Introduction à la programmation

5 Composants graphiques - suite

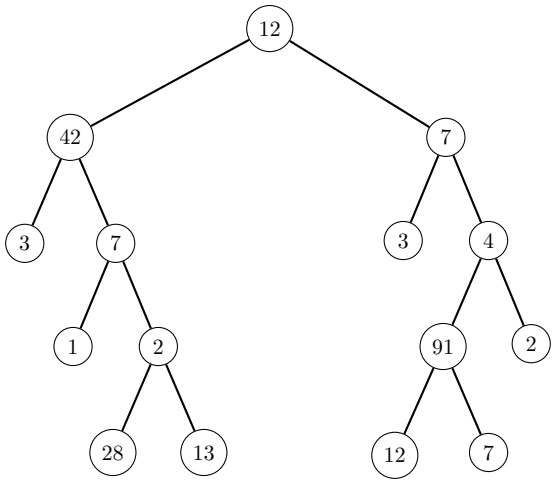
Le but de ce TP est de faire un composant graphique qui affiche un arbre binaire dont chaque nœud contient des entiers.

5.1 Les arbres binaires

Un arbre binaire est une structure de donnée du même ordre que celle de la liste chaînée. Le différence est que chaque élément de l'arbre contient deux éléments suivants : un fils gauche et un fils droit :

```
public class Noeud {
  int elt;
  Noeud filsGauche;
  Noeud filsDroit;
}
```

Voici un exemple d'arbre binaire :



Si un nœud n'a pas de fils gauche (respectivement de fils droit), son pointeur vers le fils gauche (respectivement vers fils droit) est `null`. Le premier nœud d'un arbre s'appelle la **racine**. Un nœud qui n'a pas de fils s'appelle une **feuille**. Chaque nœud d'un arbre est la racine de ce que l'on appelle un **sous-arbre** de l'arbre principale. Pour un nœud donné, on appelle **sous-arbre gauche** le sous-arbre dont la racine est le fils gauche de ce nœud, et **sous-arbre droit** le sous-arbre dont la racine est le fils droit de ce nœud.

5.2 Question 1

Récupérez le fichier d'arbre binaire `BinaryTreeConsole.java` disponible sur Eprel. Ce fichier contient en particulier une fonction d'affichage d'un arbre binaire pour la console. La méthode utilisée pour afficher un arbre binaire en mode console est complexe. L'affichage en mode graphique sera plus simple, grâce notamment à la possibilité d'afficher des nœuds aux coordonnées que l'on veut (contrairement au mode console où ce que l'on affiche est forcément là où se trouve le curseur).

Dans un `main`, créez quelques arbres et affichez-les avec cette fonction, afin de prendre en main la structure.

5.3 Question 2

Créez une classe `PanelTree` dont le but sera d'afficher un arbre binaire. Commencez par faire une méthode :

```
public void drawNode(Graphics g, int x, int y, String text);
```

dont le but sera d'afficher un nœud de l'arbre à la position `x` et `y`, contenant le texte `text`. La taille de votre nœud devra dépendre de la taille du texte. Pour ce faire, vous pouvez initialiser la font à utiliser dans la constructeur de `PanelTree` ainsi qu'un attribut `fm` de type `FontMetrics` qui vous permettra de connaître la taille en pixel d'une chaîne de caractère.

```
setFont(new Font("Helvetica",Font.PLAIN,12)); //utilisez une autre font
    si vous voulez
fm = getFontMetrics(font);
```

Pour connaître la taille en pixel d'une chaîne de caractère, utilisez

```
fm.stringWidth("chaine");
```

Surchargez à présent la méthode `paintComponent` de votre classe `PanelTree` pour appeler votre méthode `drawNode` avec plusieurs paramètres, pour tester.

5.4 Question 3

Le but est ici de créer une structure de données contenant les informations d'affichage de chaque noeud d'un arbre. Créez pour cela une classe :

```
class NodeDisplay {
  int posx;
  int posy;
  int val;

  NodeDisplay filsGauche;
  NodeDisplay filsDroit;
}
```

Le but de cette classe est de faire une duplication de votre structure d'arbre initiale, en rajoutant en plus les informations d'affichage de chaque nœud : une position en x et

une position en y. L'idée générale est simple :

1. Soit x_n la position en x d'un nœud.
2. Soit s_g la largeur nécessaire pour afficher le sous-arbre droit du sous-arbre gauche de ce nœud.
3. alors le fils gauche de ce nœud doit être affiché en position $x_n - s_g$.
4. Soit s_d la largeur nécessaire pour afficher le sous-arbre gauche du sous-arbre droit de ce nœud.
5. alors le fils droit de ce nœud doit être affiché en position $x_n + s_d$.

La première étape sera donc de calculer la largeur nécessaire pour afficher un arbre. C'est un calcul qui est naturellement récursif : La largeur nécessaire pour afficher un arbre est égale à la largeur nécessaire pour afficher sa racine, plus la largeur nécessaire pour afficher son sous-arbre gauche, plus la largeur nécessaire pour afficher son sous-arbre droit.

Dans votre classe `Node`, rajoutez une fonction `int countPlaceGraphique` qui calcule la largeur nécessaire pour afficher l'arbre dont la racine est le nœud courant. Votre fonction devra prendre divers paramètres permettant de calculer cette largeur (notamment un `FontMetrics` pour connaître la largeur d'un nœud, mais aussi des paramètres d'espacement entre les nœuds). Dans votre classe `PanelTree`, faites ensuite une fonction :

```
NodeDisplay getDisplayInfo(Node tree);
```

Dont le but sera de faire une "copie" de l'arbre `tree`, tout en calculant les positions d'affichage en x et en y de chaque nœud.

5.5 Question 4

Modifiez votre méthode `paintComponent` pour qu'elle affiche un arbre qui sera passé en paramètre dans le constructeur de `PanelTree`.