Licence Monnaie - Finance

Algorithmique et programmation java
Fascicule de TD

O. Auzende
auzende@orange.fr

Année 2012 – 2013

Modalités du cours

Programme : introduction à l'algorithmique ; mise en œuvre d'algorithmes élémentaires en langage de programmation java. Réalisations : programmes financiers sous forme d'applications java ou d'applets java.

Pré-requis :
- **Clavier** : maîtrise absolument impérative.
- **Windows** : manipulation de dossiers et fichiers, distinction entre types des fichiers (formats texte et RTF), installation de logiciels, configuration du poste de travail et des variables d'environnement.
- **Excel** : connaissance des fonctions logiques SI, ET, OU.

Cours : les lundis du second semestre, de 12 h à 13 h 30, en amphithéâtre 3 (centre Assas). Fascicules de cours (polycopiés à trous) distribués en amphithéâtre.

Groupes de TD : 8 groupes, chacun fonctionnant sur les 12 semaines du second semestre. Ce polycopié de travaux dirigés est distribué lors de la première séance de TD.

Jour	Heure	Salle	Enseignant
Mardi	18 h 50	salle 501	O. Auzende
Mardi	20 h 25	salle 501	O. Auzende
Mercredi	14 h 05	salle 503	N. Thibault
Mercredi	15 h 40	salle 503	N. Thibault
Vendredi	9 h 20	salle 502	F. Jézéquel
Vendredi	10 h 55	salle 502	F. Jézéquel
Vendredi	12 h 30	salle 502	N. Thibault
Vendredi	14 h 05	salle 502	O. Auzende

En **cours et TD**, avoir **TOUJOURS** les **fascicules de cours** sur soi, à jour.
En **TD**, avoir **le fascicule de TD** et disposer **impérativement** d'une **clé USB** standard

Evaluation : contrôle continu et projet

- Deux contrôles courts **C1** et **C2** (chacun sur 20 points) aux séances de **TD 5** et **8**.
- Un contrôle long **C3** (sur 30 points) à la séance de **TD 11**.
- Un projet **P** (sur 30 points) à réaliser en binôme. Rendu progressif obligatoire, du **TD 6** au **TD 11**. Soutenance lors de la séance de **TD 12**.

Note finale sur 10 : (C1 + C2 + C3 + P) / 10

Sommaire

TD n°1 : Commandes MS-DOS, compilation et interprétation de java

Récupération de fichiers sur le serveur

Recopiez dans *Mes documents* le dossier **java** qui se trouve sur **SERV_SAL** dans **t1-cdrom**.
→ **A la fin de la séance, vous recopierez sur votre clé USB l'ensemble du dossier.**

Initiation au MS-DOS

Le MS-DOS est le premier système d'exploitation de Microsoft, où l'on s'exprime en tapant des lignes de commande (analogue aux shells Unix). On en a besoin lorsqu'on souhaite réaliser des opérations n'exploitant pas Windows.

Le MS-DOS est accessible par : *Démarrer → Exécuter → cmd* ou par *Démarrer → Programmes → Accessoires → Invite de commandes* ou bien directement par une icône sur le bureau, si celle-ci existe**.**

Dans la fenêtre ainsi ouverte, l'interpréteur MS-DOS attend les commandes de l'utilisateur. Le prompt initial est normalement **C:\Documents and Settings\etud**.

1) Tapez la commande **dir**. Il s'agit de l'abréviation de « directory ».
Le dossier *Mes documents* apparaît.

2) Pour aller dans Mes documents, taper **cd "Mes documents".** A noter que cd est l'abréviation de « change directory » qui permet de changer de dossier courant. Remarquer le changement de prompt.

3) Aller dans le dossier **java** : comme le dossier **java** est dans *Mes documents*, il suffit de taper : **cd java** et valider. Remarquer le changement de prompt.

4) Regarder ce qui se trouve dans le dossier **java**.
dir liste tous les fichiers et les sous-dossiers du dossier courant
dir e*.* liste les fichiers et dossiers commençant par la lettre e.

