# AP5: Conteneur racine et Gestionnaires d'agencement

---

**Exercice I: Le conteneur racine JFrame**

Nous allons nous concentrer sur le principal conteneurs de Swing : la JFrame, qui correspond à une fenêtre.

Il y a relativement peu de méthodes à connaitre sur la classe JFrame. Voici les principales méthodes que vous aurez à utiliser :

```
void setContentPane(Container) // Affecte le conteneur de base de la fenêtre
void setVisible(boolean) // Affiche ou masque la fenêtre à l'écran
void pack() // Demande explicitement le calcul de l'agencement des composants et la taille de la fenêtre
void setDefaultCloseOperation(int) // action quand on ferme la fenêtre (Jframe.EXIT_ON_CLOSE la plupart du temps)

Container getContentPane() // Retourne le conteneur de base de la fenêtre
void setSize(int, int) // Fixe la taille de la fenêtre ne pas utiliser en même temps que pack()
```

Faire un programme qui affiche une fenêtre avec le texte de votre choix (ex.: « J'aime AP5! ») dedans. Les exemples du support de cours peuvent vous aider...

---

**Exercice II: Expérimentations sur les principaux gestionnaires d'agencement (*LayoutManager*)**

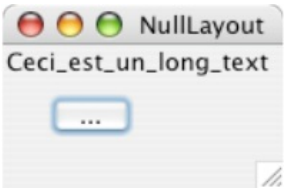
Comme vous le savez, les gestionnaires d'agencement ont pour responsabilité le placement spatial des composants graphiques dans un conteneur. Le but des exercices est de reproduire les interfaces graphiques données en exemples, et bien sûr de comprendre les caractéristiques de chaque gestionnaire.

**Exercice II.a: Le *layout* "Null"**

Comme son nom l'indique, c'est le plus "nul" des gestionnaire d'agencement. Il est quand même présenté ici car il est souvent utilisé par les logiciels de création d'IHM (qui génèrent ensuite automatiquement le code correspondant à l'interface conçue "à la souris").

Pourquoi est-ce le plus "nul" ? Tout simplement parce qu'il consiste à spécifier précisément l'ensemble des positions et des tailles de chacun des composants. Cela donne une interface absolument pas adaptable : lorsque l'on redimensionne la fenêtre les composants ne se réarrangent pas pour utiliser au mieux l'espace disponible.

Exemple graphique:

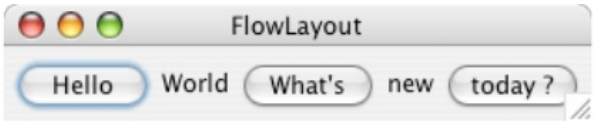



Pour cela, réglez juste le gestionnaire d'agencement à null (c'est-à-dire setLayout(null)) et ensuite intéressez-vous à la méthode setBounds qui permet de spécifier les dimensions d'un composant graphique (setSize pour régler la taille de la fenêtre sera aussi utile !).

Note : Dans certain cas, ce *layout* ne fonctionne tout simplement pas. Il est déconseillé de l'utiliser.

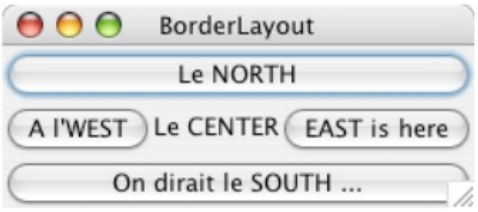
**Exercice II.b: Le *Flowlayout***

Le FlowLayout est le gestionnaire défini par défaut pour le conteneur JPanel. Il aligne les composants les uns à la suite des autres, en respectant leur taille. Si il y a trop de composants, des lignes peuvent être créées par le gestionnaire. Sur chaque ligne, les composants sont soit centrés (par défaut), alignés à droite ou à gauche.



**Exercice II.c: Le *Borderlayout***

Le BorderLayout définit 5 zones correspondant aux 4 points cardinaux (BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST et BorderLayout.WEST) ainsi qu'au centre (BorderLayout.CENTER). Il est donc possible lorsque l'on utilise ce gestionnaire de spécifier la zone dans laquelle le composant doit être ajouté. Ce gestionnaire d'agencement est utilisé par défaut par les conteneurs suivants : JFrame , JWindow et JApplet .




Note : Utilisé pour gérer une barre d'outils « flottante », le contenu est au centre et la barre peut-être déplacée par l'utilisateur sur les bords

**Exercice II.d: Le *Boxlayout***

Ce gestionnaire dispose les composants en ligne ou en colonne, en respectant les dimensions de chacun des composants. Pour obtenir les bons alignements , intéressez-vous à la méthode setAlignmentX/Y.

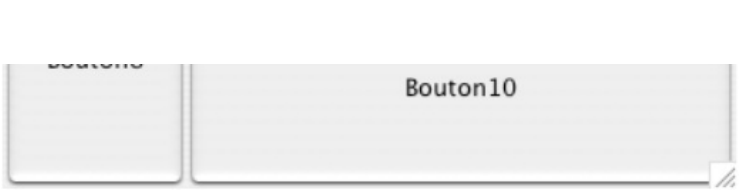

**Exercice II.e: Le *GridBaglayout***

Positionne les élément dans une grille avec un numéro de ligne et de colonnes prédéfinies. En fait le plus important est le numéro de colonnes.

Tous les éléments ont la même taille.

**Exercice II.f: Le *GridBaglayout***

Le plus complexe mais qui offre plus de possibilités. Nécessite d'utiliser des GridBagConstraints pour chaque composant.


**Exercice II.g: Layout « manuel »**

On peut aussi composer la plupart des layouts en imbriquant des JPanels ayant chacun leur propre BoxLayout.

Il peut cependant être difficile de gérer correctement les redimensionnement de fenêtre.