
	TP J5 JAVA	
---	---------------------------------	---

Java met à notre disposition de classes et des interfaces, qu'on doit importer, dans tous les cas :

- On instancie les classes
- On ré-déclare et on définit **toutes** méthodes (qui sont juste déclarées) d'une interface, dans la classe héritière afin d'en utiliser une.

Interface graphique

Exceptionnellement nous allons créer une seule classe



On va créer les objets séparément et on va les assembler pour obtenir un SEUL OBJET → **POO**

1. **Création** d'un panneau (**JPanel**)
2. **Création** de gestionnaire de disposition : **GridLayout (nbLigne, nbCol)**
3. **Rajout** du gestionnaire de disposition au panneau créé: **setLayout()**
4. **Création** des contrôles en mémoire (label ou étiquette, champ de texte, bouton)
(**JLabel, JTextField, JButton**)

```
JLabel label1 = new JLabel("Nom");
```

5. **Rajout** des contrôles au panneau : **add()**
6. **Création** du cadre : **JFrame**
7. **Rajout** du panneau à la fenêtre (cadre) : **setContentPane()**
8. Réglage de la dimension du cadre : **setSize(200 ,100)** //argument largeur et hauteur
9. Rendre visible le cadre : **setVisible(true)**

Création	<p>Revient à instancier la classe qui se trouve dans les packages :</p> <ul style="list-style-type: none"> - awt de java <p>OU</p> <ul style="list-style-type: none"> - swing de javax : JNomClasse <p>Exemple : Création d'une paire de lunette qu'on mettra sur la tête</p> <p>1. <u>Créer la tête</u></p> <pre>Tete tete = new Tete(); Lunette lunette = new Lunette();</pre>
Affectation	<p>2. <u>Rajout les lunettes à la tête</u></p> <p>Selon ce qu'on rajoute, la méthode change :</p> <p>Attention : on prend d'abord la tete (conteneur) et on rajoute les lunettes(contenu)</p> <pre>tete .addXXX(lunette) //add() prend en argument l'élément à rajouter</pre>

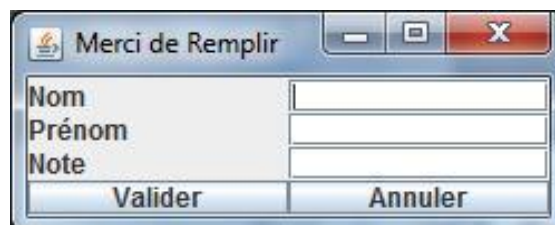
	<p style="text-align: center;">TP J5</p> <hr/> <p style="text-align: center;">JAVA</p>	
---	--	---

1. Exercice 1 : Interface graphique seule

Ecrire un programme java avec juste une classe, qui permet de créer et afficher la fenêtre ci-dessous.

Elle est composée des contrôles :

- ⇒ Etiquette (label) et champ de texte : **Nom - Prénom - Note**
- ⇒ Bouton : **Valider - Annuler**



Création étape par étape d'une interface graphique fonctionnelle (avec un moteur)

Mot clé : fonction, argument, retour, interface, appel de fonction

2. Exercice 2 : Avec Moteur et affiche sur une boîte de dialogue

Avec 2 classes : Classe **Moteur** et la classe principale **GestionMoteur**

1. Déclarer et définir la classe **Moteur** qui va hériter de l'interface **ActionListener**
2. Dans sa définition, On définit l'unique méthode de l'interface **ActionListener** à savoir la méthode **actionPerformed()**
 - ⇒ Qui ne retourne rien et encapsulé public
 - ⇒ Qui prend pour argument un objet qu'on appellera **evenement** de type **ActionEvent**

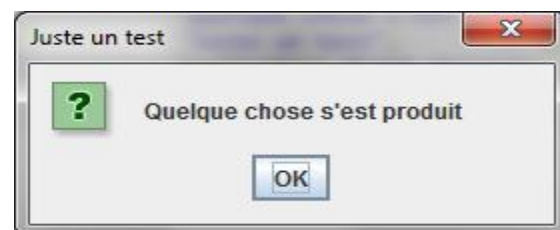
Dans la définition de la méthode **actionPerformed()**, on fait appel à la méthode **showConfirmDialog()** qui a 4 arguments :