5) Aller dans le dossier **programme** : comme on est déjà dans le dossier **java**, il suffit de taper : **cd programme**. Remarquer encore le changement de prompt.

6) Regarder ce qui se trouve dans le dossier **programme** : taper **dir**

7) Revenir dans le dossier **java** : taper **cd ..**

8) Créer un dossier **test** dans le dossier **java** : il s'agit de la commande **md** (abréviation de « make directory ») donc taper : **md test**

9) Recopier le fichier emploi.doc du dossier **java** dans le dossier **test** : il s'agit de la commande **copy** donc taper : **copy emploi.doc test**
Le système doit vous répondre « 1 fichier(s) copié(s) »

10) Vérifier le contenu du dossier **test** : taper **dir test**

11) Aller dans le dossier **test** : **cd test**

12) Renommer emploi.doc en truc.doc : il s'agit de la commande **rename**. Taper : **rename emploi.doc truc.doc**

13) Supprimer le fichier truc.doc. Il s'agit de la commande **del**, abréviation de delete. Taper : **del truc.doc**

14) Revenir dans le dossier **java** et supprimer le dossier **test**. Il s'agit de la commande **rd**, abréviation de « remove directory ». Taper : **rd test**

Compilation, exécution, modification d'une application java (1)

1) Ouvrir avec le traitement de texte WordPad ou avec un éditeur de script le fichier **Programme.java** (attention, **première lettre en majuscule**) qui se trouve dans le dossier **programme**.
Comme tous les fichiers source java, c'est un fichier sauvegardé en format texte uniquement. Son contenu est le suivant :

```
public class Programme {
          public static void main(String args[]) {
                int a=5;
                int b=7;
                System.out.println("a vaut : " + a);
                System.out.println("b vaut : " + b);
                }
          }
```

2) Dans la fenêtre MS-DOS, taper la commande **PATH** et valider. Vérifier que le PATH contient le chemin menant à un compilateur java, c'est-à-dire contient un chemin du type JDK….\BIN.

Rôle du PATH ? Rechercher sur Internet et noter ci-dessous le résultat.
……..
……..
……..
……..
……..
……..

3) Se placer dans le dossier **programme**. Taper la commande : **javac Programme.java** (attention à la casse !). Si la compilation se passe bien, un fichier de nom **Programme.class** est généré par le compilateur (vérifier avec la commande **dir**). Sinon, il faut corriger la commande et recompiler.

Qu'appelle-t-on la casse ?
……..
……..
……..

4) Toujours dans la fenêtre MS-DOS, faire exécuter le fichier par la commande **java Programme** (sans l'extension .class). L'interpréteur java lance automatiquement la méthode main.

Affichage obtenu :
……..
……..

5) Ajouter à la fin du main de la classe Programme les lignes permettant de déclarer deux entiers c et d, de calculer c=a+b et d=a*b, puis de faire afficher les valeurs de c et de d. Noter les lignes ajoutées :
……..
……..
……..
……..
……..
……..
……..
Compiler et faire exécuter le fichier ainsi modifié.

6) On voudrait à présent que a, b, c et d puissent être des **réels** (de type primitif **double**) et non plus des entiers. Modifier la classe Programme en conséquence, en prenant cette fois a=4.56 et b=-8.65. Compiler le fichier **Programme.java** ainsi modifié et le faire exécuter.

7) On veut faire ajouter une ligne en fin de programme faisant afficher « valeur entiere de d : » puis la partie entière de d. Utiliser un opérateur de cast pour obtenir ce résultat. Noter la(les) ligne(s) ajoutée(s) :
……..
……..

Compiler le fichier **Programme.java** ainsi modifié et le faire exécuter.

