|  |  |
| --- | --- |
| Formatif D gr 01 | **Interfaces Swing avec WindowBuilder** |

|  |
| --- |
| Légende pour tous les formatifs:  ✪ Question pour les Pro ou non-essentielle (non requis dans les mini-projets à remettre)  ❓ Question théorique seulement: vous devriez connaître la réponse à cette question, mais on ne vous  demande pas de fournir cette réponse, ni d'intégrer votre solution au code du projet à remettre. |

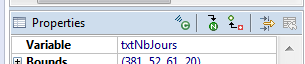
Pour tous les exercices, prenez soin de renommer adéquatement tous les composants (exception pour les étiquettes de titrage).

1. Chargez le projet *InteractionsDiverses*. Si les onglets Source et Design sont absents: ouvrez la classe *InteractionsDiverses .java* avec le clic-droit + *Open With WindowBuilder Editor*.

Cliquez sur la zone de saisie de texte, en haut. La fenêtre *Properties* (vis-à-vis *Class*) vous indique qu’il s’agit d’un composant *JTextField* et vous montre ses caractéristiques de départ. Exécuter l'application.

* 1. Sélectionnez de nouveau la zone de saisie de texte. Enfoncez la touche CTRL, puis cliquer sur le tourniquet, et ensuite sur chacun des 3 boutons : 5 composants sont maintenant sélectionnés. Dans les *Properties*, choisissez la caractéristique commune *font*. Choisir une police Arial, Bold, de 15 points.
  2. Agrandissez l’interface vers le bas. Dans le bas de l’application, ajoutez une étiquette qui dit « Zone informative ». Puis ajoutez une autre étiquette vide. Tous les messages y seront affichés. Donnez des noms judicieux à ces deux composants (voir *Properties /Variable*).
  3. Ajoutez une ligne de séparation horizontale bleue entre les composants pour le commentaire et ceux pour la zone d’information.
  4. On veut que le premier bouton affiche le double de la valeur contenue dans la zone de texte. Double-cliquez sur le premier bouton « Doubler? » : vous êtes dans l’écouteur. Ajouter les commentaires *//debut* et *//fin*. Entre les deux, ajouter du code qui calcule et affiche le résultat dans la zone d’information.
  5. Sélectionnez le tourniquet. Modifiez son modèle pour qu’il accepte une valeur entre 0 et 25, par intervalles de 5 seulement.
  6. Ajoutez le code pour qu’au clic du second bouton « Doubler », le double du nombre présentement choisi dans le tourniquet s’affiche dans la zone d’information.
  7. Ajoutez le code pour qu’au clic du bouton « Ajouter un mot », le mot "OK" soit **ajouté** à ce qu’il y a déjà dans la zone de texte multi lignes prévue pour entrer les commentaires (OKOKOK….).

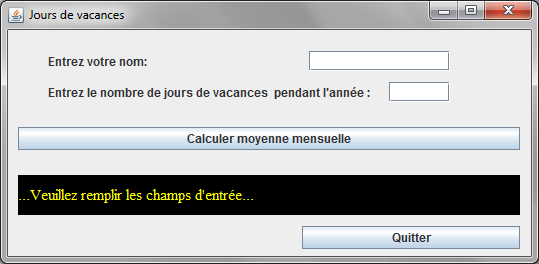
1. Chargez le projet *MoyenneMensuelle*.
2. Le programmeur a oublié d'importer plusieurs paquetages. Utilisez CTRL+MAJ+O (pour ***o****rganise imports*) pour forcer Eclipse à réviser toutes les importations nécessaires, d'un seul coup. Eclispe vous offre de choisir d’importer une classe javafx ou une classe awt / swing? Choisissez **awt / swing**. Entregistrez. Il reste encore une erreur…(on y reviendra).
3. Les onglets Source et Design sont absents: ouvrez la classe *Application* avec le clic-droit *Open With WindowBuilder Editor*.
4. Autour de la ligne 75 : le compilateur vous dit que *txtNbJours* est inconnu. La raison? Parce que la **déclaration** de cet identificateur survient *plus loin* dans le code. Retournez en mode Design. Sélectionnez la zone txtNbJours, puis appuyez sur le bouton *Convert local to field*. Maintenant txtNbJour est un champ, déclaré au haut de la classe: il est donc connu en tout temps, partout dans le fichier|!



