Ecole Supérieure de Technologie d'Essaouira **Licence ISIL**
Enseignant : F. Karami

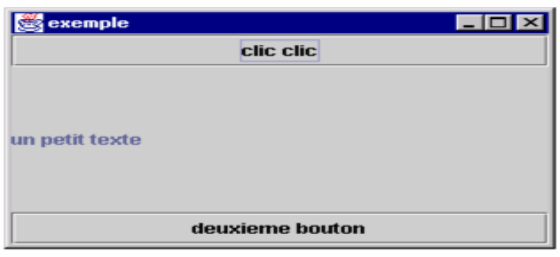
Java- IHM

EXERCICE 1 : Ecrire un programme qui crée une fenêtre (de type *JFrame*) et qui détecte les événements "appui" et "relâchement" associés à la souris et ayant la fenêtre comme source. On se contentera de signaler chacun de ces deux événements en affichant en fenêtre console un message précisant sa nature (appui ou relâchement), ainsi que les coordonnées correspondantes.
On proposera quatre solutions :
1. la fenêtre est son propre écouteur de souris et elle implémente l'interface *MouseListener*,
2. on utilise un écouteur différent de la fenêtre, objet d'une classe implémentant l'interface *MouseListener*,
3. on utilise un objet écouteur différent de la fenêtre en recourant à l'adaptateur *MouseAdapter*,
4. on utilise un écouteur différent de la fenêtre, objet d'une classe anonyme dérivée de *MouseAdapter*.

EXERCICE 2 : Écrire un programme qui crée une fenêtre (*JFrame*) et qui y affiche *n* boutons portant les étiquettes *BOUTON1*, *BOUTON2*... disposés comme dans cet exemple :

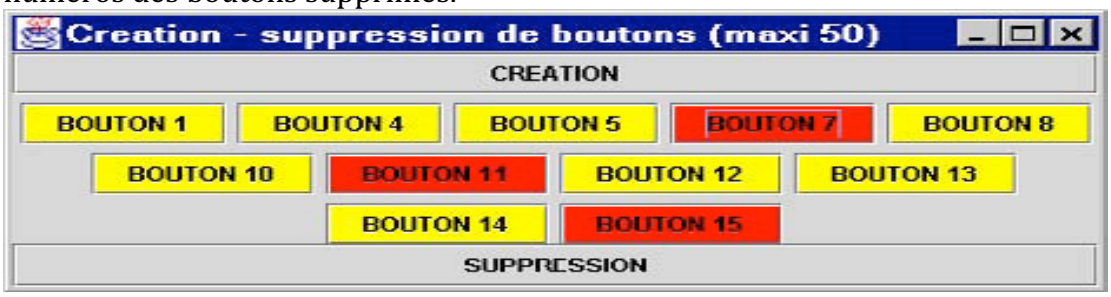
La valeur de *n* sera lue au clavier.

EXERCICE 3 : Réalisez la fenêtre suivante (BorderLayout)



EXERCICE 4 : (Création et suppression de boutons)
Écrire un programme qui affiche une fenêtre comportant deux boutons d'étiquettes "CREATION" et "SUPPRESSION" placés respectivement en haut et en bas.
Chaque action sur le bouton CREATION conduira à la création d'un bouton jaune à l'intérieur de la fenêtre. Chaque action sur l'un des boutons de la fenêtre le "sélectionnera" (s'il ne l'est pas déjà) ou le "désélectionnera" (s'il l'est déjà). On

visualisera un bouton sélectionné en le colorant en rouge. Chaque action sur le bouton SUPPRESSION supprimera tous les boutons sélectionnés (rouges).
Les boutons seront numérotés dans l'ordre de leur création. On ne réutilisera pas les numéros des boutons supprimés.

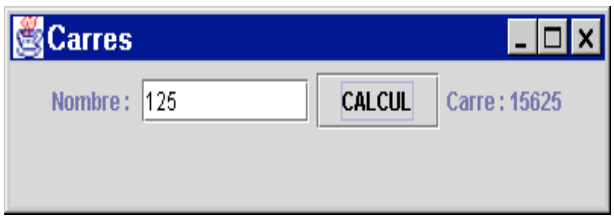


Par souci de simplicité, on fournira au constructeur de la fenêtre le nombre maximum de boutons susceptibles d'être créés.

