Programmation Orientée Objet avec JAVA
TP N°1
Environnement de développement.
Compilation et exécution des premières classes.
Utilisation de quelques classes de l'API

Objectif du TP :

- **Se familiariser avec l'environnement de développement en langage JAVA**
  - **Mode commande**
  - **Utilisation de l'IDE Eclipse**
- **Compiler et exécuter une application "standalone" et une applet java**
- **Créer une classe et utiliser les constructeurs, les getters et les setters**
- **Utiliser la méthode toString de la classe Object et en créer une pour sa propre classe**
- **Utiliser quelques classes de l'API : Scanner, JOptionPane, String**

Organisation du travail :

Il est recommandé de créer un répertoire pour contenir tous les fichiers de vos TPs (ex:JAVA/TP). Pour chaque TP, vous créerez un sous répertoire. Par exemple pour le TP1, créez le sous répertoire JAVA/TP/TP1. De même pour chaque exercice vous créerez un sous répertoire (JAVA/TP/TP1/Exo1, JAVA/TP/TP1/Exo2, …etc.).

Exercice 1 : compiler et exécuter une application "standalone"

a-Lecture d'une donnée au clavier avec la classe Scanner et affichage d'un message avec System.out.println(message)

Ecrire une classe **BonjourConsole** qui utilise la classe **java.util.Scanner** de l'API pour lire au clavier votre nom, puis affiche "*Bonjour " suivi par votre nom* sur l'écran de la console.
(Voir Annexe 1 pour apprendre à utiliser la classe Scanner)

b-Lecture d'une donnée au clavier avec la boite de dialogue JOptionPane

Ecrire une classe **BonjourBoiteDialogue** qui utilise la classe **javax.swing.JOptionPane** de l'API pour lire au clavier votre nom en utilisant la méthode **showInputDialog**, puis affiche "*Bonjour " suivi par votre nom* dans une boite de dialogue avec la méthode **showMessageDialogue**.
(Voir Annexe 2 pour apprendre à utiliser la classe JOptionPane)

Rappel :

1. Une application standalone possède au moins une classe qui contient la méthode main ayant exactement la signature « **public static void main(String[]*args*)**»
2. Pour compiler, utiliser la commande javac suivi du nom du fichier
   (**javac <nomFichier>**).
   Exemple:
   - javac BonjourConsole.java
   - javac BonjourBoiteDialogue.java

   Attention : vous devez avoir modifié le PATH pour exécuter les commandes java à partir de n'importe quel répertoire.
3. Pour exécuter une application, il faut présenter à la JVM la classe principale (celle qui contient la méthode) en donnant le nom *de la classe*:
   java <nomClasse>.

Chercher à ne pas respecter ces règles et relevez les erreurs qui se produisent lors de la compilation et lors de l'exécution.
(Conseil : créer un ficher word qui contient deux tableaux. Mettez dans le premier tableau les erreurs de compilation et dans le deuxième les erreurs à l'exécution).

Exercice 2 : utiliser l'utilitaire appletviewer pour exécuter une applet

Voici le code source de l'applet BonjourApplet :

```
import java.awt.Graphics;
import java.applet.Applet;

public class BonjourApplet extends Applet {
  public void paint(Graphics g) {
    g.drawString("Bonjour tout le monde!",100 , 50);
  }
}
```

Compilez cette classe à l'aide de la commande : **javac BonjourApplet.java**

L'applet doit s'exécuter à l'intérieur d'une page HTML contenant un tag `<APPLET CODE=nom de la classe HEIGHT=... WIDTH=...> </APPLET>`

Source de la page BonjourApplet.html :

```
<HTML>
<BODY>
  Ici se trouve notre applet :
  <P>
  <APPLET CODE="BonjourApplet.class" WIDTH=200 HEIGHT=100>
    Votre Browser ne permet pas de visualiser les applets Java
  </APPLET>
  <P>
  Et voici du texte placé après la zone réservée à l'applet.
</BODY>
</HTML>
```

Testez avec l'appletviewer : **appletviewer BonjourApplet.html**

Ensuite, faites afficher la page en utilisant un navigateur (Internet Explorer ou Mozilla).