❓Quel réglage du workspace aurait permis d’éviter ce genre de situation, et que le programmeur n'a visiblement pas utilisé?

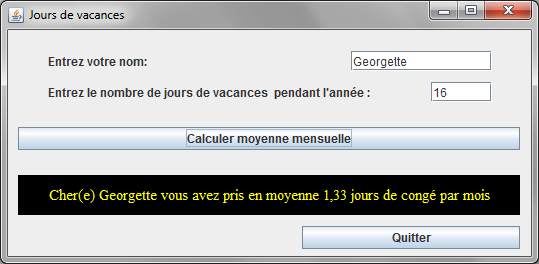
1. En utilisant les outils d'alignement d'Eclipse:

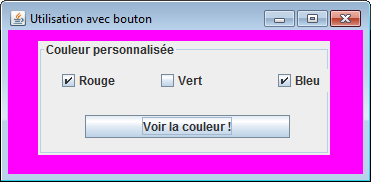
vous désirez obtenir le résultat suivant, en suivant les étapes ci-dessous.



* Sélectionnez le bouton *Calculer* (ce sera votre modèle). Enfoncez la touche CTRL puis sélectionnez aussi l'étiquette noire. Utiliser l'outil "replicate width" pour que l'étiquette prenne la même largeur que le bouton. Centrez-les tous les deux dans la fenêtre, avec le bouton "center horizontally in window".
* Sélectionnez le bouton Calculer ou l'étiquette noire, au choix (ce sera votre modèle). Touche CTRL + sélectionnez le bouton Quitter. Utiliser l'outil "Align right edge".
* Maintenant, utilisez le même processus pour aligner des deux zones de saisie blanches entre elles. Et faites de même avec les deux étiquettes immuables "Entrez votre nom" et "Entrez le nombre de jours…".

1. Sélectionnez l'étiquette noire : dans la section *Properties* de WindowBuilder, modifiez la caractéristique qui définit son alignement horizontal, de façon à ce que le texte jaune soit centré dans l'étiquette (fouillez un peu…).
2. Complétez le code pour qu'au clic du bouton de calcul, l'application affiche dans l'étiquette noire la moyenne des jours de vacances par mois. La phrase doit contenir le prénom entré par l'utilisateur, par exemple *"Cher <untel>, vous avez pris en moyenne <x.xx> jours de congé par mois".* Quelques indices: vous aurez besoin de *Integer.parseInt* et d'une méthode *setText*.
   * Affichez le résultat avec 2 décimales de précision (voir diapositive intitulée « gestion des décimales », dans le PowerPoint D1.



1. ❓ Vous aimeriez ajouter un bouton qui permet d'agrandir la police de caractères de l'étiquette noire à 20 points. Vous ne savez pas quelle instruction permet de faire cela, et vous êtes trop paresseux pour lire la documentation en ligne. Comment pourriez-vous procéder?
2. ✪ Ajoutez la fonctionnalité suivante à l'application: quand on clique sur le bouton Calculer et que l'une ou l'autre des deux entrées est absente, un message d'erreur est affiché et la couleur de fond de la zone manquante passe au jaune (et questionnez-vous sur le moment logique où elle devrait revenir au blanc par la suite…)
3. ✪ Créez un nouveau projet vide *CouleurDeFond*. Créez-y une classe graphique nommée *CouleursAvecBouton*. Placez sur l'interface trois **cases à cocher**, titrées "Rouge", "Vert" et "Bleu". Placez également un bouton "Voir la couleur". Entourez les cases et le bouton d'un JPanel avec bordure.
   1. Au clic du bouton, l'arrière-plan de l'application doit changer de couleur (par exemple si rouge et bleu sont cochés, alors la couleur sera magenta, c'est-à-dire (255, 0, 255) ). Contrainte pour cet exercice seulement:
      * + Votre classe doit posséder un seul écouteur de type *actionListener* (celui du bouton).
        + Généralisez votre solution: évitez de tester systématiquement toutes les combinaisons de couleurs possibles! Indice: vous devrez utiliser des variables pour mémoriser les quantités de rouge, de vert et de bleu nécessaires. Exemple:

Color maCouleur = new Color(*quantiteRouge*, *quantiteVert*, *quantiteBleu*);

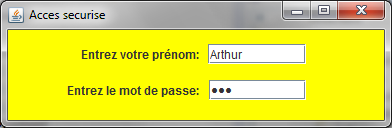
contentPane.setBackground( maCouleur );

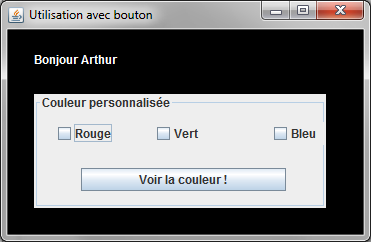
* 1. Faites une copie de votre classe. Renommez cette dernière *CouleursSansBouton*. Modifiez cette version du projet ainsi: ôtez le bouton. La couleur de fond doit maintenant s'adapter instantanément selon les cases sélectionnées. Aucune contrainte au niveau du nombre d'écouteurs.

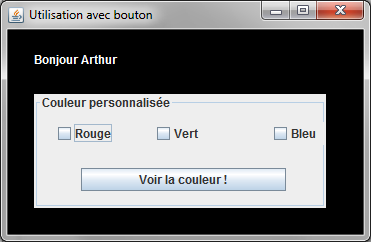
1. [*Lire l’énoncé au complet avant de débuter la programmation*] Créez un projet nommé *AnimauxEnCourse*. Créez-y une classe graphique nommée *Coureurs*. Cette application doit permettre de calculer la distance parcourue en un temps déterminé, pour divers animaux dont la vitesse de course est connue.

Votre interface doit posséder au minimum les éléments suivants (ajoutez des titres et libellés partout où il est utile de le faire) :

* Un panneau à deux onglets, ainsi qu'un bouton "Quitter".
* Sur le premier onglet:
  + Une **liste déroulante** offrant le choix d'animaux suivants: crabe, chevreuil, cheval, guépard. Le crabe a une vitesse de course moyenne de 12 km/h; le cheval 61 km/h; le chevreuil 72 km/h; le guépard 94 km/h (pour chaque choix, montrez le nom de l’animal et sa vitesse).
  + Un **curseur** permettant de sélectionner un temps de course, allant de 0 à 15 secondes inclusivement.
  + Une **étiquette** affichant la distance parcourue par l'animal après le temps sélectionné. Rappel: *distance (m) = vitesse (m/sec) \* temps(sec).* L’étiquette doit être mise à jour à mesure que le curseur est déplacé et lorsque l'animal choisi change. De plus, vous devez afficher exactement 2 décimales de précision dans votre réponse.
* Sur le deuxième onglet:
  + Un sélecteur de couleur, qui change instantanément l'arrière-plan du premier onglet. Pour le faire fonctionner, vous pouvez partir du code que vous trouverez dans *codeAnimaux.txt*.
* L'application doit avoir un menu *Aide* qui doit contenir un item *Instructions* affichant des instructions (très simples,1 ou 2 phrases) avec *JOptionPane*.
* Avez-vous pensé à renommer adéquatement tous les composants?
* Votre code de calcul et d'affichage du résultat est-il dupliqué ? Placez-le dans une méthode privée, qui sera appelée à deux occasions (curseur déplacé et animal choisi a changé).
* ✪ Pour les pros: ajouter une barre de progression qui reflètera en tout temps la proportion des animaux testés jusqu’à présent : quand l’utilisateur essaie avec un nouvel animal, 25% s’ajoute à la barre (puisqu’il y a 4 animaux à tester en tout).