Outils :
Comme le suggère l'image fournie dans l'énoncé, les deux boutons CREATION et SUPPRESSION peuvent être disposés dans la fenêtre en utilisant son gestionnaire par défaut de type BorderLayout. Il suffira simplement de préciser les paramètres "North" et "South". En revanche, les boutons gérés dynamiquement devront être placés dans un panneau distinct qu'on placera au centre de la fenêtre (option par défaut de la méthode add). Le gestionnaire par défaut d'un panneau est de type FlowLayout, ce qui nous conviendra ici. Nous faisons de la fenêtre l'écouteur de tous les boutons. Comme il est nécessaire de conserver une information de couleur pour chacun des boutons dynamiques, nous prévoyons à cet effet un tableau boutons comportant les références des boutons dynamiques et un tableau boutSelec contenant une information booléenne de sélection. Ces deux tableaux auront une taille fournie lors de l'appel du constructeur de la fenêtre.
Chaque fois qu'on modifie le contenu du panneau, soit en ajoutant un nouveau bouton, soit en supprimant les boutons sélectionnés, on fait appel à sa méthode validate, afin de forcer son gestionnaire de mise en forme à recalculer les positions des différents composants. En revanche, cet appel n'est pas nécessaire lors de la modification de la couleur d'un bouton (par setBackground) car il est alors réalisé automatiquement.
Vous utilisez classiquement une variable statique nBout pour numéroter nos boutons. On notera que le premier bouton porte le numéro 1, alors qu'il correspond à l'indice 0 dans les tableaux boutons et boutSelec.

EXERCICE 5 : Champ de texte et événements Action et Focus

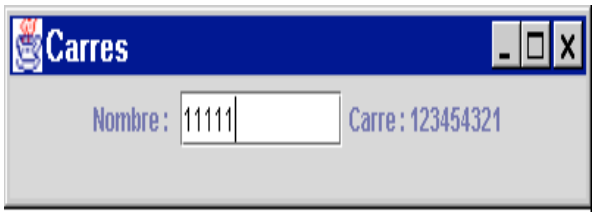
1 : Écrire un programme qui permet à l'utilisateur de saisir un nombre entier dans un champ texte et qui en affiche le carré lorsqu'il agit sur un bouton marqué CALCUL :



Le programme devra gérer convenablement le cas où l'utilisateur entre autre chose qu'un nombre dans le champ texte ; il pourra par exemple remettre ce champ à blanc.

Outils 1 *: Vous pouvez introduire directement dans la fenêtre les différents contrôles dont vous avez besoin. Vous remplacez simplement le gestionnaire par défaut par un gestionnaire de type FlowLayout.*
Vous utilisez des objets de type JLabel pour les libellés, ainsi que pour la valeur du carré. La saisie du nombre se fait dans un objet nommé nombre de type JTextField.
Ici, vous n'aves pas à vous préoccuper des événements générés par nombre puisque le calcul proprement dit est déclenché par une action extérieure à l'objet. En revanche, vous devez traiter les événements de type Action déclenchés par le bouton. Vous y récupérez le contenu du champ texte que vous convertissez en entier avec la méthode Integer.parseInt. Celle-ci déclenche une exception NumberFormatException lorsque la chaîne ne correspond pas à un nombre entier (y compris lorsqu'elle contient trop de chiffres). Dans le gestionnaire d'exception correspondant, vous vous contentez de remettre à blanc le contenu du champ texte.
C'est mieux de calculer le carré du nombre dans le type long, ce qui évite tout problème de dépassement de capacité.

2- Adapter le programme ci-dessus en supprimant le bouton CALCUL et de manière que le carré du nombre s'affiche lorsque l'utilisateur valide l'information saisie ou lorsque le champ de texte perd le focus :



Outils 2 *:*
Il suffit que les actions précédemment réalisées dans l'écouteur du bouton soient transposées :
- *dans l'écouteur de l'événement focusLost associé au champ de texte,*
- *dans l'écouteur de l'événement Action associé à ce même champ de texte.*

Pour éviter de dupliquer les instructions correspondantes, vous prévoyez une méthode de service nommée actualise.

EXERCICE 6 : Cases à cocher

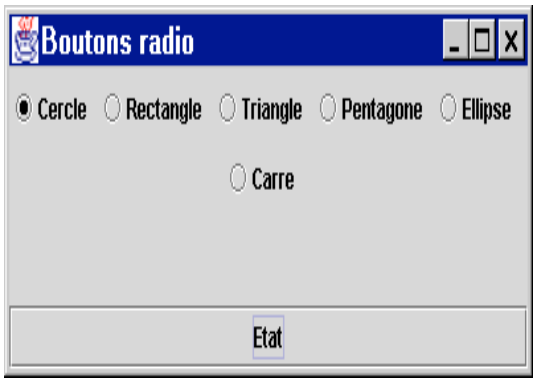
Écrire un programme qui affiche deux boutons marqués RAZ et Etat et trois cases à cocher, de la façon suivante :

L'action sur le bouton Etat provoquera l'affichage en fenêtre console des cases sélectionnées.
L'action sur RAZ remettra les trois cases à l'état non coché. Enfin, on signalera en fenêtre console les événements de type Action et Item associés à chacune des trois cases (en précisant la source concernée).

