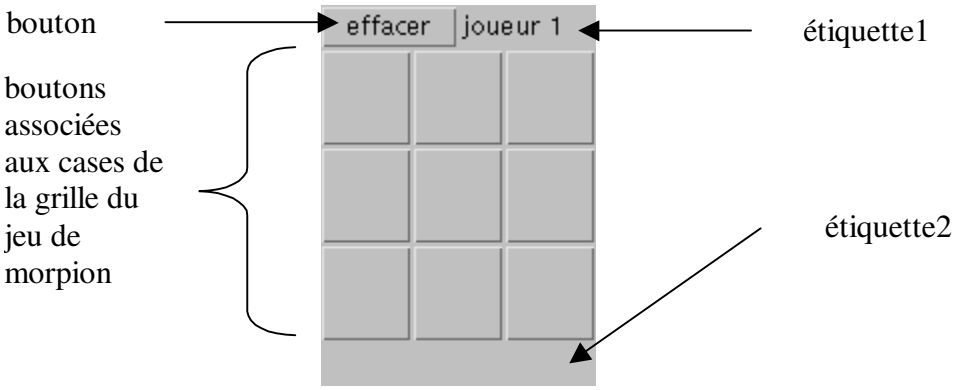
## TP – Rétro-conception du patron Composite

### 1.1 Un petit exemple

On veut implanter la classe *MorpionGraphique* correspondant à l'interface graphique ci-dessous :




Le code de la classe *MorpionGraphique* devrait être plus ou moins le suivant :

```
/**
* Cette classe permet de construire et manipuler un jeu de morpion graphique
*/
import java.awt.* ;
import java.awt.event.* ;
public class MorpionGraphique extends Panel
{
    private Label noJoueur ;      // étiquette fournissant le numéro du joueur
    private Label resultat ;      // étiquette fournissant le n° du joueur gagnant
    private Button tableauBoutons [][] ; // tableau des boutons de la grille
    private Button effacer ;      // bouton d'effacement d'une partie

    /**
    * construit une grille de morpion graphique
    */
    public MorpionGraphique ()
    {
        // construire le panneau associé à la grille des cases
        Panel cases = new Panel () ;
        cases.setLayout (new GridLayout (3, 3)) ;
        // créer le tableau des boutons
        this.tableauBoutons = new Button [3][3] ;
        // créer les boutons de la grille
        for (int i=0 ; i<3 ; i++)
        {
            for (int j=0 ; j<3 ; j++)
            {
                tableauBoutons[i][j] = new Button ("");
                cases.add (this.tableauBoutons[i][j]) ;
            }
        }
        // création du panneau du haut
        Panel haut = new Panel () ;
        haut.setLayout (new GridLayout (1, 2)) ;
        this.effacer = new Button ("effacer") ;
        haut.add (effacer) ;
        this.noJoueur = new Label ("joueur 1") ;
        haut.add (this.noJoueur) ;
        // agencement des composants dans le panneau
        this.setLayout (new BorderLayout ()) ;
```

```
        this.add ("North", haut) ;
        this.add ("Center", cases) ;
        this.resultat = new Label () ;
        this.add ("South", this.resultat) ;
    }
}
```

## 1.2 Questions :

- En vous appuyant sur ce programme, dessiner l'arbre de composition représentant l'agencement des différents composants graphiques de l'IHM *MorpionGrpahique.*
    - A chaque nœud correspondra la référence (au sens objet) et le type de l'objet graphique.
    - Ajouter la stratégie de placement à chaque nœud interne de l'arbre (non feuille).
- Qu'est-ce qui différencie la racine et les feuilles des autres nœuds internes de l'arbre de composition ?

