Cours Apl 11 - Introduction à la gestion des écrans

Dyalog Apl vous permet de créer les écrans de 2 manières compatibles :
- à la souris, comme dans la plupart des langages (via le ws wdesign),
- en écrivant des fonctions qui construisent et/ou modifient les écrans. C'est cette voie que nous allons explorer.

* Quelques notions préliminaires

- **Objet** : on appelle objet une fenêtre, un bouton, une zone de liste, ...

- **Propriété** : une propriété défini les caractéristiques d'un objet telles que sa taille, sa couleur, son contenu, son libellé, son nom, ...

- **Evènement** : Il peut arriver diverses choses à un objet du fait de l'utilisateur du programme. Par exemple, dans le cas d'un bouton, il peut être cliqué, enfoncé, relâché, être survolé par la souris, ce qui nous fait déjà 4 évènements.

A quoi servent les évènements ?
Lorsqu'un évènement précis se produit, on veut déclencher une action.
Par exemple, si on clique sur un bouton intitulé "Enregistrer", on lance une fonction qui enregistrera le travail.

- **Méthode** : On peut assimiler une méthode à une fonction de l'objet auquel elle appartient. Par exemple, dans le cas de l'objet Grid (tableur Apl), il existe une méthode AddRow qui permet d'ajouter des lignes.
Une méthode est appelée par un programme alors qu'un évènement résulte généralement d'une action de l'utilisateur.

* Premiers objets et propriétés :

- L'objet de base est la fenêtre : Form
Elle a un certain nombre de propriétés dont le type de l'objet, un libellé, un emplacement et une taille.
Pour créer un objet window, on utilise la fonction système ⎕*WC* (window create).
L'argument gauche est le nom que portera l'objet et l'argument droit contient les propriétés de l'objet.

Maintenant passons à l'action.
Nous allons créer notre première fenêtre qui s'appellera F1.

```
'F1' ⎕WC 'Form' ('Caption' 'Multiplication')('Size' 50 50)
```

Vous trouverez une nouvelle icône dans la barre des tâches. Si vous cliquez dessus, vous verrez apparaître une fenêtre vide, dont le titre sera "Multiplication", occupant la moitié de l'écran.

Ajoutons deux zones de saisie avec leurs libellés à cet écran.
D'abord, le libellé :

```
'F1.L1' ⎕WC 'Label' 'Série de nombres' ('Posn' 10 5)('Size' 0 25)
```

- A chaque fois qu'on crée un objet, on doit respecter la syntaxe suivante :

```
'objet_père.nouvel objet' ⎕WC 'Type d'objet' ('nom_propriété 1'
valeur(s)_propriété 1) ('nom_propriété 2' valeur(s)_propriété 2)
```

- Les propriétés ont toutes un emplacement par défaut. Ainsi, on peut directement indiquer la valeur d'une propriété sans en mentionner le nom, si on la place au bon endroit.

- Type d'objet
Tous les objets ont pour première propriété leur type. C'est pourquoi on l'indique immédiatement après le ⎕WC sans préciser un nom de propriété.
On peut également définir le type ailleurs qu'en 1ère position. Dans ce cas, il faudra écrire un couple ('Type' 'Form') pour une fenêtre.
- 'Label' signifie que F1.L1 est une étiquette.
C'est un objet ne pouvant contenir que du texte et dans lequel on ne peut pas saisir.

- 'Série de nombres' est le texte affiché. Le nom de cette propriété est 'Caption'. Pour un Label c'est toujours la deuxième propriété définie. Aussi, si on la met à cet endroit, il n'est pas obligatoire de mentionner le nom de la propriété définie.

- 'Posn' indique la position du coin supérieur gauche de l'objet par rapport au coin supérieur gauche de l'objet père. Par défaut ces coordonnées sont exprimées en pourcentage de la taille de l'objet père.
On indique toujours les coordonnées en hauteur puis largeur.

- 'Size' défini la taille de l'objet.
Par défaut, elle est également exprimée en pourcentage de la taille de l'objet père
`('Size' 0 25)` signifie qu'on laisse Apl choisir la hauteur la plus appropriée et qu'on force la largeur à 25% de celle de F1.

Ajoutons maintenant le champ de saisie de la série de nombres :

```
'F1.S1' ⎕WC 'Edit' ('Posn' 10 40)('Size' 0 50)
```

Nouveau label et nouveau champ :

```
'F1.L2' ⎕WC 'Label' 'Coefficient : ' ('Posn' 30 5)('Size' 0 25)
'F1.S2' ⎕WC 'Edit' ('Posn' 30 40)('Size' 0 20)
```