Outils : *Vous placeez les trois cases dans un panneau associé à la fenêtre. Vous faisez de la fenêtre l'écouteur des boutons et des cases. Comme l'impose l'énoncé, vous redéfinissez à la fois les méthodes actionPerformed et itemStateChanged.*

EXERCICE 7 : Boutons radio en nombre quelconque

Écrire un programme qui affiche un bouton marqué Etat et un (seul) groupe de boutons radio de la façon suivante :

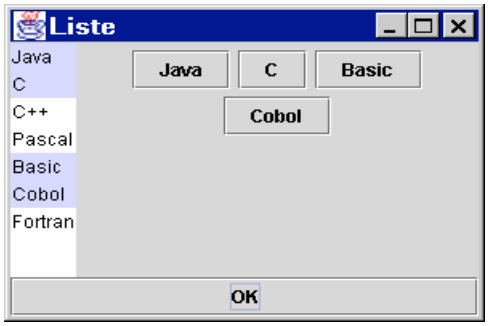


Les libellés des boutons radio seront fournis en argument du constructeur de la fenêtre. L'action sur le bouton Etat provoquera l'affichage en fenêtre console du libellé associé au bouton radio sélectionné. On signalera en fenêtre console les événements de type Action associés.

EXERCICE 8 : Gestion d'une boîte de liste

Écrire un programme affichant dans une fenêtre des boutons dont les étiquettes sont des noms de langage sélectionnés dans une boîte de liste. La liste permettra de sélectionner un nombre quelconque de plages de valeurs. Les noms des langages seront fixés dans la méthode main (et non dans la fenêtre). On proposera deux solutions :

- une où la sélection sera validée par l'action sur un bouton OK :
-



- une où les boutons affichés dans la fenêtre seront actualisés à chaque modification de

la sélection dans la liste (il n'y aura plus de bouton OK).

Outils : *Les noms de langages sont définis par un tableau de chaînes de la méthode main qu'on fournit en argument au constructeur de la fenêtre. La boîte de liste est ajoutée à la fenêtre elle-même avec l'option "West". Un panneau est ajouté au centre de la fenêtre, en vue d'y afficher les boutons voulus.*
Le bouton OK est ajouté avec l'option "South" et on gère ses événements de type Action. La méthode actionPerformed réalise les actions suivantes :

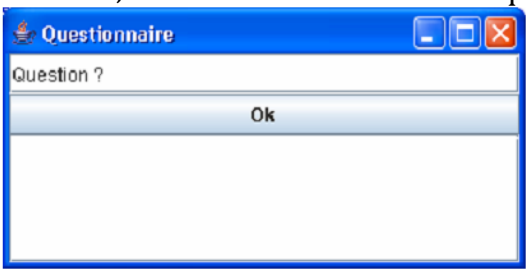
• *suppression des boutons du panneau par la méthode removeAll (qui supprime tous les composants d'un conteneur) ;*

• *récupération des valeurs sélectionnées dans la boîte de liste à l'aide de getSelectedValues. Elle fournit un tableau d'éléments de type Object qui seront convertis en String, avant d'être transmis au constructeur de chacun des boutons ;*

• *appel de la méthode validate du panneau pour forcer le recalcul par le gestionnaire de mise en forme.*

EXERCICE 9 : Réaliser une fenêtre qui contient un label.
Le label change de couleur chaque fois le curseur de la souris passe dessus.
Essayer d'implémenter le gestionnaire d'événement en 4 façons différentes:
- intégré
- classe externe,
- classe interne,
- classe anonyme.

EXERCICE 10 :
Réaliser un formulaire comprenant un champ de saisie, un bouton et une plage de texte. (cf. ci-dessous). Puis faire en sorte que lorsque l'on a validé une question dans le champ de saisie, on affiche un écho de cette question dans le composant réponse.



Précision: la production de la réponse doit être déclenchée quand on "appuie" sur le bouton JButton ET quand on "valide" le champ de saisie JTextField (en appuyant sur la touche <ENTER>). Quel est le type d'événement qui permet de gérer ces deux actions de la même manière?

Considérations architecturales: essayer d'implémenter le gestionnaire d'événement en 4 façons différentes:
- intégré
- classe externe,
- classe interne,
- classe anonyme