Compilation, exécution, modification d'une application java (2)

Ouvrir avec le traitement de texte WordPad ou avec un éditeur de script le fichier **Saisie.java** qui se trouve dans le dossier **saisie**. Son contenu est le suivant :

```
import java.util.Scanner ;
public class Saisie {
	static Scanner sc ;
	public static void main(String args[]) {
		System.out.println("Veuillez saisir un entier :");
		sc = new Scanner(System.in);
		int a = sc.nextInt();
		System.out.println("Veuillez saisir un autre entier :");
		int b = sc.nextInt();
		System.out.println("Vous avez saisi les nombres " + a + " et "+b);
		}
	}
```

Ne pas se préoccuper pour l'instant de la classe Scanner. C'est une classe utilitaire qui permet de récupérer aisément ce qui est tapé au clavier (on la verra plus tard).

1) Compiler puis exécuter plusieurs fois ce programme.

2) Ajouter à la fin de la méthode main deux lignes faisant afficher respectivement a*b puis a/b. Compiler et exécuter.

A noter : il est indispensable de préciser, par un opérateur de cast, que le résultat de a/b est un float. Sinon le programme affiche le quotient entier de a par b.

Compilation et exécution d'une applet java (1)

Dans la fenêtre d'invite de commandes, se placer dans le sous-dossier **applet** du dossier **java**. Regarder son contenu avec la commande **dir**.

Le dossier contient :
- un fichier **Geometrique.java** qui est le code source d'une **applet**
- un fichier **Exemple.html** qui est une page HTML très simple, chargeant l'applet.

Taper la commande **javac Geometrique.java** pour compiler l'applet. Regarder avec **dir** quels sont les fichiers générés et contrôler leur date et heure.

Une applet ne s'exécute pas de la même manière qu'une application.

Avec un explorateur, regarder le contenu du dossier **applet** et faire afficher le fichier **Exemple.html** par un navigateur (selon la configuration de la machine, soit double-cliquer sur le nom du fichier, soit cliquer sur le bouton droit de la souris, sélectionner « Ouvrir avec… » et choisir un navigateur : Internet Explorer, Mozilla, Chrome, etc.).

Le navigateur affiche la page HTML contenant l'applet. Regarder ce que fait cette applet en jouant avec la souris.

Compilation et exécution d'une applet java (2)

Dans la fenêtre d'invite de commandes, se placer dans le sous-dossier **dessin** du dossier **java**. Regarder son contenu avec la commande **dir**.
Le dossier contient :
- un fichier **Dessin.java** qui est le code source d'une **applet**
- un fichier **DessinExemple.html** qui est une page HTML très simple, chargeant l'applet.

Taper la commande **javac Dessin.java** pour compiler l'applet. Regarder avec **dir** que le sous-dossier a été complété par des fichiers .class à la date du jour.

Ouvrir le fichier **DessinExemple.html** avec un navigateur. Regarder ce que fait cette applet en jouant avec la souris. Fermer ensuite le navigateur.

Sauvegarde des fichiers avec l'explorateur Windows

Avec l'explorateur**,** recopier **tout le dossier java** de *Mes documents* sur votre clé USB.

Règles de sauvegarde à respecter impérativement

Disposer d'une clé USB.

A la fin du **premier TD**, recopier tout le dossier java de *Mes documents* sur votre clé USB.

Au début de chaque TD suivant, afin de repartir de vos résultats précédents, faire l'opération inverse : recopier le dossier java de votre clé USB vers *Mes documents*.

A la fin de chaque TD suivant, recopier de *Mes documents* sur votre clé USB les sous-dossiers qui ont été modifiés en cours de TD.

Installation de java sur votre machine personnelle

Il est indispensable de disposer très rapidement sur votre machine personnelle du compilateur java, de l'interpréteur java, des librairies associées et de la documentation.

L'ensemble compilateur java, interpréteur java et librairies associées forme le JDK (Java Development Kit).
La documentation est un fichier ZIP séparé.

L'ensemble est **libre de droits** et **téléchargeable** sur le site de l'entreprise SUN.