Exercice 3 : Créer une classe à partir d'un diagramme de classe UML

Voici la spécification en UML de la classe Livre :

Livre
-titre : String -auteur : String -nbPages : int {nbPages > 0}
+Livre() +Livre(auteurValue : String, titreValue : String) +Livre(auteurValue : String, titreValue : String, nbPagesValue : int) +getAuteur() : String +getTitre() : String +getNbPages() : int +setAuteur(auteurValue : String) : void +setTitre(titreValue : String) : void +setNbPages(nbPagesValue : int) : void +toString() : String +plusVolumineuxQue(l : Livre) : boolean +<<static>>plusVolumineuxQue(l1:Livre, l2:Livre) : boolean +aMemeAuteurQue(l : Livre) : boolean +<<static>>aMemeAuteurQue(l1 : Livre, l2 : Livre) : boolean

La méthode plusVolumineuxQue(l : Livre) : méthode d'instance qui compare le nombre de pages du livre courant avec celui du livre passé en argument.
La méthode plusVolumineuxQue(l1 : Livre, l2 : Livre) : méthode de classe qui compare le nombre de pages des deux livres passés en arguments.
La méthode aMemeAuteurQue(l : Livre) : méthode d'instance qui teste si le livre courant a le même auteur que celui passé en argument.
La méthode aMemeAuteurQue(l1 : Livre, l2 : Livre) : méthode de classe qui teste si les deux livres ont le même auteur.

Travail à faire :

1- Créer la classe Livre
2- Créer une classe TestLivre qui utilise chacun des constructeurs pour créer respectivement les trois livres ci-dessous :
   Livre l1 (Créé par le constructeur par défaut) :
      Titre : *Au Cœur de java* : Notions Fondamentales
      Auteur : *Horstman*
      NbPages : 820
   Livre l2 (Créé par le deuxième constructeur)
      Titre :Entreprise Java Bean
      Auteur : Richard Monson-Haefel
      NbPages : 568

   Livre l3 (Créé par le troisième constructeur)
      Titre : *Au Cœur de java* : Fonctions Avancées
      Auteur : *Horstman*
      NbPages : 950

3- Transformer votre méthode toString() en commentaire en la délimitant par /* et */, puis exécuter l'instruction `System.out.println(l1);`
4- Afficher le titre, l'auteur, et le nombre de pages du premier livre en utilisant l'instruction `System.out.println(l1);`
5- Comparer les nombres de pages des deux premiers livres en utilisant la méthode d'instance
6- Comparer les nombres de pages des deux premiers livres en utilisant la méthode de classe
7- Tester si Livre1 a le même auteur que Livre2 en utilisant la méthode d'instance
8- Tester si Livre1 a le même auteur que Livre3 en utilisant la méthode de classe
9- Calculer et afficher la somme des nombres de pages des deux premiers livres

Exercice 4: Manipulation des chaînes de caractères : Classe java.lang.String

Ecrire une classe **ManipuleString** qui permet d'effectuer les opérations suivantes:

A. Créez une variable de type **int**, affectez-lui une valeur, puis convertissez cette variable en chaîne (Ex: le nombre **123** devient **"123"**) .

B. Lire une string au clavier, puis lui enlever les éventuels blancs au début et à la fin, convertir tous ses caractères en majuscules puis afficher le résultat.

C. Lire deux chaînes s_1 et s_2 et tester si elles commencent par le même caractère.

D. Lire deux chaînes s_1 et s_2 et afficher les résultats renvoyés par les expressions « s_1 == s_2 »,

« s_1**.equals(**s_2**)**», « s_1**.compareTo(**s_2**)**» et , « s_1**.compareToIgnoreCase(**s_2**)**».

Tester avec s1=**"abcd"**et s2=**"abcd"**, puis s1=**"abcd"**et s2= **"AbcD"**.

E. Lire deux chaînes s_1 et s_2 et effectuer les tests suivants:

   a. «s_1 commence-t-elle par s_2 ? »,
   b. «s_1 finit-elle par s_2 ? »
   c. «s_1 contient-elle s_2 ? »

F. Lire deux chaînes s_1 et s_2 et si s_1 contient s_2, renvoyer s_1 privée de s_2 (s'intéresser à **substring**), sinon renvoyer s_1.

