Concept. et prog. objet

\1 TP n°1 : héritage

Dépublié

Détails

Écrit par stéphane Domas

Catégorie : M3105 - Concept. et prog, objet avancée (/index.php/menu-cours-s3/menu-mmi3-test)

Publication : 11 septembre 2009

Affichages : 4171

1°/ Le jeu de la vie

L'objectif est de modéliser la rencontre entre des humains et de faire évoluer une population. Pour cela, on va définir une hiérarchie simple de classes : Humain, Homme, Femme.

ATTENTION ! Les principes de la programmation objet devront être suivi au maximum, notamment en terme de réutilisation de code. Par exemple, dans le cas de méthodes de la classe Humain redéfinies dans les classe Homme et Femme, on s'efforcera, quand c'est possible, de ne pas réécrire totalement ces méthodes en appelant celle de la super classe.

1.1°/ La classe `Humain`.

Créer le fichier Humain.java à partir du canevas suivant :

```
import java.io.*;
import java.util.*;

public class Humain {

    protected static Random loto = new Random(Calendar.getInstance().getTimeInMillis());
    protected int age;
    protected int poids;
    protected String nom;
    protected int esperanceVie;

    Humain(String nom) {

    // a compléter
    }

    Humain(int age, int poids, String nom) {

    // à compléter
    }

    void setNom(String nom) {
    // à compléter
    }

    void setAge(int age) {
    // à compléter
    }

    void setPoids(int poids) {
    // à compléter
    }

    int getAge() {
    // à compléter
    }

    int getPoids() {
    // à compléter
    }

    String getNom() {
    // à compléter
    }

    protected void setEsperanceVie() {
    // à compléter
    }

    public void vieillir() {
    // à compléter
    }

    public void grossir(int p) {
    // à compléter
    }
```

```
57
58      public boolean isDead() {
59      // à compléter
60      }
61
62      public void print() {
63      // à compléter
64      }
65  }
```

Pour compléter le code :

- le premier constructeur prend en paramètre une variable de type String et initialise les attributs age à 0 et poids à 3, le nom avec la valeur de la variable, puis met à jour la valeur d'`esperanceVie`. Le deuxième constructeur fait la même chose mais permet di'nitialiser l'âge et le poids à une valeur précise.
- `void vieillir()` incrémente de 1 l'âge.
- `void setEsperanceVie()` fixe l'espérance moyenne d'un Humain à 70 an.
- `void grossir(int p)` ajoute au poids de la valeur p, sachant que p peut être négatif.
- `boolean isDead()` renvoie true si age > esperanceVie.
- `void print()`, affiche un message donnant la valeur des attributs.

1.2°/ la classe Femme.

Créer le fichier `Femme.java` à partir du canevas suivant :

```java
import java.io.*;

public class Femme extends Humain {

    private int fertilite;

    Femme(String nom) {
    // à compléter
    }

    Femme(int age, int poids, String nom, int fertilite) {
    // à compléter
    }

    int getFertilite() {
    // à compléter
    }

    public void vieillir() {
    // à compléter
        if (age <= 20) poids = 3+(int)(2.6*age);
        else if (age >= 50) poids += (age % 2);
    }

    public Humain rencontre (Homme h) {
    // à compléter
    }

    protected void setEsperanceVie() {
    // à compléter
    }
}
```

Pour compléter le code :

- l'attribut `private int fertilite` représente la chance d'avoir un bébé.
- le constructeur qui prend en paramètre une variable de type String appelle le constructeur de `Humain` correspondant, puis met à 0 `fertilite`. L'autre constructeur prend en paramètres quatre variables servant à initialiser les attributs en appelant le constructeur de `Humain` correspondant.
- `void setEsperanceVie()` fixe l'espérance moyenne d'une Femme entre 55 et 95 ans.
- la méthode `void vieillir()` :
  - incrémente l'âge de 1
  - vérifie si age = 15 et si c'est le cas, tire une valeur entre 0 et 100 grâce à loterie et affecte cette valeur à l'attribut fertilite.
  - fait augmenter le poids (cf. code déja écrit)
- la méthode `Humain rencontre(Homme h)` :
  - vérifie si this.age > 15 **et** this.age < 50 **et** h.age > 15, et si ce n'est pas le cas, la méthode renvoie immédiatement `null`.
  - vérifie si this.poids > 150 **ou** h.poids > 150, auquel cas la méthode renvoie immédiatement `null`.
  - tire un chiffre f entre 0 et 100 grâce à `loterie`.
  - vérifie si f > this.fertilite, auquel cas la méthode renvoie immédiatement `null.`
  - tire un chiffre p entre 0 et 100 grâce à `loterie`.
  - si p < 50, alors la méthode instancie un objet Homme, sinon, la méthode instancie un objet Femme. Le nom du bébé est la concaténation des noms du père et de la mère.
  - tire un chiffre g entre 0 et 20, et fait grossir l'homme de g et la femme de 10