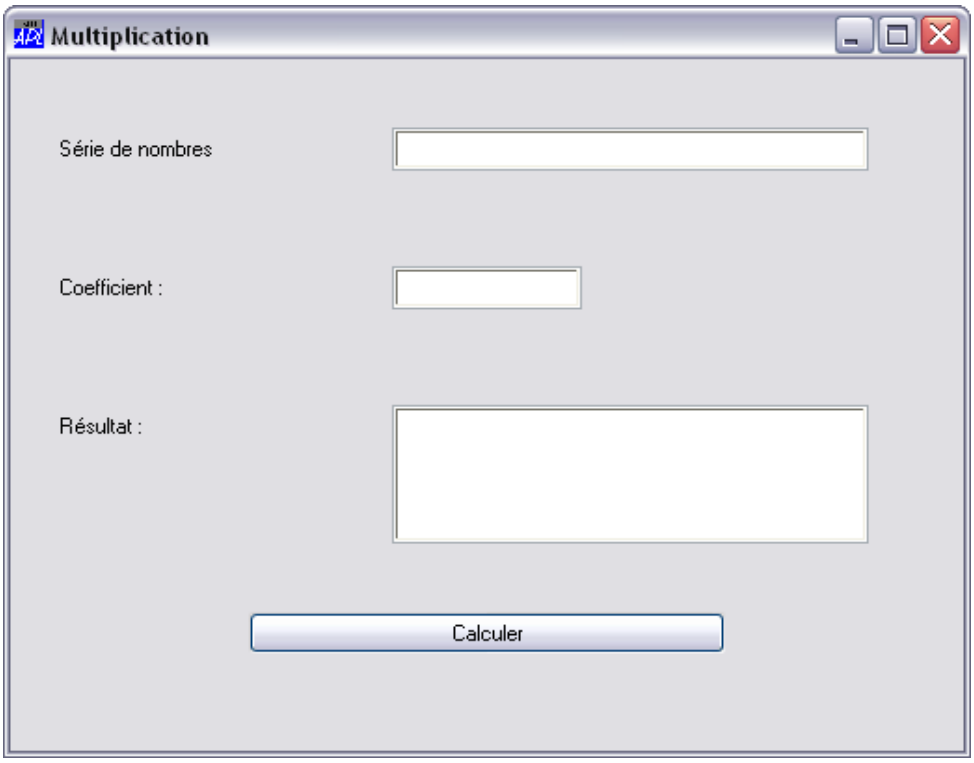
Affichage du résultat du calcul :

```
'F1.L3' ⎕WC 'Label' 'Résultat : ' ('Posn' 50 5)('Size' 0 25)
'F1.S3' ⎕WC 'Edit' ('Posn' 50 40)('Size' 20 50)
```

Bouton de lancement du calcul :

```
'F1.B1' ⎕WC 'Button' 'Calculer' ('Posn' 80 0)('Size' 0 50)
```

Une fois tout ça saisi, vous devez avoir à l'écran votre nouvelle fenêtre :



En entrant ce code, vous avez créé un nouvel objet : F1
Si vous tapez ⎕NL 9, Apl vous répondra :
F1
⎕NL signifie 'Name List'. Il rend la liste des noms correspondant à la ou les classes données en argument droit.
⎕NL 2 rend la liste des variables et ⎕NL 3 la liste des fonctions.
⎕NL 9 rend la liste des namespaces et des objets graphiques.

Pour en savoir plus :
- aide en ligne
- ⎕NC : réciproque de ⎕NL, rend la classe de l'objet nommé en argument droit.

Pour l'instant l'écran existe mais il ne fait rien.
Effacez F1 en tapant :
)erase F1

puis créez la fonction Mulvec, localisez F1 et reprenez les lignes de définition de l'écran dans la fonction.
Si vous lancez la fonction, il ne se passe rien, vous ne voyez rien à l'écran. C'est normal : l'écran est créé, mais comme il est localisé dans la fonction, il est détruit, dès qu'on en sort.
Pour que l'écran soit actif, vous devez "donner la main" à Windows.
C'est la fonction ⎕DQ qui s'en charge.
Elle active l'objet dont le nom est mentionné en argument droit.
Ajoutez cette ligne dans votre fonction et relancez-la :

```
⎕DQ 'F1'
```

Listing de la fonction :

```
Mulvec;F1;CR
 'F1'⎕WC'Form'('Caption' 'Multiplication')('Size' 50 50)
 'F1.L1'⎕WC'Label' 'Série de nombres'('Posn' 10 5)('Size'0 25)
 'F1.S1'⎕WC'Edit'('Posn' 10 40)('Size'0 50)
 'F1.L2'⎕WC'Label' 'Coefficient : '('Posn' 30 5)('Size'0 25)
 'F1.S2' ⎕WC 'Edit' ('Posn' 30 40)('Size' 0 20)
 'F1.L3'⎕WC'Label' 'Résultat : '('Posn' 50 5)('Size'0 25)
 'F1.S3'⎕WC'Edit'('Posn' 50 40)('Size' 20 50)
 'F1.B1'⎕WC'Button' 'Calculer'('Posn' 80 0)('Size'0 50)
 ⎕DQ'F1'
```

L'écran s'affiche, on peut même saisir des valeurs mais pour l'instant aucun calcul ne s'effectue.
Pour quitter votre nouvel écran et revenir à APL, fermez la fenêtre en cliquant sur la croix en haut à droite.

C'est ici qu'interviennent les évènements.
Quand l'évènement "clic sur le bouton calculer" se produit, on veut que la multiplication soit effectuée et que le résultat soit affiché dans F1.S3
On a le choix entre 2 méthodes :
- soit on analyse le résultat du ⎕DQ dans la fonction Mulvec pour voir si le calcul a été demandé
- soit on appelle une fonction de traitement spécifique (fonction callback). Nous allons d'abord explorer cette voie.

