Héritage et récursivité
Pièces de quincaillerie

Eléments de correction

Nous allons nous attacher à quelques aspects de la réalisation d'un logiciel de gestion des pièces pour un fabricant de quincaillerie. Dans un premier temps, nous étudions la représentation des pièces.

1 Description des pièces

Trois sortes de pièces se distinguent essentiellement, les pièces de base, les pièces composites en kit et les pièces composites assemblées.
Les *pièces de base* correspondent à des éléments de quincaillerie simples (vis, clou, rayon de roue, chambre à air, etc ...). Elles sont décrites par une référence de préfixe `00`, un prix, une durée de garantie (en mois), une durée de fabrication (en jours). On doit pouvoir éditer une fiche caractéristique sous le format suivant (les données propres à la pièce apparaissent en caractères italiques) :

`nom :` *vis*
`référence :` *007152*
`prix :` *0.01* €
`garantie :` *12* `mois`
`duréee de fabrication :` *1* `jour(s)`

Les *pièces composites* correspondent à des éléments de quincaillerie construits à partir d'autres éléments, simples ou eux-mêmes composites.
Les pièces composites *en kit* sont livrées en pièces détachées avec une notice de montage. Elles se caractérisent par une référence de préfixe `01` et une durée moyenne de montage par un particulier (en minutes). Leur prix se calcule en prenant la somme des prix de leurs composants. Leur durée de garantie s'obtient en prenant la plus courte duréee de garantie parmi celles de leurs composants et en la divisant par deux (on ne fait pas confiance aux montages effectués par les particuliers). Leur durée de fabrication s'obtient en prenant la durée de fabrication la plus longue d'un composant. La fiche caractéristique a la forme suivante :

`nom :` *roue de brouette en kit*
`référence :` *011512*
`prix :` *24* €
`garantie :` *10* `mois`
`duréee de fabrication :` *4* `jour(s)`
`durée de montage particulier :` *15* `mn`
`composants :`
pneu - 004741
chambre a air - 004565
jante - 014541
disque de jante - 001214
rayon - 004748
rayon - 004748
rayon - 004748

Les pièces composites *assemblées* se caractérisent par une référence de préfixe `02`, un prix de montage et une durée de montage en atelier (en jours). Leur prix se calcule en prenant la somme des prix de leurs composants à laquelle on ajoute le prix de montage. Leur durée de garantie s'obtient en prenant la plus courte duréee de garantie parmi celles de leurs composants et en lui ajoutant un bonus de garantie de 6 mois. Leur durée de fabrication s'obtient en prenant la durée de fabrication la plus longue d'un composant augmentée de la durée de montage en atelier. La fiche caractéristique a la forme suivante :

`nom :` *roue de brouette*
`référence :` *021512*
`prix :` *175* €
`garantie :` *26* `mois`
`durée de fabrication :` *5* `jour(s)`
`durée de montage atelier :` *1* `jour(s)`
`prix du montage :` *15* €
`composants :`
pneu - 004741
chambre a air - 004565
jante - 024541
disque de jante - 001214
rayon - 004748
rayon - 004748
rayon - 004748

2 Mise en œuvre

Question 1. Sur la base des informations précédentes, proposez un diagramme de classes UML pour décrire les pièces.

Réponse question 1.

La figure 1 vous donne un schéma UML indicatif pour les pièces de quincaillerie. Nous l'utiliserons dans la suite. L'attribut `referenceDeBase` sera appelé `suffixe` dans la suite.

Question 2. Ecrivez des classes Java très simplifiées correspondant à cette hiérarchie, en ne faisant figurer que les attributs (privés) et les accesseurs.

Réponse question 2.

```
abstract public class Piece
{
    // ---------- Attributs ----------------------
    private String nom="",  suffixe="";
    // -------- Accesseurs ---------------------
    public String getNom(){return nom;}
    public void setNom(String n){this.nom=n;}
    public String getSuffixe(){return suffixe;}
    public void setSuffixe(String r){this.suffixe=r;}

    abstract String prefixe();
    public String reference() {return this.prefixe()+this.getSuffixe();}
}
```