Si vous avez un **PC sous Windows**, suivre les indications du fichier **emploi.pdf** fourni dans le dossier **java** pour installer le plus rapidement possible le JDK et la documentation.

Vous en aurez très rapidement besoin pour préparer vos contrôles et faire le projet.

Si vous avez un **PC sous Linux** ou un **Mac**, **tout est déjà installé**. Lire la fin du fichier **emploi.pdf** fourni pour savoir comment on compile du java et comment on l'exécute.

Linux et Mac sont similaires, sauf utilisation du clavier pour les touches { } [ ]

TD n°2 : Tests, boucles, sauts

Exercice 1

1) Ouvrir le fichier **Tableau1.java** du dossier **tableau1**, compiler le fichier et le faire exécuter.

```
public class Tableau1 {

	public static void main(String[] args) {
		double[] t ;
		int n = 20 ;
		t = new double[n] ;

		for (int i = 0 ; i < n ; i++) {
			t[i] = i ;                              (1)
			}

		for (int i = 0; i < n ; i++) {
			System.out.print(t[i] + " * ");
			}

		System.out.println("");
		}
	}
```

Que fait la première boucle for ?

…….

Quel est l'affichage résultant de la seconde boucle for ?

…….

2) Modifier la ligne (1) de telle sorte que le tableau ne soit plus initialisé avec 0, 1, …, 19 mais soit initialisé avec les multiples de 10 : 0, 10, 20, 30, 40, … 190. Noter l'instruction modifiée :

…….

Compiler et exécuter.

Exercice 2

1) Compiler le fichier **Tableau1bis.java** du dossier **tableau1**. Le faire exécuter en entrant successivement les 10 entiers 12 45 9 6 44 23 56 78 56 3 séparés par un espace, puis valider.

2) Ouvrir le fichier avec un éditeur :

```
import java.util.Scanner ;
public class Tableau1bis {

	public static void main(String[] args) {
		int[] t ;
		int n = 10 ;
		t = new int[n] ;
		Scanner sc = new Scanner(System.in);
		for (int i = 0 ; i < n ; i++) {
			t[i] = sc.nextInt();
			}

		System.out.println("Saisie terminee");
		for (int i = 0; i < n ; i++) {
			System.out.print(t[i] + " * ");
			}
		System.out.println("");          (2)
		}
	}
```

Que fait la première boucle for ?
……

3) Ajouter juste après la ligne (2) une boucle **while** pour afficher les éléments du tableau t **tant que ceux-ci sont inférieurs à 50**. Noter les instructions ajoutées :
……
……
……

Compiler et exécuter en entrant les 10 entiers 12 45 9 6 44 23 36 28 49 3 séparés par un espace. Si une erreur apparaît (**OutOfBoundsException** : dépassement de la taille d'un tableau), corrigez la boucle while.

Exercice 3

1) Ouvrir le fichier **Tableau2.java** du dossier **tableau2**, compiler le fichier et le faire exécuter plusieurs fois de suite.

```
public class Tableau2 {

      public static void main(String[] args) {
            int n=20 ;
            int[] t = new int[n] ;
            java.util.Random r = new java.util.Random();        (1)
            for (int i = 0 ; i < n ; i++) {
                  t[i] = r.nextInt(50) ;                        (2)
                  }

            for (int i = 0; i < n ; i++) {
                  System.out.print(t[i] + " * ");
                  }
            System.out.println("");
            ...                                                 (3)
            }
      }
```

Ce programme fait appel à un **générateur de nombres aléatoires** appelé r, qui est créé par la ligne (1) :
`java.util.Random r = new java.util.Random();`

Le générateur r génère successivement 20 nombres entiers tous compris entre 0 et 50 car à la ligne (2) `r.nextInt(50)` fournit à chaque fois un **entier** aléatoire **compris entre 0 et 50** (0 compris, 50 non compris).

2) Ajouter à partir de la ligne (3) une boucle **do … while** pour faire afficher **le premier élément** du tableau tab, puis les éléments suivants **tant que ceux-ci sont inférieurs ou égaux au premier élément**.