2. Créez un projet nommé *ProjetAvecDeuxFenetres*.
   1. Créez-y une classe graphique nommée AccesProtege. Cette interface doit contenir une zone de texte pour entrer un prénom, ainsi qu'une zone de saisie aveugle.



* 1. Copiez la classe résultant du numéro 2(a) dans votre projet. Retirez le *main* de cette classe. Ajoutez une étiquette qui dit "Bonjour" au haut de l'interface.
  2. Quand l'utilisateur démarre le projet, seule la fenêtre d'accès s'affiche. Il entre son prénom; ensuite il entre le mot de passe et appuie sur la touche *Entrée*. S'il entre le mauvais mot de passe, l'application se termine. S'il entre le bon mot de passe alors la fenêtre de choix de couleurs lui est affichée, et la fenêtre d'accès disparait (utilisez *dispose()* pour se faire elle-même disparaitre à jamais).
* Pour fins de correction : le bon mot de passe sera "abc", peu importe le prénom entré!
  + - À savoir: quand on tape *Entrée* dans une zone de texte ou de saisie aveugle, l'événement *actionPerformed* est déclenché.
    - Pour fins de sécurité, vous devez effacer en mémoire toute trace du mot de passe quand l'utilisateur accède à la seconde fenêtre.
  1. Le prénom que l'utilisateur a spécifié doit apparaitre dans l'étiquette que vous avez ajoutée sur la fenêtre de choix de couleurs (exemple: "Bonjour Arthur"). Comme dans tous les exercices, vous devez continuer d’appliquer les règles d’encapsulation, donc tous les champs des deux classes doivent être déclarés *private*. De plus, aucune méthode *static* ne doit être utilisée.
* Approche suggérée: communiquer à l'aide d’une méthode *afficherPrenom(String lePrenom)* ajoutée à la classe *CouleursAvecBouton*. Cette méthode affichera dans l’étiquette de salutations le prénom reçu en paramètre.
  1. Prévoyez maintenant le cas où l'utilisateur entre les informations dans l'ordre inverse, c'est-à-dire il entre le mot de passe en premier, puis le prénom + *Entrée :* le comportement devrait être équivalent. Attention! Pour éviter toute duplication de code inutile, utilisez une méthode privée appelée en deux occasions.

✪ De plus, donnez un message d'erreur quand l'une ou l'autre des entrées est vide (sur la fenêtre d'accès) au moment de la validation.

* 1. Ouvrez ArgoUML. Chargez-y le fichier *Projet2Fenetres.argo*. Complétez le diagramme UML pour qu'il reflète le projet que vous venez de terminer. Précisez les relations entre les classes, les multiplicités et les navigabilités.
* Par souci de simplification, les classes *JButton*, *JTextField*, *JLabel*, etc ne seront pas représentées sur le diagramme. Seules les classes *JFrame* et *JPanel* seront présentes pour exprimer la nature des classes via des généralisations.
* Omettre ici les attributs primitifs et les opérations, à part l’opération *afficherPrenom* que vous ajouterez pour sa pertinence.

1. ❓Êtes-vous d'accord avec l'affirmation suivante?

*Très souvent, les écouteurs associés aux cases à cocher commencent avec une instruction de type "if". Mais ceci n'est pas vrai pour les écouteurs associés aux boutons radio.*

1. ✪ Créer une application *DuoDeComposants* qui permet à un utilisateur d’entrer la masse d’un objet (limites à votre choix), et qui affiche la force gravitationnelle exercée sur cette masse sur la terre. Contrainte : pour entrer la masse, vous devez offrir à la fois un curseur et une zone de texte. Évidemment, les deux composants doivent en tout temps refléter le tout dernier choix de l’utilisateur.