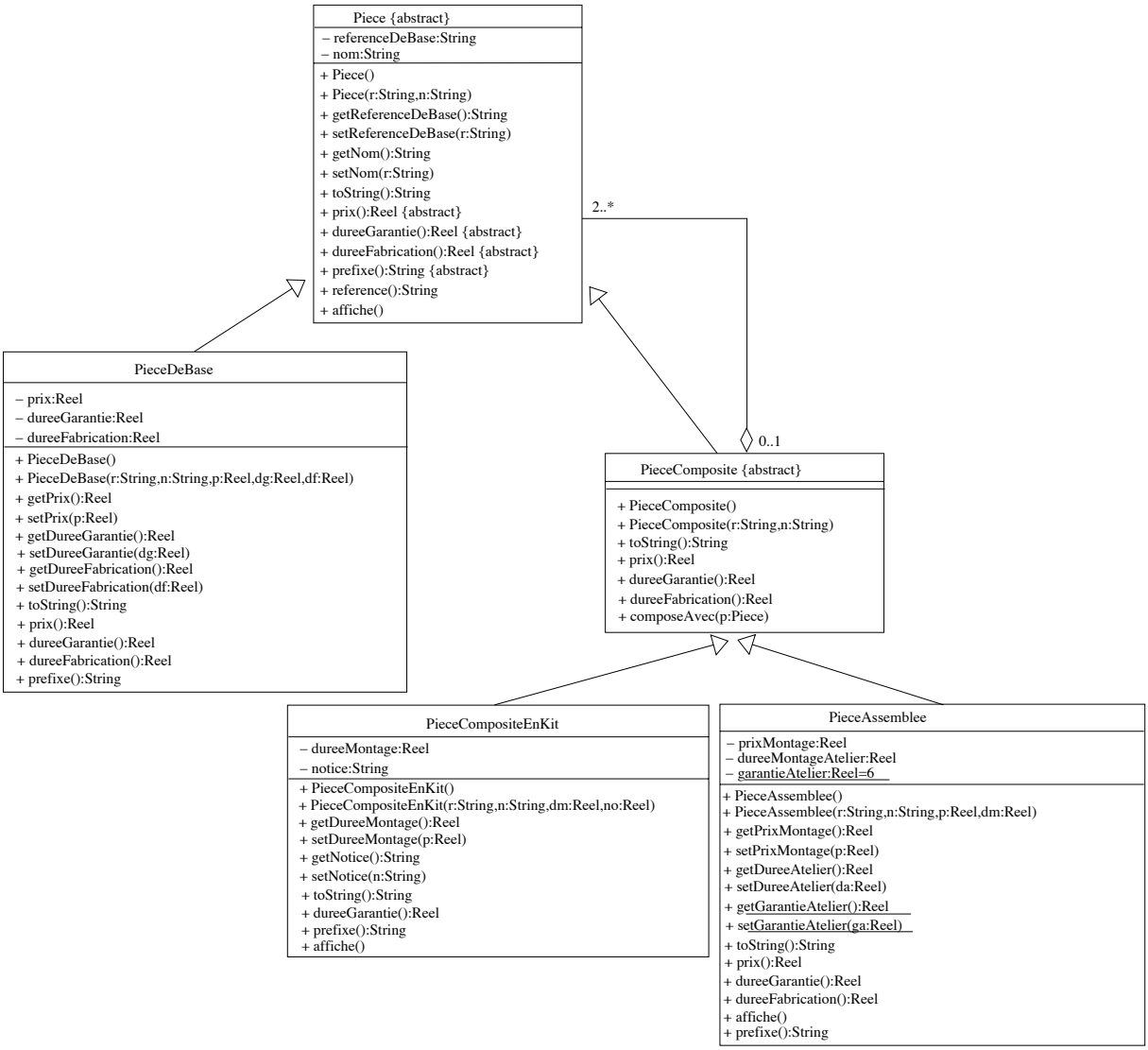

FIGURE 1 – Schéma indicatif de la hiérarchie d'héritage des pièces

```
    public class PieceDeBase extends Piece
{
    //-------- Attributs --------------------------
    private double prix, dureeGarantie, dureeFabrication;
    private final static String prefixePieceDeBase = "00";
    //-------- Accesseurs --------------------------
    public double getPrix(){return prix;}
    public void setPrix(double p){prix=p;}
    public double getDureeGarantie(){return dureeGarantie;}
    public void setDureeGarantie(double d){dureeGarantie=d;}
    public double getDureeFabrication(){return dureeFabrication;}
    public void set(double d){dureeFabrication=d;}
    public String prefixe(){return PieceDeBase.prefixePieceDeBase;}
}

abstract public class PieceComposite extends Piece
{
```

```
    //-------- Attributs
    private Vector<Piece> listePieces;
    //-------- pas d'accesseur !
}

public class PieceCompositeKit extends PieceComposite
{
    //-------- Attributs
    private double dureeMontage;
    private String notice="";
    private final static String prefixePieceCompositeKit = "01";
    //-------- Accesseurs
    public double getDureeMontage(){return dureeMontage;}
    public void setDureeMontage(double duMontage){this.dureeMontage = duMontage;}
    public String getNotice(){return notice;}
    public void setNotice(String n){this.notice=n;}
    public String prefixe(){return PieceCompositeKit.prefixePieceCompositeKit;}
}

public class PieceCompositeAssemblee extends PieceComposite
{
    //-------- Attributs
    private double prixMontage=0,  dureeMontageAtelier=0;
    private  static double garantieAtelier = 6;
    private final static String prefixePieceCompositeAssemblee = "02";
    //-------- Accesseurs
    public double getPrixMontage(){return prixMontage;}
    public void setPrixMontage(double p){prixMontage=p;}
    public double getDureeMontageAtelier(){return dureeMontageAtelier;}
    public void setDureeMontageAtelier(double d){dureeMontageAtelier=d;}
    public static double getGarantieAtelier(){return garantieAtelier;}
    public static void setGarantieAtelier(double g){garantieAtelier=g;}
    public String prefixe() {return
      PieceCompositeAssemblee.prefixePieceCompositeAssemblee;}
}
```

