Société Java UML

Introduction

Une société est composée d'individus. Les individus sont des hommes ou des femmes. Les hommes et les femmes peuvent donner naissance à d'autres individus. Au départ, seuls Adam, un homme, et Eve, une femme, existent dans la société. Ils donnent naissance à des individus qui eux-mêmes donnent naissance à d'autres individus, etc. Un individu possède donc un père, une mère, des fils et des filles. Un individu peut malheureusement mourir.

Objectif et méthode

En annexe figure un programme Java modélisant cette société. Le but de l'examen est de comprendre ce que fait ce programme et de corriger les erreurs qui y sont encore contenues.

Question 1

java ne peut compiler le `main`. Pourquoi ? Corriger le `main` en conséquence.

Question 2

On suppose que l'utilisateur tape **A** pour afficher tous les individus de la société. Donner la sortie du programme. Que se passe-t-il ? Modifier le programme et donner la nouvelle sortie du programme.

Question 3

Dessiner le *diagramme de généralisation* UML de ce programme. Mettre les classes `Object` et `VectorI`.

Question 4

Dessiner le *diagramme classe-association*. Ne pas mettre les classes `Object` et `VectorI`.

Question 5

a) Quelles sont les méthodes redéfinies ?
b) Quelle est la différence entre les méthodes `toNom()` et `toString()` ?
c) A quoi sert la classe `VectorI` ?

Question 6

a) Ce programme permet-il de faire naître plusieurs individus ? Peut-on faire naître des individus de parents de même sexe ? l'inceste est-il possible ?
b) Dessiner un *diagramme de séquence* UML correspondant à l'interaction de l'utilisateur avec ce programme en supposant qu'adam et **eve** ont une fille **lrbbqs**, que **lrbbqs** et **adam** ont un fils **bhcwxc** puis que l'utilisateur quitte le programme.

Question 7

On suppose que l'utilisateur et le programme ont suivi le diagramme de séquence de la question précédente. Dessiner le *diagramme d'instances* UML correspondant à l'état du programme juste avant que l'utilisateur quitte le programme.

Question 8

a) On suppose que l'utilisateur et le programme sont encore dans l'état du diagramme d'instances de la question précédente et que l'utilisateur tape **M** puis **lrbbqs** pour que lrbbqs meurt, puis **A** pour afficher tous les individus de la société. Donner la sortie du programme. Que se passe-t-il ?
b) Pour résoudre ce problème, on déclare les méthodes suivantes :

```
void cetEnfantEstMort(Individu)
void mesEnfantsJeSuisMort(VectorI)
```

dans la classe `Individu`. Donner les définitions de ces deux méthodes.
c) Dans la méthode `mort()`, ajouter les appels aux méthodes ci-dessus afin de résoudre ce problème.
d) Donner la nouvelle sortie du programme.

Annexe

```
class Societe {

    public char menu() {
       System.out.println( "Societe :");
       System.out.println( "\t N. Naissance");
       System.out.println( "\t M. Mort");
       System.out.println( "\t A. Affichage");
       System.out.println( "\t Q. Quitter");
       char r = Keyboard.getChar();
       switch(r) {
       case 'N': Individu.naissance(); break;
       case 'M': Individu.mort(); break;
       case 'A': System.out.println(Individu.mesInstances); break;
       }
       return r;
    }

    public static void main(String argv[]) {
       System.out.println( "bonjour");
       Societe  s;
       Homme  adam = new Homme("adam");
       Femme  eve = new Femme("eve");
       while (s.menu()!='Q');
       System.out.println( "au revoir");
    }

    public static String nomHasard() {
       char c[] = { ' ', ' ', ' ', ' ', ' ', ' ', ' ' };
       for (int i=0; i<6; i++) c[i] = (char) ( ((int)'a') + engendrer(26));
       String s = new String(c);
        return s;
    }

    private static Random ra = new Random();

    public static int engendrer(int n) {
       int a = Math.abs(ra.nextInt()) % n ;
       return a;
    }
}

class Individu {

    public static VectorI mesInstances = new VectorI();
    public String nom;
    public Homme  monPere;
    public Femme  maMere;
    public VectorI mesFils;
    public VectorI mesFilles;

    public Individu(String n) {
       nom = n;
       monPere = null;
       maMere = null;
       mesFils = new VectorI();
       mesFilles = new VectorI();
       mesInstances.addElement(this);
    }

    public void detruire() {
       mesInstances.removeElement(this);
    }

    public String toString() {
       return ("\n"+ toNom()+ toParents() + toEnfants());
    }
```

```
    public String toNom() {
       return ("individu " + nom + "\n");
    }

    public String toEnfants() {
       return ("mes fils: \n"   + mesFils.toNom()   +
              "\nmes filles: \n" + mesFilles.toNom() +"\n");
    }

    public String toParents() {
        return (toPere()+toMere());
    }
    public String toPere() {
       return ("mon pere: \n" + monPere.toNom() + "\n");
    }
    public String toMere() {
       return ("ma mere: \n" + maMere.toNom() + "\n");
    }

    public static void naissance() {
       System.out.print("nom pere ? ");
       String sh = Keyboard.getString();
       Homme  h = (Homme) Homme.mesInstances.getInstance(sh);
       if (h==null) {
           System.out.println("pas d'homme portant ce nom.");
           return;
       }
       System.out.print("nom mere ? ");
       String sf = Keyboard.getString();
       Femme  f = (Femme) Femme. mesInstances.getInstance(sf);
       if (f==null) {
           System.out.println("pas de femme portant ce nom.");
           return;
       }

       int sexe = Societe.engendrer(2);
       String s = Societe.nomHasard();

       Individu  enfant = null;
       switch(sexe) {
       case 0:
           enfant = new Homme(s);
           h.mesFils.addElement(enfant);
           f.mesFils.addElement(enfant);
           break;
       case 1:
           enfant = new Femme(s);
           h.mesFilles.addElement(enfant);
           f.mesFilles.addElement(enfant);
           break;
       }
       enfant.monPere = h;
       enfant.maMere = f;
       System.out.println(enfant);
    }

    public static void mort() {
       System.out.print("nom ? ");
       String s = Keyboard.getString();
       Individu  i = Individu.mesInstances.getInstance(s);
       if (i!=null) {
           i.detruire();
           System.out.println("individu mort.");
       }
       else System.out.println("pas d'individu portant ce nom.");
    }
}
```

```
class Homme extends Individu {

    public static VectorI mesInstances = new VectorI();

    public Homme(String n) {
       super(n) ;
       mesInstances.addElement(this);
    }

    public void detruire() {
       mesInstances.removeElement(this);
       super.detruire() ;
    }

    public String toNom() {
       return ("homme "+nom+"\n");
    }
}



class Femme extends Individu {

    public static VectorI mesInstances = new VectorI();

    public Femme(String n) {
       super(n);
       mesInstances.addElement(this);
    }

    public void detruire() {
       mesInstances.removeElement(this);
       super.detruire() ;
    }

    public String toNom() {
       return("femme "+nom+"\n");
    }
}




class VectorI extends Vector {
    public String toNom() {
       String s = "[\n";
       for (int x=0; x< size(); x++) {
           Individu i = (Individu) elementAt(x);
           if (i==null) return s+"]\n";
           s = s + i.toNom() + "\n,\n";
       }
       return s+"]\n";
    }

    public Individu getInstance(String  n) {
       for (int x=0; x< size(); x++) {
           Individu i = (Individu) elementAt(x);
           if (i==null) return null;
           if (i.nom.equals(n)) return i;
       }
       return null;
    }
}
```