G. Si *s* est une chaîne de caractères, l'expression *s***.intern()**renvoie une chaîne ayant les mêmes caractères que *s* mais appartenant à une collection de chaînes *sans doublons* que Java maintient et dont les chaînes figurant dans le programme source font partie d'office.

Lire deux chaînes **s1**et **s2**et constater qu'à la suite des transformations « **s1 = s1.intern();**» et

« **s2 = s2.intern();**» les expressions « **s1.equals(s2)**» et « **s1 == s2**» deviennent équivalentes.

Annexe 1 : Classe java.util.Scanner

public final class **Scanner** extends Object implements Iterator<String>, Closeable

Exemple de programme

Pour plus d'imformation consulter l'API

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class TestScanner03 {

	public static void main(String[] args) {
	 try{
			Scanner stdin = new Scanner(System.in);
			System.out.print("Donner un entier de type byte(-128 à 127) :");
			byte b = stdin.nextByte();
			System.out.print("Donner un entier de type short (-32768 à 32767) :");
			short h = stdin.nextShort();
			System.out.print("Donner un entier de type int (-2^31 à 2^31-1) :");
			int n = stdin.nextInt();
			System.out.print("Donner un entier de type long (-2^63 à 2^63-1) :");
			long l = stdin.nextLong();
			System.out.print("Donner un réel de type float (utiliser , pour partie fractionnaire) :");
			float f = stdin.nextFloat();
			System.out.print("Donner un réel de type double (utiliser , pour partie fractionnaire) :");
			double d = stdin.nextDouble();
			System.out.println("Donner une suite de tokens séparés par des espaces :");
			String s1 = stdin.next(); //lit le premier token
			String s2 = stdin.nextLine(); //lit tout le reste de la ligne
			System.out.println("b = " + b + "\th = " + h + "\tn = " + n + "\tl = " + l);
			System.out.println("f = " + f + "\td = " + d);
			System.out.println("s1 = " + s1 + "\ts2 = " + s2);
			stdin.close();
	 }catch(InputMismatchException e){
			 System.out.println("Vous avez saisie une valeur erronée "
					 +"\nAttention : utiliser la virguel et non le point pour les nombres décimaux : " +
e);
	 }
	}
}
```

Exécution:

```
Donner un entier de type byte(-128 à 127) :10
Donner un entier de type short (-32768 à 32767) :20
Donner un entier de type int (-2^31 à 2^31-1) :30
Donner un entier de type long (-2^63 à 2^63-1) :40
Donner un réel de type float (utiliser , pour partie fractionnaire) :50,25
Donner un réel de type double (utiliser , pour partie fractionnaire) :125,75
Donner une suite de tokens séparés par des espaces :
abcd efgh ijklm
b = 10	h = 20	n = 30	l = 40
f = 50.25	d = 125.75
s1 = abcd	s2 =  efgh ijklm
```

Annexe 2 : Classe javax.swing.JOptionPane

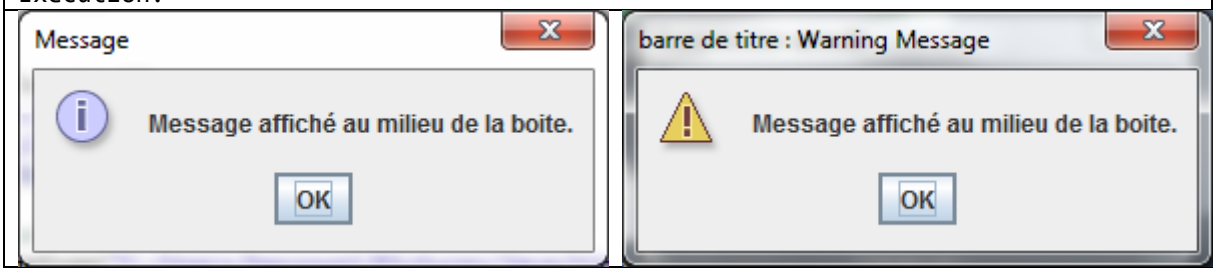
public class **JOptionPane** extends JComponent implements Accessible

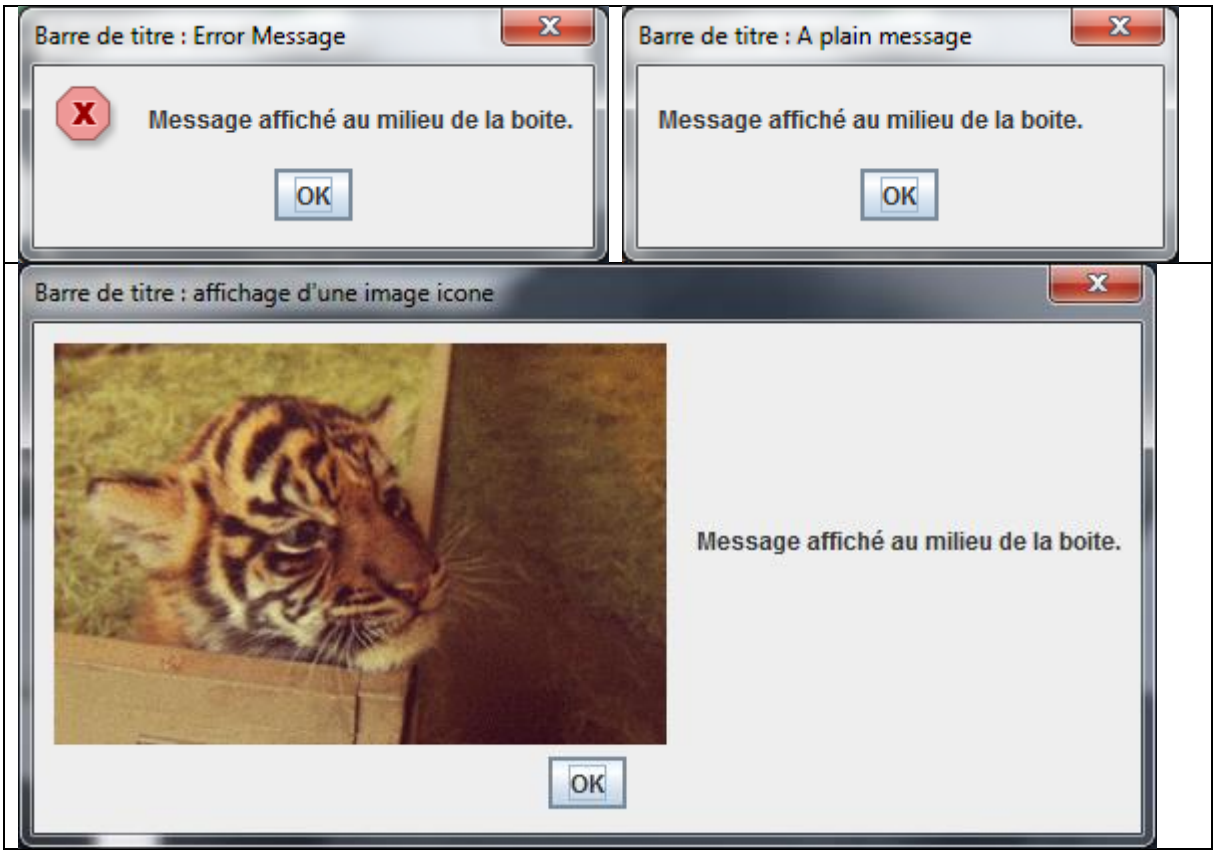
Exemple de programme
Pour plus d'imformation consulter l'API

Classe JOptionPane : Affichage de messages avec la méthode *showMessageDialog*

```
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class TestShowMessageDialog {
 public static void main(String[] args){
       //default title and icon
        JOptionPane.showMessageDialog(null,
            "Message affiché au milieu de la boite.");

       //custom title, warning icon
        JOptionPane.showMessageDialog(null,
            "Message affiché au milieu de la boite.",
            "barre de titre : Warning Message",
            JOptionPane.WARNING_MESSAGE);