Question 3. Ajoutez les constructeurs et des méthodes String toString() retournant une chaîne de caractères décrivant succinctement les objets.

Réponse question 3.

```
 //-------- Dans  Piece
    public Piece(){}
    public Piece(String n, String suff) {this.nom = n; this.suffixe = suff;}
    public String toString(){return getNom()+" "+reference()+" "+prix();}

 //-------- Dans  PieceDeBase
    public PieceDeBase(){}
    public PieceDeBase(String nom, String suff, double pr, double dg, double df)
    {super(nom, suff); this.prix=pr; this.dureeGarantie=dg; this.dureeFabrication=df;}
    public String toString(){return super.toString()+"-de base-";}

 //-------- Dans  PieceComposite
```

```
    public PieceComposite(){listePieces = new Vector<Piece>();}
    public PieceComposite(String nom, String suff)
    {super(nom, suff); listePieces = new Vector<Piece>();}
    public String toString(){return super.toString()+"-composite-";}

 //-------- Dans  PieceCompositeKit
    public PieceCompositeKit() {}
    public PieceCompositeKit(String nom, String suff, double duMontage, String n)
    {super(nom, suff);  this.dureeMontage = duMontage; this.notice = n;}
    public String toString()
    {return super.toString()+"kit, se monte en "+getDureeMontage()+" mn";}

//-------- Dans PieceCompositeAssemblee
    public PieceCompositeAssemblee(){}
    public PieceCompositeAssemblee(String nom, String suff,
                                   float prMontage, float duMontage)
    {super(nom, suff);  prixMontage = prMontage; dureeMontageAtelier = duMontage;}
    public String toString()
    {return super.toString()+" montee, pour "+getPrixMontage()+" frs";}
```

Question 4. Munissez les pièces composites d'une méthode `void ajoute(Piece p)` permettant de leur ajouter une pièce.

Réponse question 4.

```
  //-------- Dans  PieceComposite
public void ajoute(Piece p){listePieces.add(p);}
```

Question 5. Ajoutez la méthode `double prix()`, qui retourne le prix d'une pièce quelconque.

Réponse question 5.

```
 //-------- Dans  Piece
     abstract public double prix();

 //-------- Dans  PieceDeBase
     public double prix(){return prix;}

 //-------- Dans  PieceComposite
     public double prix()
    {
   double p=0;
   for (int i=0; i<listePieces.size(); i++)
    {
   p = p + listePieces.get(i).prix();
    }
    return p;
    }

 //-------- Dans  PieceCompositeKit
 // rien car la méthode héritée convient
```

```
//-------- Dans PieceCompositeAssemblee
public double prix(){return super.prix() + getPrixMontage();}
```

Question 6. Ajoutez la méthode `double dureeGarantie()`, qui retourne la durée de garantie d'une pièce quelconque.

Réponse question 6.

```
 //-------- Dans  Piece
 abstract public double dureeGarantie();

 //-------- Dans  PieceDeBase
 public double dureeGarantie(){return dureeGarantie;}

 //-------- Dans  PieceComposite
 public double dureeGarantie()
{
if (listePieces.isEmpty())
        return 0;
else
    {
             double min = listePieces.firstElement().dureeGarantie();
             for (int i=1; i<listePieces.size(); i++)
             {
                    double dmc = listePieces.get(i).dureeGarantie();
                     if (min > dmc)
                         min = dmc;
             }
            return min;
    }
}

 //-------- Dans  PieceCompositeKit
 public double dureeGarantie(){return super.dureeGarantie()/2;}

//-------- Dans PieceCompositeAssemblee
public double dureeGarantie()
    {
return super.dureeGarantie() + PieceCompositeAssemblee.getGarantieAtelier();
    }
```

Question 7. Ajoutez la méthode `double dureeFabrication()`, qui retourne la durée de fabrication d'une pièce quelconque.