Exemples :
Si t commence par 24 12 35 8 45 67 78 5 12 …, la boucle do… while doit afficher 24 12
Si t commence par 48 12 35 8 45 67 78 5 12 …, la boucle do… while doit afficher 48 12 35 8 45
Si t commence par 36 52 35 8 45 67 78 5 12 …, la boucle do… while doit afficher 36

Noter les lignes ajoutées :
……
……
……
……
……

Compiler et exécuter.

Exercice 4

1) Ouvrir le fichier **Tableau3.java** du dossier **tableau3**, compiler le fichier et le faire exécuter plusieurs fois de suite.

```
public class Tableau3 {

	public static void main(String[] args) {
		int n = 10 ;
		float[] t = new float[n] ;
		java.util.Random r = new java.util.Random();
		for (int i = 0 ; i < n ; i++) {
			t[i] = r.nextFloat() ;                          (1)
			}
		for (int i = 0; i < n ; i++) {
			System.out.println(t[i]);
			}
		System.out.println("");
		...                                                 (2)
		}
	}
```

Le générateur r génère successivement 10 nombres réels de type float, tous compris entre 0 et 1 car à la ligne (1) `r.nextFloat()` fournit à chaque fois un **réel** aléatoire de type float **compris entre 0 et 1**

2) Ajouter à partir de la ligne (2) les lignes permettant de faire afficher les éléments du tableau t **tant que ceux-ci sont inférieurs à 0.8** (utiliser un **break**). Noter les lignes ajoutées :

……
……
……
……

Compiler et exécuter.

3) Ajouter à la fin de la méthode main une **autre** boucle for permettant de faire afficher les éléments du tableau t **sauf ceux compris entre 0.5 et 0.8** (utiliser un **continue**). Noter les lignes ajoutées :

……
……
……
……

Compiler et exécuter.

4) Recopier les dossiers **tableau1**, **tableau2** et **tableau3** sur votre clef USB.

TD n°3 : Méthodes et tableaux

Exercice 1 : méthodes de classe et d'instances

Se positionner dans le dossier **progobjet1**. Compiler le fichier **ProgObjet1.java** puis le faire exécuter. Noter le résultat de son exécution :

……
……
……
……
……
……
……

Le code de ce programme est le suivant :

```
class ObjetTableau {                    // début de la classe
       static int nbobjets = 0;
       int n ;
       int[] t ;

       ObjetTableau(int a) {            // le constructeur
              n = a;                    // n prend la valeur a
              t = new int[n] ;          // on réserve la place pour le tableau t
              for (int i=0;i<n;i++) {   // remplissage du tableau
                     t[i] = i+1 ;
                     }
              nbobjets++;
              }                         // fin du constructeur

       void affiche() {
              System.out.println("affichage d'un objet");
              System.out.println("n vaut "+n);
              for (int i = 0; i < n; i++){
                     System.out.print(t[i]+" * ");
                     }
              System.out.println("");
              }

       static void nombreObjets() {
              System.out.println("Nombre d'objets : " + nbobjets) ;
              }

       }                                // fin de la classe


public class ProgObjet1 {               // début de classe
       public static void main (String args[ ]) {     // le main
              ObjetTableau.nombreObjets() ;
              ObjetTableau obj1 = new ObjetTableau(6) ;
              obj1.affiche();
              ObjetTableau.nombreObjets() ;
              ObjetTableau obj2 = new ObjetTableau(3) ;
              obj2.affiche();
              ObjetTableau.nombreObjets() ;
              }                         // fin du main
       }                                // fin de la classe
```

Dessiner ci-dessous l'allure générale d'un objet de la classe ObjetTableau :