Créez la fonction Mulvec_cal
Elle admet comme argument droit le compte rendu de l'événement ayant occasionné son appel. Toute fonction callback possède au moins cet argument car APL l'appelle systématiquement en le lui passant et il est souvent utilisé, comme nous le verrons par la suite.
Entrez ces lignes dans la fonction :

```
Mulvec_cal d
F1.S3.Text←⍕(⍎F1.S1.Text)×(⍎F1.S2.Text)
```

La fonction exécute (⍎), exécute la chaîne de caractères qui lui est passée en argument droit. Dans le cas d'une chaîne de caractères contenant uniquement des numériques, elle rend un ou plusieurs nombres (plusieurs si ils sont séparés par des espaces ou des virgules) et la fonction format (⍕) convertit un ou plusieurs nombres en chaîne de caractères.

Le contenu d'un champ texte correspond à sa propriété 'Text' et ne peut être qu'une chaîne de caractères, même si elle n'est constituée que de chiffres. C'est pourquoi on formatte le résultat du calcul avant de le mettre dans F1.S3.

De même, on ne peut multiplier que des chiffres, c'est pourquoi on numérise le contenu des champs S1 et S2.

Branchons maintenant notre fonction de calcul.
Nous devons donc dire à APL que quand le bouton B1 est actionné, on doit appeler la fonction Mulvec_cal.
Pour ce faire, nous allons modifier la ligne de définition du bouton comme suit :

```
'F1.B1'⎕WC'Button' 'Calculer'('Posn' 80 0)('Size'0 50)('Event' 'Select' 'Mulvec_cal')
```

Event signifie qu'on défini la propriété Evènement.
Select est le nom de l'évènement correspondant au simple click sur un bouton
Mulvec_cal est la fonction callback qui doit être appelée lorsque l'évènement se produit.

Essayez à nouveau votre fonction.

Si les valeurs entrées ne sont pas correctes, la fonction Mulvec_cal va se planter. Pour éviter cela, on va gérer l'erreur avec le `:Trap`.
C'est un traitement conditionnel dont l'idée est : si ça se passe bien, je fais ci, sinon je fais ça.
Nouveau texte de la fonction :

```
Mulvec_cal d
:Trap 0
F1.S3.Text←⍕(⍎F1.S1.Text)×(⍎F1.S2.Text)
:Else
  F1.S3.Text←'Erreur !'
:EndTrap
```

Pour en savoir plus sur le `:Trap` consultez l'aide en ligne. On gère également les erreurs avec le `⎕Trap`.
Réessayez en saisissant des valeurs incorrectes.

Nous allons maintenant voir comment gérer les événements sans utiliser de fonction callback.
Commençons par remplacer la zone de saisie F1.S2 par un objet Spinner.
C'est une zone de saisie avec des flèches permettant d'en augmenter ou diminuer la valeur.
La valeur du coefficient pourra aller de -100 à 100.
Nous allons également demander à Apl de reprendre la main lorsqu'on clique sur une des flèches pour modifier la valeur du spinner afin de recalculer immédiatement le résultat avec le nouveau coefficient.
Remplacez la définition de F1.S2 par :

```
'F1.S2'⎕WC'Spinner'('Posn' 30 40)('Size'⍬ 40)('Event' 'Spin'
1)('Limits' ¯100 100)
```

Remplacez également la dernière ligne de la fonction Mulvec par les suivantes :

```
ET_AFF:CR←⎕DQ 'F1'
 :If 0≠ ⍴CR
   F1.S2.Value←3⊃CR
   Mulvec_cal''
   →ET_AFF
 :EndIf
```

`ET_AFF:` est une étiquette qui indique une ligne de la fonction.
`:If 0≠⍴CR` marque le début d'un bloc d'instructions exécutées uniquement si la taille de CR n'est pas nulle. En effet, si on ferme la fenêtre en cliquant sur la croix en haut à droite, CR reçoit un vecteur vide.
Si le test est concluant, on met le 3ème élément de CR (nouvelle valeur) dans F1.S2, on refait un calcul en appelant Mulvec_cal '' (l'argument n'a aucune importance puisqu'il n'est pas exploité par la fonction), et avec →ET_AFF on redonne la main à Windows.

Dans le respect des règles d'hygiène des développeurs pas trop cochons, ajoutez CR à votre liste de variables locales.

Testez votre fonction.

Au cas où quelque chose ne fonctionnerait pas correctement, le texte complet de la fonction est :

```
Mulvec;F1;CR
 'F1'⎕WC'Form'('Caption' 'Multiplication')('Size' 50 50)
 'F1.L1'⎕WC'Label' 'Série de nombres'('Posn' 10 5)('Size'0 25)
 'F1.S1'⎕WC'Edit'('Posn' 10 40)('Size'0 50)
 'F1.L2'⎕WC'Label' 'Coefficient : '('Posn' 30 5)('Size'0 25)
 'F1.S2'⎕WC'Spinner'('Posn' 30 40)('Size'0 40)('Event' 'Spin'
1)('Limits' ¯100 100)
 'F1.L3'⎕WC'Label' 'Résultat : '('Posn' 50 5)('Size'0 25)
 'F1.S3'⎕WC'Edit'('Posn' 50 40)('Size' 20 50)
 'F1.B1'⎕WC'Button' 'Calculer'('Posn' 80 0)('Size'0 50)('Event'
'Select' 'Mulvec_cal')
ET_AFF:CR←⎕DQ'F1'
 :If 0≠⍴CR
   F1.S2.Value←3⊃CR
   Mulvec_cal''
   →ET_AFF
 :EndIf
```

