {
        int i;
        Adresse mesAdresses[] = new Adresse[5];
        for (i=0; i<5; ++i) if (i%2==0) mesAdresses[i] = new Adresse();
        for (i=0; i<5; ++i)
            if (mesAdresses[i]!=null)
                System.out.println(mesAdresses[i].toString());
    }
}
```

Que fait ce programme ?

Travaux Pratiques

But : On vous a engagé pour réfléchir à la mise en place, en programmation orientée objets, d'un petit moteur de jeu de rôle / aventure. Après plusieurs discussions, vous arrivez aux conclusions suivantes :

De façon générale, on compte trois grandes familles d'éléments dans le jeu :

- Les personnages (héros, adversaires, monstres, ...)
- Les objets (équipement des personnages, trésors, ...)
- Les décors (essentiellement des murs)

Un niveau de jeu est décrit de la façon suivante :

- Une carte, qui comprend les éléments de décors, les objets et les personnages
- Un titre, qui décrit le niveau
- Un personnage spécial : celui du joueur.

Pour résumer :

un personnage **est un** élément
un objet **est un** élément
un décor **est un** élément
un niveau **est composé d'**une carte et d'un titre

Partie 1 (programmation objet)

Dans tous les cas :

- vous pouvez utiliser autant d'attributs et de méthodes privées que vous jugez nécessaire.
- créer un constructeur par défaut (sans paramètres),

Question 1 : Coder la classe abstraite `Element`.

Celle-ci doit contenir les méthodes publiques suivantes :

`getDescription()` qui retourne une chaîne de caractère décrivant l'élément.
`setDescription(String s)` qui met à jour la description d'un objet .
`getType()` qui retourne une chaîne de caractère indiquant le type de l'élément :
OBJET pour un objet, PERSONNAGE pour un personnage et DECOR pour un décor.

Question 2 : Coder la classe `Objet`

Cette classe dérive de la classe élément. Elle n'a aucun nouveau attributs ni méthodes, mais penser à implémenter la méthode `getType()`.

Question 3 : Coder la classe `Personnage`

Cette classe hérite de la classe `Element`

Elle doit implémenter les méthodes publiques suivantes :

`getName()` qui retourne une chaîne de caractère (le nom du personnage);
`setName(String s)` qui met à jour le nom du personnage ;
`getVitalite()` qui retourne un entier correspondant aux points de vie actuel du personnage
`setVitalite(int i)` qui met à jour la vitalité actuelle du personnage
`getVitaliteMax()` qui retourne un entier correspondant aux points de vie total du personnage
`setVitaliteMax(int i)` qui met à jour la vitalité totale du personnage

la description du personnage (retournée par la méthode `getDescription()`) doit comporter le nom du personnage, ainsi que des indications sur sa vitalité (totale et actuelle).

Penser à vérifier que la vie actuelle d'un personnage ne dépasse pas son maximum.

Un personnage peut porter jusqu'à trois objets.

Implémentez les méthodes suivantes de la classe personnage :

`afficheInventaire()` retourne une chaîne de caractère décrivant l'équipement
`ajouteObjet(Objet i)` ajoute l'objet `i` à l'équipement si il y a encore de la place
`poseObjet(int numero)` retire l'objet `numero` de l'équipement pour faire de la place et renvoie une référence vers cet objet

Pour la méthode `ajouteObjet(Objet i)`, vous pouvez gérer le cas où on tente de rajouter un équipement à un

personnage qui est déjà entièrement chargé de deux façons : soit en changeant la valeur de retour de cette méthode (plus simple) soit en utilisant les exceptions (mieux noté). Dans tous les cas, utiliser un tableau d'objets semble une solution simple à mettre en oeuvre.

Question 4 :

Implémentez la classe Niveau qui doit comporter les attributs et les méthodes suivantes :

Attributs :

carte : tableau d'Elements à deux dimensions (10 x 10 cases)
PJ : référence sur un objet de type Personnage (le personnage qui sera contrôlé par le joueur)
titre : le titre du niveau

Méthodes :

void setElement(int i, int j, Element k) : Ajoute un élément k à l'emplacement [i,j] de la carte (une case ne peut contenir qu'un seul objet à la fois).
Element getElement(i, j) : retourne une référence sur l'élément contenu à la case [i,j]
String decritCarte(int i, int j) : retourne la description de l'élément contenu dans la case, ou le texte « case vide » si la case ne contient pas d'éléments.
String decritCarte() : retourne la description, sous forme de chaîne de caractères, de chaque case de la carte (avec toujours « case vide » pour décrire les cases qui ne contiennent pas d'éléments).
Personnage getPersonnagePrincipal() : retourne une référence sur le personnage joueur.
void setPersonnagePrincipal(Personnage p) : donne une nouvelle valeur à PJ.
String getTitre() : retourne une référence sur le titre du niveau.
void setTitre(String t) : change le titre du niveau.

constructeurs :

Vous implémenterez un constructeur par défaut, qui crée un personnage principal
Vous implémenterez un constructeur ayant un personnage pour paramètre (personnage qui sera donc créé hors du constructeur)

Question 6 : tester vos implémentations en utilisant le programme principal mis à votre disposition, et en le modifiant si besoin.

Question 7 : à faire après la partie sur l'interface graphique,

Trois objets pour s'équiper, c'est peu... les concepteurs ont alors l'idée d'une classe d'objets plus spécifiques, les « contenants » qui sont des objets ayant la propriété de pouvoir contenir d'autres objets (sac, sacs, besaces, etc.)
Proposer et implémenter une solution pour ce type d'objet.

Partie 2 (interface graphique)

Créer une interface graphique ayant cet aspect :

- En haut, le titre du niveau
- Au centre, la carte du niveau :
 - on peut considérer qu'on utilise une grille de boutons pour représenter le niveau.
 - Le label du bouton est « P » pour un personnage « O » pour un objet, « D » pour un décor et rien du tout si cette partie de la carte est vide.
- Sur la droite, deux boutons l'un en dessus de l'autre.
 - Le label du premier bouton est « Personnage », le label du deuxième bouton est « Inventaire »
- Sur la gauche, les indications sur votre personnage : Son nom, sa vitalité actuelle et sa vitalité totale.
- Le bas de l'interface est réservé pour donner des indications, par exemple avec un label ou un champ de texte (à votre convenance)

Le comportement des boutons doit être le suivant :

Lorsque le bouton « Personnage » est actionné, le bas de l'écran doit afficher la description du personnage
Lorsque le bouton « Inventaire » est actionné, le bas de l'écran doit afficher l'inventaire du personnage

Lorsque l'un des boutons de la carte est actionné, le bas de l'écran doit afficher la description de l'élément qu'elle contient.

Annexe : Exemple de programme principal