       //custom title, error icon
        JOptionPane.showMessageDialog(null,
               "Message affiché au milieu de la boite.",
            "Barre de titre : Error Message",
            JOptionPane.ERROR_MESSAGE);

       //custom title, no icon
        JOptionPane.showMessageDialog(null,
             "Message affiché au milieu de la boite.",
            "Barre de titre : A plain message",
            JOptionPane.PLAIN_MESSAGE);

       //custom title, custom icon
        Icon tigerIcon = new
ImageIcon("C:/Users/hassouni/Pictures/java/tiger.gif");
        JOptionPane.showMessageDialog(null,
            "Message affiché au milieu de la boite.",
            "Barre de titre : affichage d'une image icone",
            JOptionPane.INFORMATION_MESSAGE,
            tigerIcon);
 }
}
```

Exécution:

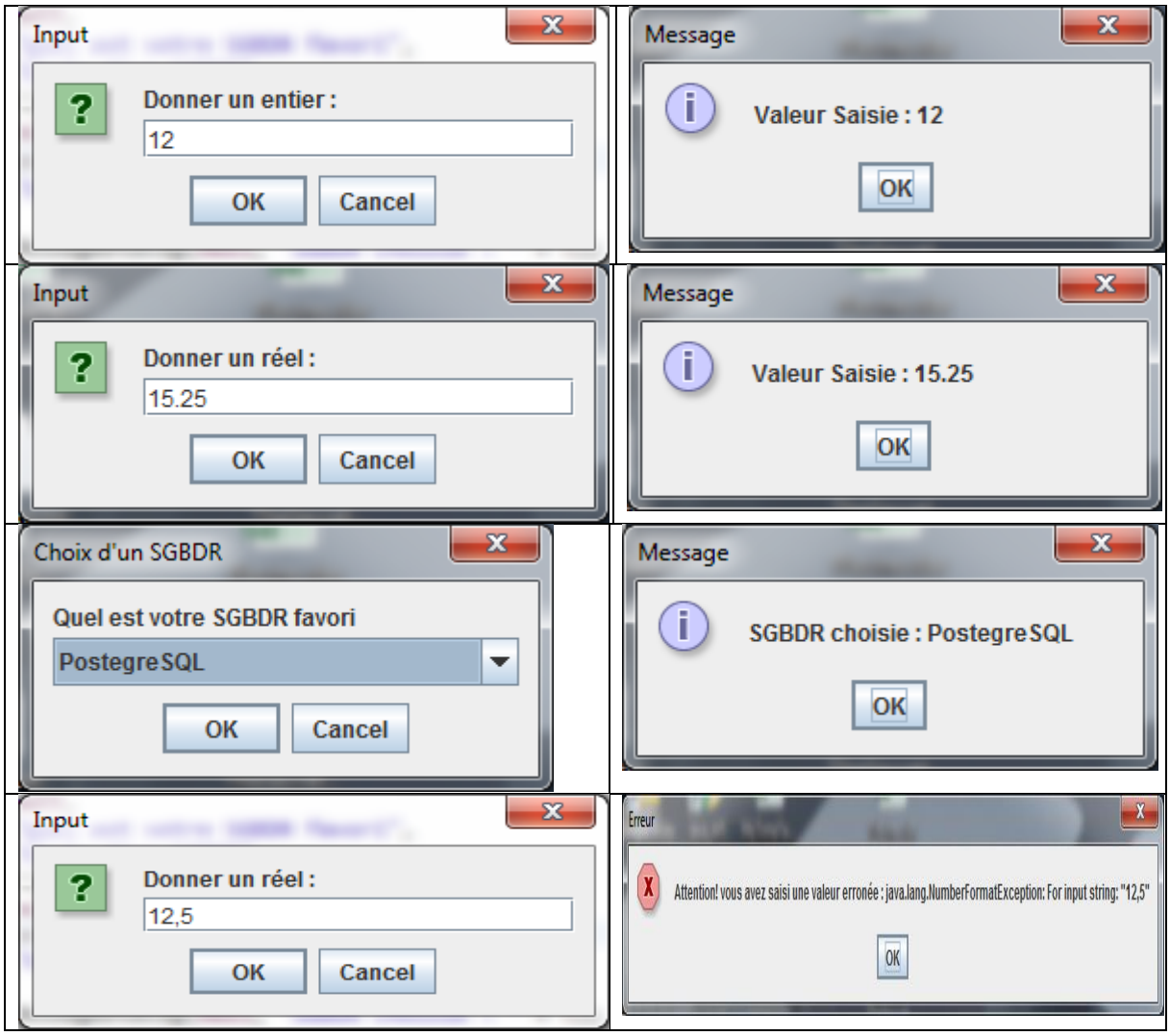




Classe JOptionPane : Lecture de donnée avec la méthode *showInputDialog*