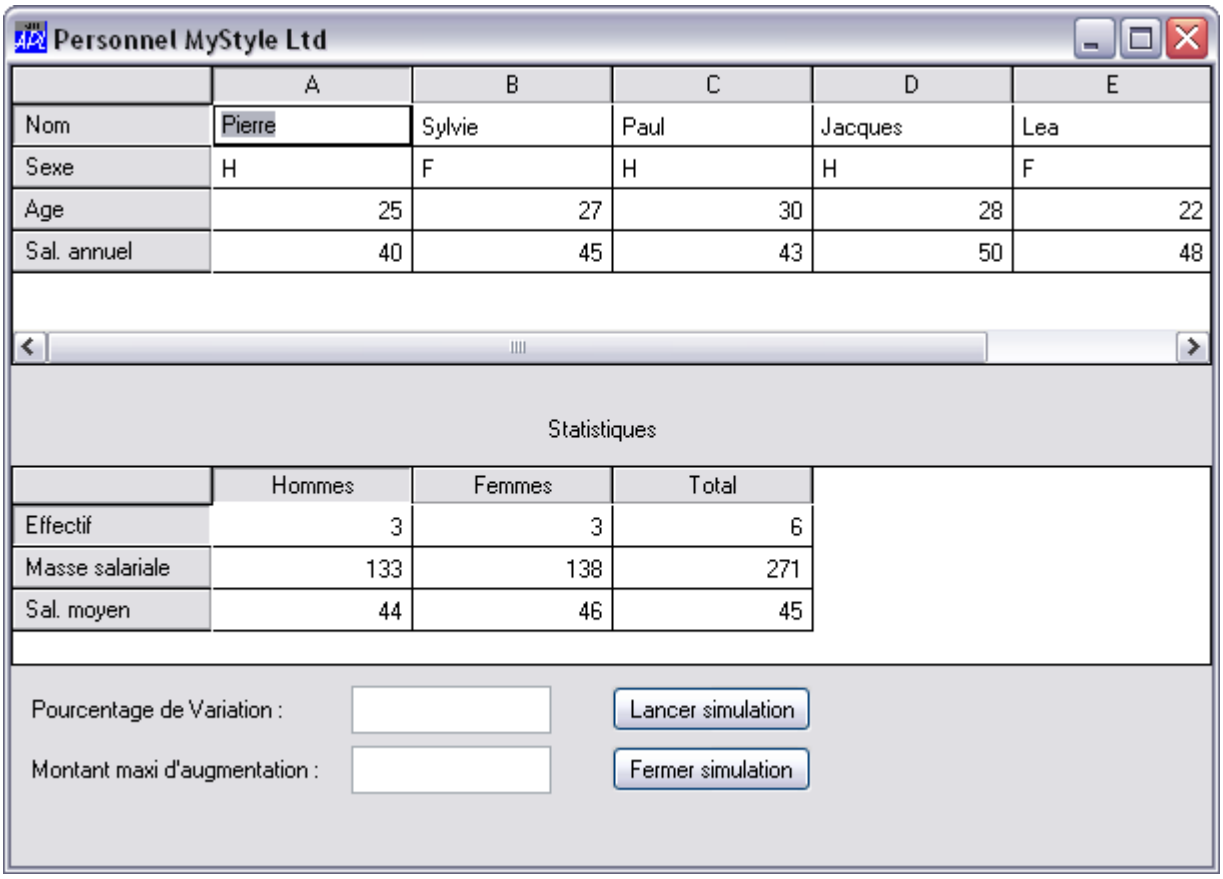
Pour en savoir plus sur les objets graphiques mis à votre disposition dans Dyalog Apl, il existe 2 tutoriels très riches :
- Wtutor
- Wtutor95

Chargez les et explorez tout, c'est simple, efficace et vraiment beau.
Pour les passionnés de graphiques, allez faire un tour du coté du workspace Rain.
Sinon, il y a toujours l'aide en ligne.

Travaux pratiques :

0. Chargez votre Ws de travaux pratiques :
)load c:\Mes documents\pratique-apl

Nous allons continuer à travailler sur la matrice du personnel mais cette fois à travers une interface graphique que nous allons créer comme ci-dessous :



Pour tous les nouveaux objets, l'aide en ligne et les tutoriels sont là.

1. Vos objectifs sont les suivants :
- Créez une fonction "GestPerso" qui va commencer par créer un nouvel écran dont le libellé sera "Personnel MyStyle Ltd"
- Afficher la matrice du personnel dans un Grid avec la possibilité d'en modifier le contenu, de supprimer et ajouter des personnes.
Intitulés de lignes :
    - nom
    - sexe
    - age
    - salaire annuel

- Afficher dans un autre grid en dessous les statistiques suivantes :
    - l'effectif masculin, féminin et total

- la masse salariale masculine, féminine et totale
- les salaires moyens masculin, féminin et total.

Ces données devront être rafraîchies au fur et à mesure des saisies opérées dans l'autre grid.