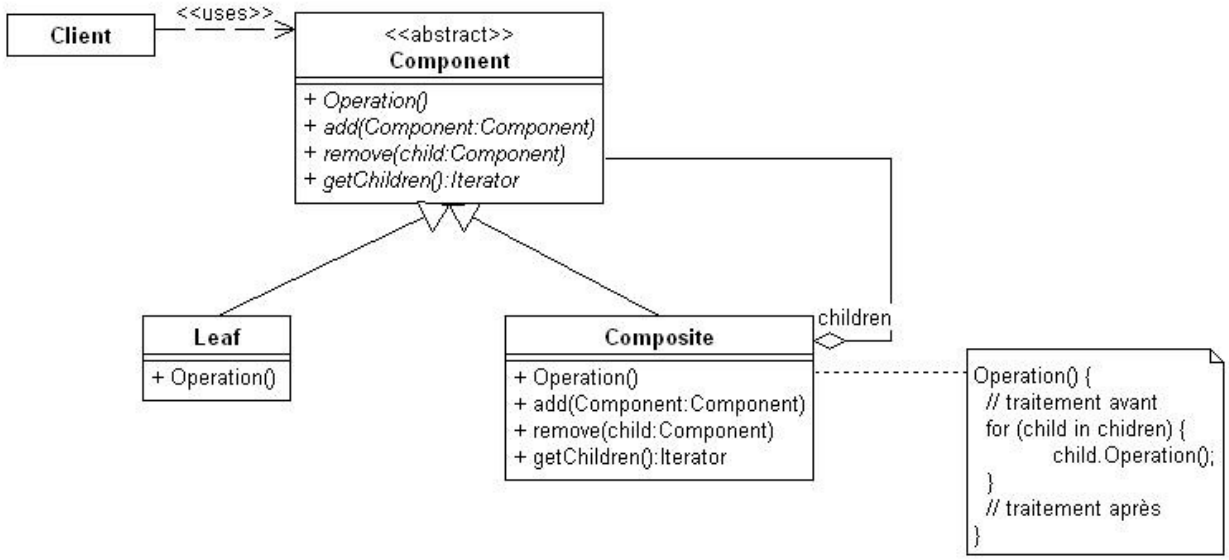
## 1.3 Patron Composite et architecture des classes du paquetage AWT

Soit une partie du tutorial du paquetage (*Abstract Window Toolkit*) fourni par sun (*http://java.sun.com*) traitant de l'agencement et la gestion des composants graphiques :

- Chaque composant graphique est sous-classe de la classe *Component*.
- Une instance de *Button* est un bouton contenant éventuellement une ligne de texte et pouvant réagir à un clic souris.
- Une instance de *Label* sert à afficher une ligne de texte.
- Une instance de *TextField* permet de saisir et visualiser une ligne de texte.
- Une instance de *TextArea* permet de saisir et visualiser plusieurs lignes de texte.
- Un *Canvas* est un composant graphique permettant de tracer différentes formes géométriques.
- Une instance de *Checkbox* est composée d'un *Label* et d'un *Button*.
- Une instance de *Choice* est un agrégat de *CheckBox*.
- Une instance de *List* est une liste d'items dont on peut sélectionner un ou plusieurs items.
- Chaque composant graphique partage plus de 100 méthodes héritées de la classe *Component*. Les plus utilisées sont :
    - *int getHeight()* : renvoie la hauteur d'un composant graphique.
    - *int getWidth()* : renvoie la largeur d'un composant graphique.
    - *int getX()* : renvoie l'abscisse du composant graphique dans le composant *Container* dans lequel il est affiché.
    - *int getY()* : renvoie l'ordonnée du composant graphique dans le composant *Container* dans lequel il est affiché.
    - *Point getLocationOnScreen()* : renvoie les coordonnées d'un composant graphique sous forme de *Point* par rapport au coin en bas à gauche de l'écran.
    - *Rectangle getBounds()* : renvoie le plus petit Rectangle « enveloppant » un composant graphique.
    - *setEnabled(boolean)* : si le composant est actif, il réagit aux entrées utilisateur et apparaît normalement à l'écran.
    - *setVisible(boolean)* : rend le composant graphique visible à l'écran si il appartient à un composant *Container* visible.
    - *setBackGround(Color)* : change la couleur de l'arrière plan d'un composant.
    - *setForeGround(Color)* : change la couleur de premier plan d'un composant.
    - *setFont(Font)* : change la police de caractères associée aux textes d'un composant.

- Un Objet *Container* est aussi un *Component*. Les objets containers peuvent être imbriqués récursivement. La classe Container définit plus de 50 méthodes spécifiques. Les plus couramment utilisées sont :
    - *public Component add(Component comp)* : ajoute le composant à l'objet *Container* et retourne comp.
    - *public void remove(Component comp)* : enlève le composant de l'objet *Container*.
    - *public Component[] getComponents()* : retourne les composants de l'objet *Container*.
- La classe *Panel* est la sous-classe de *Container* la plus souvent utilisée et est en général étendue pour créer de nouvelles interfaces graphiques. Un panel permet de définir un espace pour y rattacher d'autres composants graphiques, y compris d'autres panels.
- La classe *Window* est la racine des classes fenêtres sans bords ni barre de menu.
- La classe *Frame* est la première sous-classe de *Window* possédant un titre et des bords.

## 1.4 Structure du pattern Composite




## 1.5 Question :

- Donner le diagramme UML des différentes classes présentées, en indiquant le rôle des différentes classes dans le modèle de conception Composite.