- renvoie l'objet instancié.

1.3°/ La classe Homme.

Créer le fichier Homme.java à partir du canevas suivant :

```
import java.io.*;

public class Homme extends Humain {

    private int batifolage;

    Homme(String nom) {
    // à compléter
    }

    Homme(int age, int poids, String nom, int batifolage) {
    // à compléter
    }

    public Humain rencontre (Femme f) {
    // à compléter
    }

     public void vieillir() {
    // à compléter
        if (age <= 20) poids = 3+(int)(3.6*age);
        else if (age >= 50) poids += (age % 2);
    }

    protected void setEsperanceVie() {
    // à compléter
    }
}
```

Pour compléter le code :

- l'attribut `batifolage` qui représente le pourcentage de chance qu'un homme batifole lorsqu'il rencontre une femme, c'est-à-dire qu'il ne veut certainement pas faire un bébé.
- un constructeur qui prend en paramètre une variable de type String, qui appelle le constructeur de `Humain` correspondant. Il met l'attribut `batifolage` à 0.
- un constructeur qui prend en paramètres quatre variables servant à initialiser l'attribut de Homme et les attributs de `Humain` en appelant le constructeur correspondant
- `void setEsperanceVie()` fixe l'espérance moyenne d'un homme entre 50 et 80 ans.
- la méthode `vieillir()` :
  - incrémente l'âge de 1
  - si age > 15, tire une valeur aléatoire entre 70 et 100 pour initialiser l'attribut batifolage.
  - si age > 30, tire une valeur aléatoire entre 20 et 50 pour initialiser l'attribut batifolage.
  - si age > 60, tire une valeur aléatoire entre 50 et 100 pour initialiser l'attribut batifolage.
  - incrémente le poids (cf. code déjà écrit)
- la méthode `Humain rencontre(Femme f)` :
  - tire un nombre b entre 0 et 100.
  - vérifie si b < batifolage (c.a.d si l'homme veut juste batifoler), auquel cas la méthode renvoie immédiatement `null`.
  - vérifie si f.age > 15 **et** f.age < 50 **et** this.age > 15, et si ce n'est pas le cas, la méthode renvoie immédiatement `null`.

- vérifie si this.poids > 150 **ou** f.poids > 150, auquel cas la méthode renvoie immédiatement `null`.
- tire un nombre c entre 0 et 100.
- vérifie si c > f.fertilite, auquel cas la méthode renvoie immédiatement `null.`
- tire un chiffre p entre 0 et 100.
- si p < 50, alors la méthode instancie un objet Homme, sinon, la méthode instancie un objet Femme. Le nom du bébé est la concaténation des noms du père et de la mère.
- tire un chiffre g entre -10 et 10, fait grossir l'homme de g et la femme de 10.
- renvoie l'objet instancié.

Remarque : d'après ce qui est indiqué ci-dessus, la méthode `rencontre()` de Homme n'est pas symétrique avec celle de Femme.

2°/ Le moteur du jeu.

L'objectif est de créer une population de départ puis de la faire évoluer sur un certain nombre d'années.