- Ajoutez 2 boutons en bas de l'écran :
    - Sauver et quitter : Met à jour la variable Perso et ferme la fonction, donc l'écran puisque vous l'aurez localisé dans la fonction.
    - Quitter sans sauver.

2. Ajoutez 2 nouvelles zones de saisie numériques avec leurs intitulés :
- Pourcentage de variation
- Montant maxi d'augmentation

- Ajoutez un bouton "Lancer Simulation", qui modifie tous les éléments en fonction du % de variation et du plafond (dont la saisie est facultative).
Attention, les données de la simulation ne peuvent pas être sauvegardée.

- Ajoutez un bouton "Fermer Simulation", qui remet les données réelles à l'écran.

Solutions :

1. Vos objectifs sont les suivants :
- Créez une fonction "GestPerso" qui va commencer par créer un nouvel écran dont le libellé sera "Personnel MyStyle Ltd"

```
 GestPerso;F1
⍝ Gestion du personnel
 'F1'⎕WC'Form' 'Personnel MyStyle Ltd'('Size' 400 400)('Coord' 'Pixel')
```

- Afficher la matrice du personnel dans un Grid avec la possibilité d'en modifier le contenu, de supprimer et ajouter des personnes.
Intitulés de lignes :

- *nom*
- *sexe*
- *age*
- *salaire annuel*

```
'F1.G1'⎕WC'Grid'('Posn' 0 0)('Size' 150 400)('Values'Perso)
 F1.G1.RowTitles←'Nom' 'Sexe' 'Age' 'Sal. annuel'
```

Pour modifier le contenu du Grid, il faut créer des champs edit ou autres objets appropriés.
On procède en 2 temps :
- on crée d'abord les objets d'édition, fils du Grid :

```
'Form1.Grid1.Edit1' ⎕WC 'Edit'
```

- on les affecte aux cellules adéquates via la propriété CellTypes :

```
 'Form1.Grid1' ⎕WS 'CellTypes'((⍴Form1.Grid1)⍴1)
 'Form1.Grid1' ⎕WS 'Input' 'Form1.Grid1.Edit1'
```

Commençons par créer nos éditeurs :
- Pour la première ligne, un champ edit alphabétique suffira :

```
 'F1.G1.E1'⎕WC'Edit' ('FieldType' 'Char')
```

- Pour la deuxième, une combo avec les choix 'H' et 'M' :

```
 'F1.G1.C1'⎕WC 'Combo' ('Style' 'Drop')('Items' (2 1⍴'HF'))
```

- Pour les deuxième et troisième lignes on utilisera un éditeur acceptant uniquement des nombres :

```
 'F1.G1.E2'⎕WC'Edit' ('FieldType' 'Numeric')
```

A présent, il ne reste plus qu'à associer les bons éditeurs aux bonnes cellules.
Nous avons 3 éditeurs.
Considérons que F1.G1.E1 sera utilisé par des cellules de type 1, F1.G1.C1 par des cellules de type 2 et F1.G1.E2 par des cellules de type 3.

```
F1.G1.CellTypes←(⍴F1.G1.Values)⍴3
F1.G1.CellTypes[1;]←1
F1.G1.CellTypes[2;]←2
```

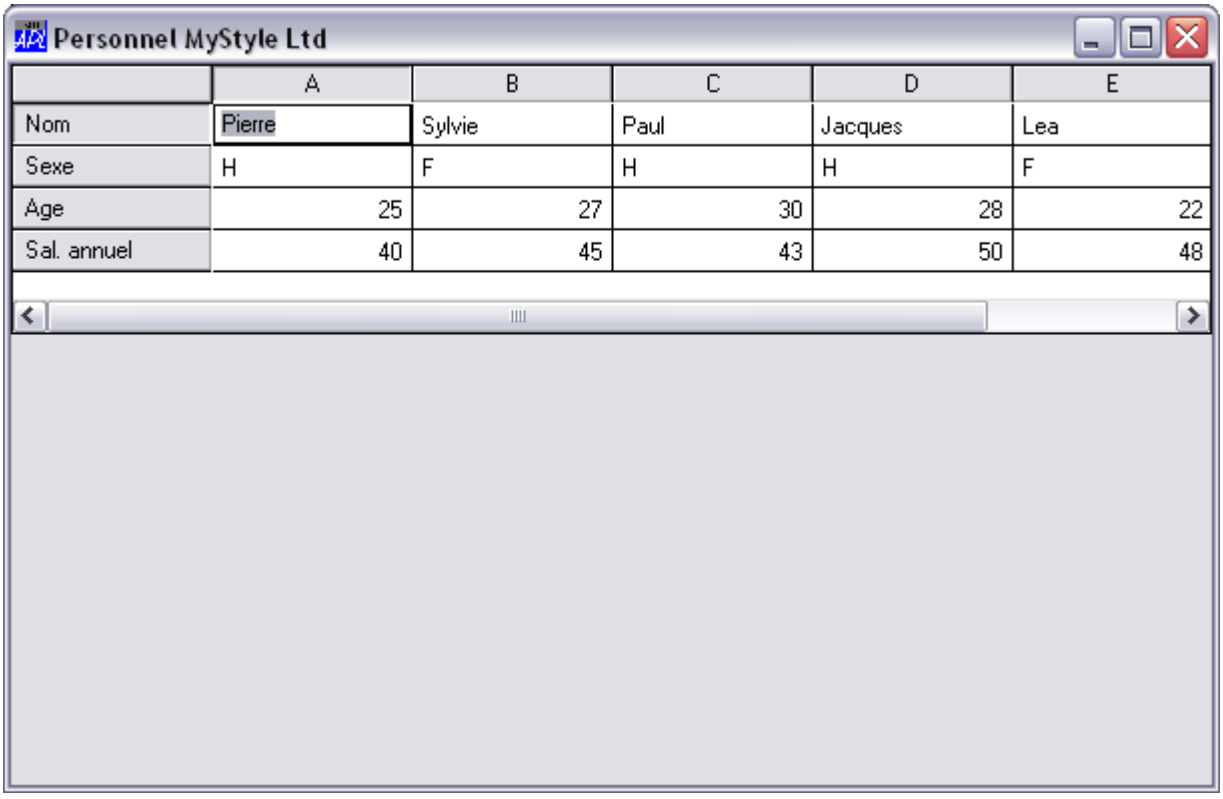
Maintenant, déclarons les objets d'édition dans le bon ordre (F1.G1.E1 pour les cellules de type 1, F1.G1.C1 pour les cellules de type 2 et F1.G1.E2 pour les cellules de type 3) :

```
F1.G1.Input←'F1.G1.E1' 'F1.G1.C1' 'F1.G1.E2'
```