Réponse question 7.

```
 //-------- Dans  Piece
 abstract public double dureeFabrication();

 //-------- Dans  PieceDeBase
```

```
 public double dureeFabrication(){return dureeFabrication;}

 //-------- Dans  PieceComposite
public double dureeFabrication()
{
  if (listePieces.isEmpty())
     return 0;
  else
   {
      double max = listePieces.firstElement().dureeFabrication();
      for (int i=1; i<listePieces.size(); i++)
        {
          double dfc = listePieces.get(i).dureeFabrication();
          if (max < dfc)
          max = dfc;
        }
     return max;
   }
}

 //-------- Dans  PieceCompositeKit
 // rien car la méthode héritée convient

//-------- Dans PieceCompositeAssemblee
public double dureeFabrication()
{return super.dureeFabrication() + getDureeMontageAtelier();}
```

Question 8. Ajoutez la méthode `void affiche()` qui imprime la fiche caractéristique d'une pièce quelconque en respectant strictement le format indiqué dans l'énoncé.

Réponse question 8.

```
 //-------- Dans  Piece
 public void affiche()
{
   System.out.println("nom : "+getNom()+"\n"+
      "reference : "+reference()+"\n"+
      "prix : "+prix()+" euros \n"+
      "garantie : "+dureeGarantie()+" mois \n"+
      "duree de fabrication : "+dureeFabrication()+
      " jour(s)");
}

protected String afficheCommeComposant(int decalage)
{
   String Decale="";
   for (int i=0; i<decalage; i++)
         Decale += "   ";
   return Decale+getNom()+" - "+reference()+"\n";
}

 //-------- Dans  PieceDeBase
 // rien car la méthode héritée convient
```

```
 //-------- Dans  PieceComposite
 protected String editeComposants(int decalage)
{
  String S="";
  for (int i=0; i<listePieces.size(); i++)
      {
         Piece pc = listePieces.get(i);
         S += pc.afficheCommeComposant(decalage);
         if (pc instanceof PieceComposite)
            S += ((PieceComposite)pc).editeComposants(decalage+1);
      }
  return S;
}

 //-------- Dans  PieceCompositeKit
 public void affiche()
{
super.affiche();
System.out.println("duree de montage particulier : "+getDureeMontage()+" mn\n"+
     "composants : \n"+editeComposants(1));
}

//-------- Dans PieceCompositeAssemblee
public void affiche()
{
super.affiche();
System.out.println("duree de montage atelier : "+
     getDureeMontageAtelier()+" jour(s) \n"+
     "prix du montage : "+getPrixMontage()+" frs \n"+
     "composants : \n"+editeComposants(1));
}
```

Question 9. Créez une petite application qui montre le fonctionnement de vos classes.
Réponse question 9.

```
 public class ProgrammePieces
{
  public static void main(String[] args)
  {
    PieceDeBase pneu = new PieceDeBase("pneu","4741", 95, 6, 1);
    PieceDeBase chambreAair = new PieceDeBase("chambre a air","4565", 45, 6, 1);
    PieceDeBase disqueDejante = new PieceDeBase("disque de jante", "1214", 20, 6, 1);
    PieceDeBase rayon1 = new PieceDeBase("rayon", "4748", 5, 5, 1);
    PieceDeBase rayon2 = new PieceDeBase("rayon", "4748", 5, 5, 1);
    PieceDeBase rayon3 = new PieceDeBase("rayon", "4748", 5, 5, 1);
    PieceCompositeKit jante = new PieceCompositeKit("jante", "4541",15,"blabla");
    jante.ajoute(disqueDejante);
    jante.ajoute(rayon1);
    jante.ajoute(rayon2);
    jante.ajoute(rayon3);
    PieceCompositeKit roue =
                new PieceCompositeKit("roue de brouette en kit","1512", 20, "blabla");
```

```
    roue.ajoute(pneu);
    roue.ajoute(jante);
    roue.ajoute(chambreAair);
    roue.affiche();

    PieceDeBase pneu2 = new PieceDeBase("pneu","4741", 95, 6, 1);
    PieceDeBase chambreAair2 = new PieceDeBase("chambre a air","4565", 45, 6, 1);
    PieceDeBase disqueDejante2 = new PieceDeBase("disque de jante", "1214", 20, 6, 1);
    PieceDeBase rayon12 = new PieceDeBase("rayon", "4748", 5, 5, 1);
    PieceDeBase rayon22 = new PieceDeBase("rayon", "4748", 5, 5, 1);
    PieceDeBase rayon32 = new PieceDeBase("rayon", "4748", 5, 5, 1);
    PieceCompositeAssemblee jante2 = new PieceCompositeAssemblee("jante", "4541",15,3);
    jante2.ajoute(disqueDejante2);
    jante2.ajoute(rayon12);
    jante2.ajoute(rayon22);
    jante2.ajoute(rayon32);

    PieceCompositeAssemblee roue2 =
                      new PieceCompositeAssemblee("roue de brouette", "1512", 15, 3);
    roue2.ajoute(chambreAair2);
    roue2.ajoute(jante2);
    roue2.ajoute(pneu2);
    roue2.affiche();
  }
}
```