2.1°/ La classe Population

Créer un fichier Population.java contenant le canevas de code suivant :

```java
import java.util.*;
import java.io.*;

class Population {

    List<Humain> pop;

    Population() {
    // à compléter
    }

    public void vider() {
    // à compléter
    }

    public void addHumain(Humain h) {
    // à compléter
    }

    public Humain getHumain(int index) {
    // à compléter
    }

    public void removeHumain(Humain h) {
    // à compléter
    }

    public void removeHumain(int index) {
    // à compléter
    }

    public int taille() {
    // à compléter
    }

    public void vieillir() {
    // à compléter
    }

    public void print() {
    // à compléter
    }
}
```

Pour compléter le code :

- le constructeur instancie lla liste pop sous forme d'un ArrayList
- `void vider()` permet de vider pop.
- `void addHumain(Humain h)` ajoute h en fin de pop.
- `Humain getHumain(int index)` renvoie l'objet stocké en index dans la liste.
- `Humain removeHumain(int index)` supprime de pop l'objet stocké en index et le renvoie.
- `Humain removeHumain(Humain h)` supprime de pop l'objet h et le renvoie.
- `int taille()` renvoie le nombre d'humains dans pop.

- `void vieillir()` incrémente l'âge de tous les humain de 1, et pour chacun vérifie s'il est mort. Si c'est le cas, l'humain est retiré de pop.
- `print()` affiche les humains de pop.

2.2°/ Le programme principal

- créez un fichier TP1`.java` contenant la méthode main.
- deux paramètres seront fournis à main lors de l'exécution : le nombre de tours de jeu dans `args[0]`, et la taille de la population initialedans `args[1]`.
- déclarez et instanciez une variable `Population population`.
- instanciez `tailleInit/2` Homme et `tailleInit/2` Femme, en les insérant dans `population`. Utilisez le constructeur qui permet de fixer la valeur de tous les attributs en prenant comme valeur :
  - nom = hommeX ou femmeY, avec X/Y un nombre >1 qui sera incrémenté après chaque instancation d'un homme ou d'une femme.
  - age = 20 pour tous
  - poids = 70 pour les hommes, et 55 pour les femmes.
  - fertilite tirée entre 1 et 100 pour chaque femme.
  - batifolage tiré entre 70 et 100 pour chaque homme (cf. méthode vieillir())
- faîtes une boucle de 0 à nbToursJeu. Pour chaque tour :

1. Tirez aléatoirement un chiffre n < taille actuelle de la population divisée par 2.

2. Faîtes n rencontres. Pour ce faire, tirez aléatoirement deux entiers i1 et i2 qui représentent des indices dans la List.
   - pour chaque tirage, récupérez les objets h1 et h2 stockés dans population aux indices i1 et i2,
   - Comme h1 et h2 sont des variables de type Humain, il faut identifier de quelle instance ils sont réellement. Pour cela, utilisez le mot clé instanceof. Si h1 et h2 sont respectivement des instances d'Homme et de Femme, ou l'inverse, alors appelez la méthodes de rencontre adéquate et récupérez ce qu'elle retourne dans une variable `Humain bebe`.
   - si `bebe != null` alors la rencontre a été fertile et vous devez insérer ce nouvel individu dans la population et afficher un message indiquant cette insertion.

3. Faîtes vieillir toute la population.

Remarque : comme h1 et h2 sont obligatoirement des variables de type Humain, il va falloir les transtyper pour pouvoir appeler la bonne méthode de rencontre. Par exemple :

```
...
Humain h1,h2;
Homme ho;
Femme fe;
...
h1 = population.getHumain(i1);
h2 = population.getHumain(i2);
if (h1 instanceof Homme) {
   ho = (Homme)h1;
...
```

3°/ Améliorations

- Le fait d'utiliser instanceof est une bidouille Java, pas dans l'esprit POO. Pour évtier son utilisation, il suffit de faire des méthodes boolean isHomme() et boolean isFemme(). A vous de trouver où il faut définir ces méthodes pour que l'on puisse ensuite faire des tests du type `if (h1.isHomme()) ...`
- Faîtes un tirage des rencontres sans doublons : si un indice à déjà été tiré, il ne peut plus être utilisé pour les autres tirages.

Remarque : refaire le tirage tant que la valeur à déjà été tirée N'EST PAS une bonne solution.

- Au lieu d'insérer les nouveaux éléments dès qu'ils sont créés, on les stocke dans une table secondaire et on les insère dans la population à la fin du tour, juste après la vérification des morts.