Afin de voir ce que donne cet écran, ajoutez `⎕DQ'F1'` et lancez la fonction.

Listing :

```
GestPerso;F1;tmp
⍝ Gestion du personnel
 'F1'⎕WC'Form' 'Personnel MyStyle Ltd'('Size' 400 400)('Coord' 'Pixel')
 'F1.G1'⎕WC'Grid'('Posn' 0 0)('Size' 150 400)('Values'Perso)
 F1.G1.RowTitles←'Nom' 'Sexe' 'Age' 'Sal. annuel'
 'F1.G1.E1'⎕WC'Edit'('FieldType' 'Char')
 'F1.G1.C1'⎕WC'Combo'('Style' 'Drop')('Items'(2 1⍴'HF'))
 'F1.G1.E2'⎕WC'Edit'('FieldType' 'Numeric')
 F1.G1.CellTypes←(⍴F1.G1.Values)⍴3
 F1.G1.CellTypes[1;]←1
 F1.G1.CellTypes[2;]←2
 F1.G1.Input←'F1.G1.E1' 'F1.G1.C1' 'F1.G1.E2'
 tmp←⎕DQ'F1'
```



- Pour ajouter des personnes, il faut déplacer le curseur à droite, au delà de la dernière colonne. Par défaut le grid n'autorise pas ceci. Il faut modifier sa propriété AutoExpand (booléen de 2 positions : lignes, colonnes) et lui mettre 0 1 afin qu'il autorise l'ajout de colonnes mais pas celui de lignes :

```
F1.G1.AutoExpand←0 1
```

- Pour supprimer les colonnes, l'utilisateur cliquera sur une entête de colonne, faisant apparaître un menu contextuel lui demandant si il veut simplement sélectionner la colonne entière ou si il veut la supprimer.
Nous allons donc commencer par créer un menu directement rattaché à F1 :

```
 'F1.M1'⎕WC'Menu'
 'F1.M1.Del'⎕WC'MenuItem' 'Supprimer'('Event' 'Select' 1)
 'F1.M1.Sel'⎕WC'MenuItem' 'Sélectionner'('Event' 'Select' 0)
```

Ensuite, nous allons lier l'affichage de ce menu à un clic sur un entête de colonne :

```
'F1.G1'⎕WS('Event' 'CellDown' 'GestPerso_sel')
```

Le menu ne doit être activé que si on a cliqué sur une entête de colonne et pas ailleurs dans le grid.
C'est pourquoi la fonction callback GestPerso_sel commence par tester le compte-rendu d'événement qu'elle reçoit automatiquement en argument droit.
Texte de la fonction :

```
 GestPerso_sel d;cr
⍝ Gestion des suppressions via sélection de colonne
⍝ Sortir si l'entête de colonne n'a pas été cliqué
 →(¯1≠7⊃d)/0

⍝ Donner la main au menu
 cr←⎕DQ'F1.M1'

⍝ Sortir si aucun élément du menu n'a été sélectionné
 →(0=⍴cr)/0
 →('F1.M1.Sel'1⊃cr)/0

⍝ Effacer la colonne qui avait été cliquée
 F1.G1.DelCol 8⊃d
```

- Afficher dans un autre grid en dessous les statistiques suivantes :

- *l'effectif masculin, féminin et total*
- *la masse salariale masculine, féminine et totale*
- *les salaires moyens masculin, féminin et total.*

Ces données devront être rafraîchies au fur et à mesure des saisies opérées dans l'autre grid.

Ajoutons la définition du nouveau grid dans GestPerso :

```
 'F1.LS'⎕WC'Label' 'Statistiques'('Posn' 170 0)
 'F1.G2'⎕WC'Grid'('Posn' 200 0)('Size' 100 600)('Values'(3 3⍴0))
 F1.G2.RowTitles←'Effectif' 'Masse salariale' 'Sal. moyen'
 F1.G2.ColTitles←'Hommes' 'Femmes' 'Total'
```

Pour mettre les données de G2 à jour, nous allons utiliser une fonction GestPerso_Stats qui sera utilisée lorsque l'événement CellChanged se produira dans G1.
Nous allons également l'appeler une première fois afin de remplir G2 avec des vraies valeurs.