```
public class main{
    public static void main(String S[])
    {
        Personnage PJ1 = new Personnage();
        PJ1.setName("Chevalier miteux...");
        PJ1.setDescription("personnage alcoolique mais chevaleresque...");

        Personnage gobelin1 = new Personnage();
        gobelin1.setName("Gobelin Teigneux");
        gobelin1.setDescription("Petit humanoide verdatre et primitif...");

        Objet gourde = new Objet();
        gourde.setDescription("Petite gourde en peau");

        Objet epee = new Objet();
        epee.setDescription("Epee Durandil");

        Objet casque = new Objet();
        casque.setDescription("Casque Lebohaum");

        System.out.println("-----");
        System.out.println("Affichage Personnages : ");
        System.out.println(PJ1.getDescription());
        System.out.println(gobelin1.getDescription());

        System.out.println("-----");
        System.out.println("Affichage Objets");
        System.out.println(gourde.getDescription());
        System.out.println(epee.getDescription());
        System.out.println(casque.getDescription());

        System.out.println("-----");
        System.out.println("Un peu d'equipement ? ");
        /* Normalement, c'est vide */
        System.out.println(PJ1.afficheInventaire());
        /* Ajoutons quelques objets */
        PJ1.ajouteObjet(epee);
        PJ1.ajouteObjet(gourde);
        PJ1.ajouteObjet(casque);

        /* Normalement, c'est plein */
        System.out.println(PJ1.afficheInventaire());

        /* Un nouvel objet pour voir */
        Objet armure = new Objet();
        armure.setDescription("Armure rutilante");

        /* On essaye de le rajouter, normalement, c'est impossible */
        PJ1.ajouteObjet(armure);
        System.out.println( PJ1.afficheInventaire() );

        /* On pose d'abord un objet, puis on essaye à nouveau */

        PJ1.poseObjet(0); /* dehors l'objet 0 */
        System.out.println( PJ1.afficheInventaire() );
        PJ1.ajouteObjet(armure);
        System.out.println( PJ1.afficheInventaire() );

        /* Testons la vitalité de notre personnage */
        System.out.println( "Point de vie actuels : "+PJ1.getVitalite() );
        System.out.println( "Point de vie maximal : "+PJ1.getVitaliteMax() );

        /* On change son nom ? */
        System.out.println(PJ1.getDescription() );
        PJ1.setName("Bob");
        System.out.println(PJ1.getDescription() );

        /*****
        MISE EN PLACE DU NIVEAU...
        *****/

        /* on crée quelques types de décors ? */
        Decors mur1 = new Decors();
        Decors mur2 = new Decors();

        mur1.setDescription("C'est un mur en bois, pas très épais");
        mur2.setDescription("C'est un mur en pierre, ça à l'air solide");
    }
}
```

```

/* Construction d'un niveau avec le constructeur par défaut */
Niveau monNiveau1 = new Niveau();

/* Mur en bois */
monNiveau1.setElement(1,1,mur1);
monNiveau1.setElement(1,2,mur1);
monNiveau1.setElement(1,3,mur1);
/* mur en pierre */
monNiveau1.setElement(5,5,mur2);
monNiveau1.setElement(5,6,mur2);
/* Certains objets trainent par terre... */
monNiveau1.setElement(3,3,epee);
monNiveau1.setElement(3,4,casque);
/* Et en plus, y'a des monstres... */
monNiveau1.setElement(7,7,gobelin1);
monNiveau1.setElement(8,8,gobelin1);
/*
   Question : Si le gobelin qui est dans la case 7,7 est modifié
   (perte de points de vie par exemple)
   Que ce passe-t-il pour le gobelin dans la case 8,8 ?
*/

System.out.println(monNiveau1.decritCarte());

/* Au fait, mon constructeur de niveau a-t-il bien crée un personnage par défaut ? */
Personnage toto = monNiveau1.getPersonnagePrincipal();
System.out.println(toto.getDescription());

/* Changeons de personnage principal */
monNiveau1.setPersonnagePrincipal(PJ1);

/* Et verifions le deroulement... */
toto = monNiveau1.getPersonnagePrincipal();
System.out.println(toto.getDescription());

/* On crée un nouveau niveau, avec le second constructeur */
Niveau nouveauNiveau = new Niveau(PJ1);

/* On teste ... */
toto = monNiveau1.getPersonnagePrincipal();
System.out.println(toto.getDescription());

System.out.println(nouveauNiveau.decritCarte());

}
}

```