 - ⇒ null // la boîte de dialogue n'a pas de classe mère
 - ⇒ "le message à afficher par la fenêtre" // fonctionne comme avec print()
 - ⇒ "le titre de la fenêtre"
 - ⇒ JOptionPane.PLAIN_MESSAGE // un seul bouton de validation marqué "OK"

```
JOptionPane.showConfirmDialog(null,
                              "quelchose s'est produit",
                              "Juste un test",
                              JOptionPane.PLAIN_MESSAGE);
```

3. Penser à instancier la classe que vous venez de créer, dans la classe principale **GestionMoteur**. On appellera la méthode **actionPerformed()**, qui prend pour argument **null**

➔ Le moteur n'est associé à aucun bouton pour l'instant



3. Exercice 2Bis : une boîte de dialogue si clic

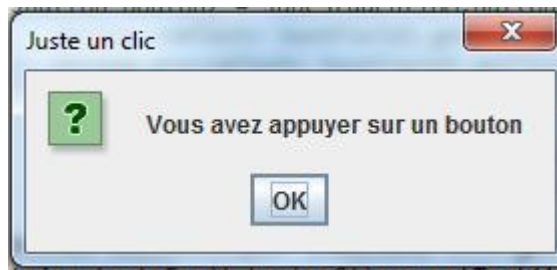
Dorénavant : 3bis. Remplace 3 → On mettra 3. en commentaire

Dans la classe principale :

- **3bis.** Ajouter le moteur au bouton : pour se faire, on fera appel à la méthode `addActionListener()`, qui prend pour **argument** l'objet créé lors de l'instanciation de la classe **Moteur**

```
bouton.addActionListener(moteur);
```

- Rajouter le code de l'interface graphique que vous avez créée dans **Exercice 1** afin d'avoir les **boutons** ...

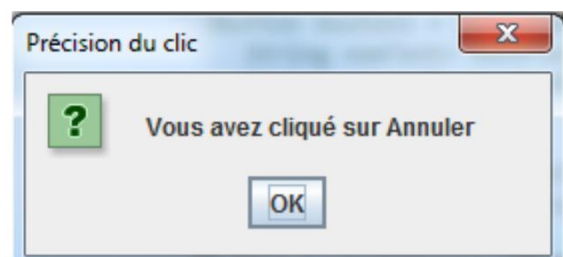
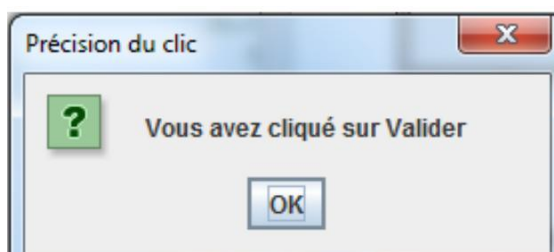


4. Exercice 2BisBis : Précision du bouton cliqué

4. Dans la classe Moteur, dans la boîte de dialogue modifier le texte par «**Vous avez appuyé sur** » en précisant le texte du bouton cliqué.

- ⇒ On fera appel à la méthode `getSource()` pour récupérer le bouton cliqué,
- ⇒ On fera appel à la méthode `getText()` pour récupérer le texte du bouton cliqué.

```
JButton boutonClique = (JButton) evenement.getSource();  
String texteEcrit = boutonClique.getText();
```



5. Exercice 2BisBisBis : récupération des **résultats** dans une boîte de dialogue

Dans la classe principale

⇒ Les boutons et les champs de textes (qu'on souhaite récupérer dans la classe **Moteur**) seront précédés du mot **clé static** et créer **avant** la méthode main() de la classe principale nommée **GestionMoteur** → on a donc des **variables globales**.

Dans la classe Moteur

Ces contrôles static peuvent être appelés dans la classe **Moteur** grâce au constructeur de la classe principale

```

TextField champTexte1= GestionMoteur.champTexte1;
JButton button1 = GestionMoteur.button1;

```

- ⇒ Récupérer le texte du champ **champTexte1**, grâce à la méthode **getText()**;
- ⇒ Le texte récupéré sera converti en entier grâce à la méthode **parseInt()** si nécessaire.

```

String d= champTexte1.getText();
int a = Integer.parseInt(d);

```

- ⇒ Détailler les 2 possibilités : button1 - button2
- Afficher le résultat avec une phrase (voir la boîte de dialogue ci-dessous)



Si clic sur	Si clic sur