Définition de GestPerso_Stats :

```
 GestPerso_Stats d;tmp;sexe
⍝ Calcul des statistiques
```

On commence par créer un vecteur des sexes avec les 3 possibilités indiquées en commentaire. En effet, quand une nouvelle personne est créée dans la base, son sexe est inconnu.

```
⍝ Homme=1, femme=2, autre=0
 tmp←F1.G1.Values[2;]
 sexe←(⍴tmp)⍴0
 (('H'≡¨tmp)/sexe)←1
 (('F'≡¨tmp)/sexe)←2

⍝ Effectifs
 F1.G2.Values[1;]←(+/sexe=1)(+/sexe=2)(+/sexe≠0)
```

Pour le calcul suivant, on s'assure que les montants récupérés existent bien. En effet, pendant la création d'une nouvelle personne le salaire n'est pas encore connu.
C'est pourquoi on prend soin de vérifier que chaque position de tmp comporte bien une valeur : `1=⍴¨,¨tmp`
Le `,⍤1` est là uniquement pour que le booléen issu du test soit bien un simple vecteur numérique.

```
⍝ Masse salariale
 tmp←(sexe=1)/F1.G1.Values[4;]
 F1.G2.Values[2;1]←+/(,⍤1=⍴¨,¨tmp)/tmp
 tmp←(sexe=2)/F1.G1.Values[4;]
 F1.G2.Values[2;2]←+/(,⍤1=⍴¨,¨tmp)/tmp
 tmp←(sexe≠0)/F1.G1.Values[4;]
 F1.G2.Values[2;3]←+/(,⍤1=⍴¨,¨tmp)/tmp
```

Le principe est le même que précédemment sauf qu'au lieu de sommer, on calcule la moyenne arrondie à l'entier le plus proche.

```
⍝ Salaires Moyens
 tmp←(sexe=1)/F1.G1.Values[4;]
 F1.G2.Values[3;1]←0 Arrondir Moyenne(,⍤1=⍴¨,¨tmp)/tmp
 tmp←(sexe=2)/F1.G1.Values[4;]
 F1.G2.Values[3;2]←0 Arrondir Moyenne(,⍤1=⍴¨,¨tmp)/tmp
 tmp←(sexe≠0)/F1.G1.Values[4;]
 F1.G2.Values[3;3]←0 Arrondir Moyenne(,⍤1=⍴¨,¨tmp)/tmp
```

Il faut également modifier la définition de F1.G1 afin qu'il appelle GestPerso_Stats quand une cellule est modifiée :
`'F1.G1'⎕WS('Event' 'CellChanged' 'GestPerso_Stats')`

Il faut aussi faire un premier appel à GestPerso_Stats immédiatement après la définition de F1.G2 :
GestPerso_Stats 1

- Ajoutez 2 boutons en bas de l'écran :

- Sauver et quitter : Met à jour la variable Perso et ferme la fonction, donc l'écran puisque vous l'aurez localisé dans la fonction.

- Quitter sans sauver.

On ajoute les lignes suivantes afin de créer 2 boutons qui rendront la main à Apl en sortie du ⎕*DQ*.

```
'F1.BSQ'⎕WC'Button' 'Sauver et Quitter'('Posn' 90 10)('Size'0
35)('Event' 'Select' 1)
 'F1.BAQ'⎕WC'Button' 'Quitter sans Sauver'('Posn' 90 55)('Size'0
35)('Event' 'Select' 1)
 tmp←⎕DQ'F1'
 →(0=⍴tmp)/0
 →('F1.BAQ'1⊃tmp)/0
 Perso←F1.G1.Values
```

Si tmp est vide ou si c'est F1.BSQ qui a été actionné, on sort sans rien faire.
Sinon Perso reçoit le contenu de F1.G1
On a créé une variable locale de plus : tmp, à laquelle on affecte le compte rendu d'événement.

2. Ajoutez 2 nouvelles zones de saisie numériques avec leurs intitulés :
- Pourcentage de variation
- Montant maxi d'augmentation

```
'F1.LS2'⎕WC'Label' 'Pourcentage de Variation :'('Posn' 310 10)('Size'0
150)
 'F1.E1'⎕WC'Edit'('FieldType' 'Numeric')('Posn' 310 170)('Size'0 100)
 'F1.LS3'⎕WC'Label' 'Montant maxi d''augmentation :'('Posn' 340
10)('Size'0 150)
 'F1.E2'⎕WC'Edit'('FieldType' 'Numeric')('Posn' 340 170)('Size'0 100)
```

- Ajoutez un bouton "Lancer Simulation", qui modifie tous les éléments en fonction du % de variation et du plafond (dont la saisie est facultative).
Attention, les données de la simulation ne peuvent pas être sauvegardées.

- Ajoutez un bouton "Fermer Simulation", qui remet les données réelles à l'écran.

