Corrigés des exercices avancés sur la récursivité

Exercice 7.2.1 *mari et femme*

Cet exercice est difficile. Le corrigé que nous proposons est un exemple intéressant de programmation, car la classe contient une variable d'instance qui est du type de la classe lui-même. Nous avons là un exemple de structure récursive qu'il n'est pas facile de comprendre quand on n'a pas l'habitude.

Question 1

Faites une première version du programme où seul le nom du conjoint apparaît dans la classe `UnePersonne`. La classe comportera une méthode pour marier la personne à quelqu'un (un autre objet instance de la classe `UnePersonne` et une autre pour faire divorcer la personne.

Il y aura des exceptions pour empêcher que les mineurs de moins de 18 ans se marient, pour empêcher que les gens déjà mariés ne se remarient et pour empêcher les gens non mariés de divorcer.

```
class UnePersonne{
    String nom;
    int age;
    boolean sexeFeminin;
    String conjoint;
    UnePersonne(String n, int a, boolean s){
        age = a;
        nom = n;
        sexeFeminin = s;
    }
    void seMarieA(UnePersonne p) throws TropJeune, PasLibre,
    SoiMeme, MemeSexe{
        if (age<18){
            throw new TropJeune();
        }
        if (conjoint != null){
            throw new PasLibre();
        }
        if (p == this){
            throw new SoiMeme();
        }
        if (p.sexeFeminin == sexeFeminin){
            throw new MemeSexe();
        }
```

```
            conjoint = p.nom;
            if (p.conjoint != this.nom){
                try{
                    p.seMarieA(this);
                }catch(TropJeune e){
                    conjoint = null;
                    throw e;
                }catch(PasLibre e){
                    conjoint = null;
                    throw e;
                }
            }
        }
        void divorce() throws PasMarie{
            if (conjoint == null){
                throw new PasMarie();
            }
            conjoint = null;
        }
}
class TropJeune extends Exception{}
class PasLibre extends Exception{}
class SoiMeme extends Exception{}
class MemeSexe extends Exception{}
class PasMarie extends Exception{}
class Exo17_2_1{
    static void essayeDeMarier(UnePersonne p1, UnePersonne p2){
        try{
            p1.seMarieA(p2);
            Terminal.ecrireStringln(p1.nom + "␣est␣marie␣a␣" + p2.nom);
        }catch(TropJeune ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣et␣");
            Terminal.ecrireStringln(p2.nom + "␣echoue");
            Terminal.ecrireStringln("Un␣des␣deux␣est␣trop␣jeune");
        }catch(PasLibre ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣et␣");
            Terminal.ecrireStringln(p2.nom + "␣echoue");
            Terminal.ecrireStringln("Un␣des␣deux␣est␣deja␣marie");
        }catch(SoiMeme ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣avec␣");
            Terminal.ecrireStringln("soi-meme␣echoue");
        }catch(MemeSexe ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣et␣");
            Terminal.ecrireStringln(p2.nom + "␣echoue");
            Terminal.ecrireStringln("Ils␣ou␣elles␣sont␣du␣meme␣sexe");
        }
    }
    static void essayeDeDivorcer(UnePersonne p1){
        try{
            p1.divorce();
            Terminal.ecrireStringln(p1.nom + "␣divorce");
        }catch(PasMarie ex){
            Terminal.ecrireString(p1.nom + "␣n'est␣pas␣marié(e)␣");
```

```
            Terminal.ecrireStringln("␣divorce␣impossible.");
        }
    }
    public static void main(String[] args){
        UnePersonne caroline = new UnePersonne("Caroline",41,true);
        UnePersonne paul = new UnePersonne("Paul",35,false);
        UnePersonne rosaria = new UnePersonne("Rosaria",25,true);
        UnePersonne virginie = new UnePersonne("Virginie",37,true);
        UnePersonne roel = new UnePersonne("Roel",37,false);
        essayeDeMarier(paul,virginie);
        essayeDeMarier(paul,rosaria);
        essayeDeMarier(rosaria,caroline);
        essayeDeDivorcer(paul);
        essayeDeMarier(paul,rosaria);
        essayeDeMarier(caroline,caroline);
    }
}
```

Un des gros défauts de ce programme, c'est qu'on ne peut pas assurer que le divorce est une opération réciproque. Pour le mariage, le conjoint étant une personne passée en paramètres, toutes les vérifications nécessaires sont faites sur ce paramètre. Le mariage est écrit de façon réciproque. Pour le divorce, il n'y a pas de moyen d'accéder à l'objet représentant le conjoint. Dans la version suivante du programme, on accède directement à cet objet à travers la variable `conjoint`.