```
import javax.swing.JOptionPane;
public class TestShowInputDialog {
	public static void main(String[] args) {
		try{
			int n = Integer.parseInt((String)JOptionPane.showInputDialog("Donner un entier : "));
			JOptionPane.showMessageDialog(null, "Valeur Saisie : " + n);

			double x = Double.parseDouble((String)JOptionPane.showInputDialog("Donner un réel : "));
			JOptionPane.showMessageDialog(null, "Valeur Saisie : " + x);

			Object[] possibilities = {"Oracle", "MySQL", "PostegreSQL"};
			String s = (String)JOptionPane.showInputDialog(
			    null,
			    "Quel est votre SGBDR favori",
			    "Choix d'un SGBDR",
			    JOptionPane.PLAIN_MESSAGE,
			    null,
			    possibilities,
			    "Oracle");

			JOptionPane.showMessageDialog(null, "SGBDR choisie : " + s);
		}catch(Exception e){
			JOptionPane.showMessageDialog(null, "Attention! vous avez saisi une valeur erronée : " +
e, "Erreur", JOptionPane.ERROR_MESSAGE);
		}
	}
}
```

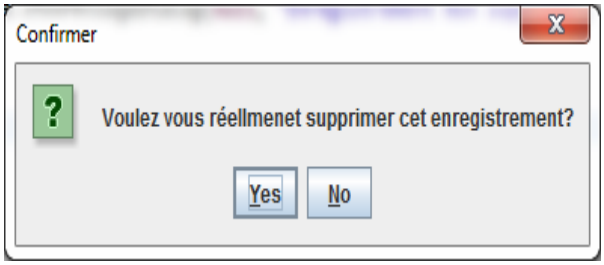
Exécution:

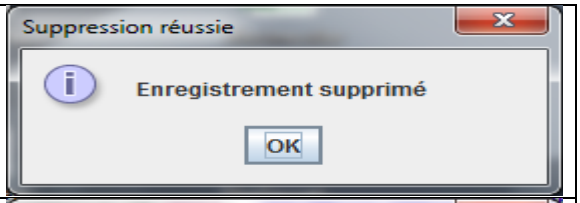


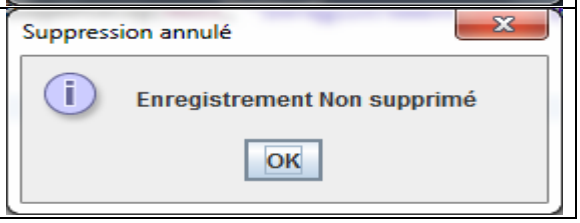
Classe JOptionPane : Lecture de donnée avec la méthode *showConfirmDialog*

```
import javax.swing.JOptionPane;
public class TestShowConfirmDialog {
        public static void main(String[] args) {
                int n = JOptionPane.showConfirmDialog(
                   null,
                  "Voulez vous réellmenet supprimer cet enregistrement?",
                  "Confirmer",
                  JOptionPane.YES_NO_OPTION);