Nous avons besoin d'une nouvelle variable locale intitulée PersoLoc qui stockera les valeurs réelles du grid pendant les simulations.

```
GestPerso;F1;tmp;PersoLoc
...
PersoLoc←Perso
...
```

Définition des 2 boutons :

```
'F1.BC'⎕WC'Button' 'Lancer simulation'('Posn' 310 300)('Size'⍬
100)('Event' 'Select' 'GestPerso_simu')
 'F1.BR'⎕WC'Button' 'Fermer simulation'('Posn' 340 300)('Size'⍬
100)('Event' 'Select' 'GestPerso_simu')
```

Texte de la fonction GestPerso_simu :

```
 GestPerso_simu d;Augm
⍝ Calculs de simulations

 :If 'F1.BC'1⊃d   ⍝ Lancer Simu
  ⍝ Toujours repartir des données réelles
   F1.G1.Values[4;]←PersoLoc[4;]
  ⍝ Désactiver le bouton de sauvegarde
   F1.BSQ.Active←0
   ⍝ Ne rien faire si il n'y a pas de pct de variation
   →(0=⍴,F1.E1.Value)/0
  ⍝ Augmentation théorique
   Augm←0.01×F1.G1.Values[4;]×F1.E1.Value
  ⍝ Flafond d'Augmentation
   :If 0≠ ⍴,F1.E2.Value
     Augm←F1.E2.Value⌊Augm
   :EndIf
   ⍝ Ajouter l'augmentation
   F1.G1.Values[4;]+←Augm
   F1.G1.AutoExpand←0 1
 :Else            ⍝ Fin Simu
 ⍝ Remettre les valeurs réelles
   F1.G1.Values←PersoLoc
  ⍝ Réactiver bouton sauvegarde
   F1.BSQ.Active←1
  ⍝ Effacer champs de simulation
   F1.E1.Value←⍳0
   F1.E2.Value←⍳0
   F1.G1.AutoExpand←0 1
 :EndIf

 ⍝ Recalculer les stats
 GestPerso_Stats 1
```

Listing de la fonction GestPerso :

```
GestPerso;F1;tmp;PersoLoc
⍝ Gestion du personnel
 'F1'⎕WC'Form' 'Personnel MyStyle Ltd'('Size' 400 600)('Coord' 'Pixel')
 'F1.G1'⎕WC'Grid'('Posn' 0 0)('Size' 150 600)('Values'Perso)
 F1.G1.RowTitles←'Nom' 'Sexe' 'Age' 'Sal. annuel'
 'F1.G1.E1'⎕WC'Edit'('FieldType' 'Char')
 'F1.G1.C1'⎕WC'Combo'('Style' 'Drop')('Items'(2 1⍴'HF'))
 'F1.G1.E2'⎕WC'Edit'('FieldType' 'Numeric')
 F1.G1.CellTypes←(⍴F1.G1.Values)⍴3
 F1.G1.CellTypes[1;]←1
 F1.G1.CellTypes[2;]←2
 F1.G1.Input←'F1.G1.E1' 'F1.G1.C1' 'F1.G1.E2'
 F1.G1.AutoExpand←0 1

 'F1.M1'⎕WC'Menu'
 'F1.M1.Del'⎕WC'MenuItem' 'Supprimer'('Event' 'Select' 1)
 'F1.M1.Sel'⎕WC'MenuItem' 'Sélectionner'('Event' 'Select' 0)
 'F1.G1'⎕WS('Event' 'CellDown' 'GestPerso_sel')

 'F1.LS'⎕WC'Label' 'Statistiques'('Posn' 170 0)
 'F1.G2'⎕WC'Grid'('Posn' 200 0)('Size' 100 600)('Values'(3 3⍴0))
 F1.G2.RowTitles←'Effectif' 'Masse salariale' 'Sal. moyen'
 F1.G2.ColTitles←'Hommes' 'Femmes' 'Total'
 'F1.G1'⎕WS('Event' 'CellChanged' 'GestPerso_Stats')
 GestPerso_Stats 1

 'F1.LS2'⎕WC'Label' 'Pourcentage de Variation :'('Posn' 310 10)('Size'0
150)
 'F1.E1'⎕WC'Edit'('FieldType' 'Numeric')('Posn' 310 170)('Size'0 100)
 'F1.LS3'⎕WC'Label' 'Montant maxi d''augmentation :'('Posn' 340
10)('Size'0 150)
 'F1.E2'⎕WC'Edit'('FieldType' 'Numeric')('Posn' 340 170)('Size'0 100)
 'F1.BC'⎕WC'Button' 'Lancer simulation'('Posn' 310 300)('Size'0
100)('Event' 'Select' 'GestPerso_simu')
 'F1.BR'⎕WC'Button' 'Fermer simulation'('Posn' 340 300)('Size'0
100)('Event' 'Select' 'GestPerso_simu')

 PersoLoc←Perso

 'F1.BSQ'⎕WC'Button' 'Sauver et Quitter'('Posn' 175 10)('Size' 14
80)('Event' 'Select' 1)
 'F1.BAQ'⎕WC'Button' 'Quitter sans Sauver'('Posn' 175 120)('Size' 14
80)('Event' 'Select' 1)

 tmp←⎕DQ'F1'
 →(0=⍴tmp)/0
 ⍝ Quitter sans Sauver
 →(('F1.BAQ'1⊃tmp)∨'F1.MB.M1.QSS'1⊃tmp)/0

 ⍝ Sauver et Quitter
 Perso←F1.G1.Values
```