Question 2

Changez le classe obtenue à la question 1 en remplaçant dans la classe le nom du conjoint par une référence à l'objet qui représente le conjoint.

On a beaucoup plus d'information sur le conjoint, ce qui permet de vérifier notamment que les deux personnes sont de sexe opposés. Vous traiterez cela avec une exception.

Question 3

On cherche à présent à assurer que les informations de mariage et de divorce soient cohérentes, c'est à dire que quand a a pour conjoint b, il faut que b ait pour conjoint a. D'autre part, si a divorce de b, alors b ne doit plus avoir pour conjoint a. Il faut pour cela que les méthodes de mariage et de divorce changent non seulement l'objet sur lequel elles sont appelées, mais aussi leur paramètre.

```
class UnePersonne{
    String nom;
    int age;
    boolean sexeFeminin;
    UnePersonne conjoint;
    UnePersonne(String n, int a, boolean s){
        age = a;
        nom = n;
        sexeFeminin = s;
    }
    void seMarieA(UnePersonne p) throws TropJeune, PasLibre,
    SoiMeme, MemeSexe{
        if (age<18){
            throw new TropJeune();
```

```
        }
        if (conjoint != null){
            throw new PasLibre();
        }
        if (p == this){
            throw new SoiMeme();
        }
        if (p.sexeFeminin == sexeFeminin){
            throw new MemeSexe();
        }
        conjoint = p;
        if (p.conjoint != this){
            try{
                p.seMarieA(this);
            }catch(TropJeune e){
                conjoint = null;
                throw e;
            }catch(PasLibre e){
                conjoint = null;
                throw e;
            }
        }
    }
    void divorce() throws PasMarie{
        if (conjoint == null){
            throw new PasMarie();
        }
        UnePersonne exConjoint = conjoint;
        conjoint = null;
        if (exConjoint.conjoint == this){
            exConjoint.divorce();
        }
    }
}
class TropJeune extends Exception{}
class PasLibre extends Exception{}
class SoiMeme extends Exception{}
class MemeSexe extends Exception{}
class PasMarie extends Exception{}
class Exo17_2{
    static void essayeDeMarier(UnePersonne p1, UnePersonne p2){
        try{
            p1.seMarieA(p2);
            Terminal.ecrireStringln(p1.nom + "␣est␣marie␣a␣" + p2.nom);
        }catch(TropJeune ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣et␣");
            Terminal.ecrireStringln(p2.nom + "␣echoue");
            Terminal.ecrireStringln("Un␣des␣deux␣est␣trop␣jeune");
        }catch(PasLibre ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣et␣");
            Terminal.ecrireStringln(p2.nom + "␣echoue");
            Terminal.ecrireStringln("Un␣des␣deux␣est␣deja␣marie");
        }catch(SoiMeme ex){
```

```
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣avec␣");
            Terminal.ecrireStringln("soi-meme␣echoue");
        }catch(MemeSexe ex){
            Terminal.ecrireString("le␣mariage␣de␣" + p1.nom + "␣et␣");
            Terminal.ecrireStringln(p2.nom + "␣echoue");
            Terminal.ecrireStringln("Ils␣ou␣elles␣sont␣du␣meme␣sexe");
        }
    }
    static void essayeDeDivorcer(UnePersonne p1){
        try{
            p1.divorce();
            Terminal.ecrireStringln(p1.nom + "␣divorce");
        }catch(PasMarie ex){
            Terminal.ecrireString(p1.nom + "␣n'est␣pas␣marié(e)␣");
            Terminal.ecrireStringln("␣divorce␣impossible.");
        }
    }
    public static void main(String[] args){
        UnePersonne caroline = new UnePersonne("Caroline",41,true);
        UnePersonne paul = new UnePersonne("Paul",35,false);
        UnePersonne rosaria = new UnePersonne("Rosaria",25,true);
        UnePersonne virginie = new UnePersonne("Virginie",37,true);
        UnePersonne roel = new UnePersonne("Roel",37,false);
        essayeDeMarier(paul,virginie);
        essayeDeMarier(paul,rosaria);
        essayeDeMarier(rosaria,caroline);
        essayeDeDivorcer(paul);
        essayeDeMarier(paul,rosaria);
        essayeDeMarier(caroline,caroline);
    }
}
```