HAI717I - Sujet supplémentaire

Figures géométriques

Nous vous donnons ci-dessous quelques éléments de classes écrites en Java, et représentant des figures dans le cadre d'un éditeur de dessins.

```
public abstract class Figure
{
  public Figure(){}
  abstract public void affiche();
}

public abstract class FigureSimple extends Figure
{
  private String couleur;
  public FigureSimple(){}
  public FigureSimple(String c){couleur=c;}
  public void affiche()
         {System.out.println("Figure Simple, de couleur"+getCouleur());}
  public String getCouleur() {return couleur;}
  public void setCouleur(String c) {couleur = c;}
}

public class Carre extends FigureSimple
{
  private float cote, xHautGauche, yHautGauche;
  public Carre() {}
  public Carre(String coul, float cot, float x, float y)
         {super(coul); cote=cot; xHautGauche=x; yHautGauche=y;}
  public void affiche()  {System.out.println("Carre "+getCouleur());}
  public float getCote(){return cote;}
  public void setCote(float c){cote = c;}
  public float getXHautGauche(){return xHautGauche;}
  public void setXHautGauche(float x){xHautGauche = x;}
  public float getYHautGauche(){return yHautGauche;}
  public void setYHautGauche(float y){yHautGauche = y;}
}

public class Cercle extends FigureSimple
{
  private float rayon, xcentre, ycentre;
  public Cercle() {}
  public Cercle(String coul, float r, float x, float y)
         {super(coul); rayon=r; xcentre=x; ycentre=y;}
  public void affiche()  {System.out.println("Cercle "+getCouleur());}
  public float getRayon(){return rayon;}
  public void setRayon(float r){rayon = r;}
  public float getXcentre(){return xcentre;}
  public void setXcentre(float x){xcentre = x;}
  public float getYcentre(){return ycentre;}
  public void setYcentre(float y){ycentre = y;}
}
```

Question 1. Ecrivez dans un main les trois instructions nécessaires pour :
- créer un cercle de rayon 3, de centre (1, 2), et de couleur rouge,
- afficher ce cercle à l'aide de la méthode `affiche`,
- changer la valeur du rayon en 7.

Question 2. Ajoutez des méthodes `toString` aux classes ci-dessus.

Question 3. Ecrivez une classe Java représentant les figures complexes, c'est-à-dire composées d'autres figures, et ne contenant que :

- le (ou les) attribut(s) qui vous semblent fondamentaux,
- un constructeur qui initialise une figure complexe vide de composants,
- une méthode `composeAvec` dont le rôle est d'ajouter une figure passée en paramètre comme composant de la figure complexe receveur de la méthode,
- une méthode `affiche` très simple, qui affiche la chaîne `"Figure complexe"`.

Soient :

- `fg1` une figure complexe composée d'un cercle rouge et d'un carré bleu
- `fg2` une figure complexe composée d'un carré vert, d'un cercle bleu et de `fg1`

Ecrivez les instructions du main qui permettent de créer `fg1` et `fg2`.

Question 4. Ajoutez à la classe `FigureComplexe` une méthode qui retourne vrai si tous les composants (au premier niveau de l'arbre de composition) d'une figure complexe (receveur de la méthode) sont des figures simples, faux sinon.
Pour cela, il y a deux méthodes : (1) soit vous utilisez l'opérateur `instanceof`, qui fonctionne de la manière suivante : `f instanceof FigureSimple` retourne vrai si f référence une instance de `FigureSimple`, faux sinon (2) soit vous ajoutez aux figures une méthode permettant de savoir si elles sont simples ou complexes.

Question 5. Ajoutez à la classe `FigureComplexe` une méthode permettant d'afficher tous ses composants **simples**, en descendant jusqu'au niveau le plus profond de composition. Dans un premier temps vous préoccupez pas d'indenter l'affichage, on ne cherche pas à montrer les niveaux de profondeur des composants simples dans l'arbre de composition de la figure. Puis ajoutez une indentation.
Par exemple, le résultat de l'instruction `fg2.affiche()` sans indentation sera le suivant :

```
Carre vert
Cercle bleu
Cercle rouge
Carre bleu
```

Question 6. Ajoutez à la classe `FigureComplexe` une méthode qui retourne le nombre de composants simples à tous les niveaux de composition (soit 2 pour la figure `fg1` et 4 pour la figure `fg2`).

Question 7. Ajoutez à la classe `FigureComplexe` une méthode
`void construitListeString c, ArrayList<...> v)` dont le rôle est de placer dans la liste `v` toutes les figures simples de couleur `c` qui entrent dans la composition du receveur de la méthode. Comme pour l'affichage, on recherchera les figures simples à tous les niveaux de composition du receveur.
Par exemple, si `v` est un vecteur vide, après l'appel `fg2.construitVecteur(''bleu'', v)`, la liste `v` construite contiendrait le carré bleu et le cercle bleu.

Question 8. Ajoutez à la classe `FigureComplexe` une méthode `FigureComplexe extrait(String c)` qui retourne une figure complexe composée de toutes les figures simples de couleur `c` dont le receveur de la méthode est constitué.
Par exemple, pour la figure `fg2` précédente, la figure complexe retournée serait composée du carré bleu et du cercle bleu.
Vous pouvez utiliser la méthode de la question précédente pour écrire le code de celle-ci.