                if (n==JOptionPane.YES_OPTION)
                        JOptionPane.showMessageDialog(null, "Enregistrement supprimé", "Suppression réussie",
JOptionPane.INFORMATION_MESSAGE);
                else
                        JOptionPane.showMessageDialog(null, "Enregistrement Non supprimé", "Suppression
annulé", JOptionPane.INFORMATION_MESSAGE);
        }
}
```

Exécution:



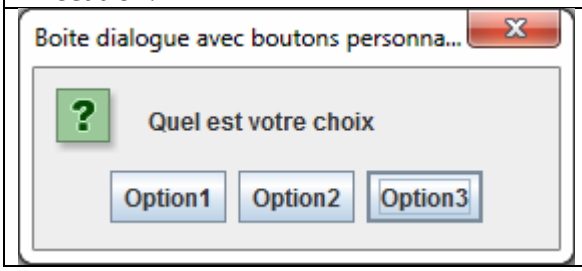


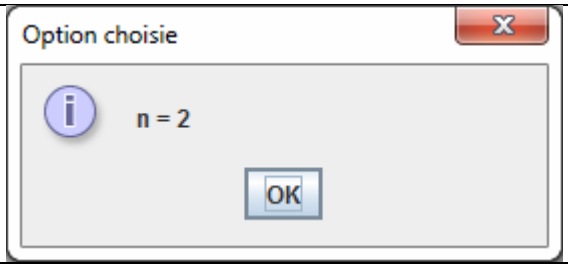


Classe JOptionPane : Lecture de donnée avec la méthode *showOptionDialog*

```
import javax.swing.JOptionPane;

public class TestShowOptionDialog {
	public static void main(String[] args){
		//Custom button text
		Object[] options = {"Option1", "Option2", "Option3"};

	int n = JOptionPane.showOptionDialog(null,
	    "Quel est votre choix ",
	    "Boite dialogue avec boutons personnalisé",
	    JOptionPane.YES_NO_CANCEL_OPTION,
	    JOptionPane.QUESTION_MESSAGE,
	    null,
	    options,
	    options[2]);

	JOptionPane.showMessageDialog(null, "n = "+ n, "Option choisie",
JOptionPane.INFORMATION_MESSAGE);
	}
}
```

Exécution:





Annexe 3 : Classe java.lang.String

public final class **String** extends Object implements Serializable, Comparable<String>, CharSequence

Modifier and Type	Method and Description
char	**charAt**(int index) Returns the char value at the specified index.
int	**compareTo**(**String** anotherString) Compares two strings lexicographically.
int	**compareToIgnoreCase**(**String** str) Compares two strings lexicographically, ignoring case differences.
boolean	**endsWith**(**String** suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(**Object** anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(**String** anotherString) Compares this String to another String, ignoring case considerations.
int	**indexOf**(int ch) Returns the index within this string of the first occurrence of the specified character.
int	**indexOf**(int ch, int fromIndex) Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
int	**indexOf**(**String** str) Returns the index within this string of the first occurrence of the specified substring.
int	**indexOf**(**String** str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
boolean	**isEmpty**() Returns true if, and only if, **length()** is 0.
String	**intern**() Returns a canonical representation for the string object.
int	**lastIndexOf**(int ch) Returns the index within this string of the last occurrence of the specified character.
int	**lastIndexOf**(int ch, int fromIndex) Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

int	**lastIndexOf**(**String** str) Returns the index within this string of the last occurrence of the specified substring.
int	**lastIndexOf**(**String** str, int fromIndex) Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
int	**length**() Returns the length of this string.
String	**replace**(char oldChar, char newChar) Returns a string resulting from replacing all occurrences ofoldChar in this string with newChar.
boolean	**startsWith**(**String** prefix) Tests if this string starts with the specified prefix.
boolean	**startsWith**(**String** prefix, int toffset) Tests if the substring of this string beginning at the specified index starts with the specified prefix.
String	**substring**(int beginIndex) Returns a string that is a substring of this string.
String	**substring**(int beginIndex, int endIndex) Returns a string that is a substring of this string.
char[]	**toCharArray**() Converts this string to a new character array.
String	**toLowerCase**() Converts all of the characters in this String to lower case using the rules of the default locale.
String	**toUpperCase**() Converts all of the characters in this String to upper case using the rules of the default locale.
String	**trim**() Returns a string whose value is this string, with any leading and trailing whitespace removed.
static **String**	**valueOf**(char[] data) Returns the string representation of the char array argument.
static **String**	**valueOf**(char[] data, int offset, int count) Returns the string representation of a specific subarray of thechar array argument.
static **String**	**valueOf**(long l) Returns the string representation of the long argument.