Noter dans le tableau suivant si la méthode appelée est une méthode de classe ou d'instance, et préciser quel est l'affichage correspondant :

instruction	méthode : classe ou instance ?	affichage
`ObjetTableau.nombreObjets();`		
`ObjetTableau obj1=new ObjetTableau(6);`		
`obj1.affiche();`		
`ObjetTableau.nombreObjets();`		
`ObjetTableau obj2=new ObjetTableau(3);`		
`obj2.affiche();`		
`ObjetTableau.nombreObjets();`		

Exercice 2 : insertion d'un élément à la fin d'un tableau

1) Se positionner dans le dossier **progobjet2**. Compiler le fichier **ProgObjet2.java** puis le faire exécuter. On obtient :

```
affichage de obj1
nb vaut 6 et n vaut 0
Elements du tableau :            // vide pour le moment
fin de l'objet

affichage de obj2
nb vaut 4 et n vaut 0
Elements du tableau :            // vide pour le moment
fin de l'objet
```

2) Ouvrir le fichier **ProgObjet2.java** :

```
class NouveauTableau {          // début de la classe
      int nb ;
      int n ;
      int[] t ;

      NouveauTableau(int a) {   // le constructeur
            nb = a;             // n prend la valeur a
            t = new int[nb] ;   // on réserve la place pour le tableau t
            n = 0 ;
            }                   // fin du constructeur

      void affiche() {
            System.out.println("nb vaut " + nb + " et n vaut " + n);
            System.out.println("Elements du tableau : ");
            for (int i = 0; i < n; i++){
                  System.out.print(t[i]+" * ");
                  }
            System.out.println("fin de l'objet\n");
            }

      void insere(int x) {
            ...
            }
      }
```

```
public class ProgObjet2 {
	public static void main (String args[ ]) {
		NouveauTableau obj1 = new NouveauTableau(6) ;
		obj1.insere(3);
		obj1.insere(7);
		obj1.insere(4);
		obj1.insere(1);
		obj1.insere(5);
		System.out.println("affichage de obj1");
		obj1.affiche();
		NouveauTableau obj2 = new NouveauTableau(4) ;
		obj2.insere(8);
		obj2.insere(6);
		obj2.insere(9);
		obj2.insere(3);
		obj2.insere(2);
		System.out.println("affichage de obj2");
		obj2.affiche();
		}
	}
```

Dessiner l'allure générale d'un objet de la classe NouveauTableau :

Compléter la méthode `void insere(int x)` afin d'insérer **à la fin du tableau t** l'entier x (algorithme vu en cours). Attention à bien faire afficher un message d'erreur si une insertion n'est plus possible (lorsque le tableau est plein). Compiler et faire exécuter le fichier.

Exercice 3 : maximum, minimum, indices

1) Ouvrir le fichier **ObjetTableau1.java** du dossier **objettableau1**. Dessiner l'allure générale d'un objet de la classe Tableau :

Compiler le fichier et le faire exécuter. **Que fait ce programme ?**
……
……
……

2) Ajouter à la fin de la classe Tableau deux méthodes d'instance `int maximum()` et `int minimum()`(algorithmes vu en cours) renvoyant respectivement le plus grand et le plus petit élément du tableau t (regarder ci-dessous **où faire cet ajout**).

```
class Tableau {           // début de la classe
	int n ;
	int[] t ;

	Tableau(int a) {    // le constructeur
		n = a;       // n prend la valeur a
		t = new int[n] ;    // on réserve la place pour le tableau t
		java.util.Random r = new java.util.Random() ;
		for (int i = 0; i < n; i++){
			t[i] = r.nextInt(50);
			}
		}                  // fin du constructeur
```

```
	void affiche() {
		System.out.println("affichage d'un tableau");
		System.out.println("n vaut " + n);
		for (int i = 0; i < n; i++){
			System.out.print(t[i]+" * ");
			}
		System.out.println("\n");
		}

	int maximum() {
		...
		}

	int minimum() {
		...
		}
	}
```

Pour tester les méthodes, compléter **la méthode main** de la classe **ObjetTableau1** de la manière suivante puis compiler et exécuter le programme :

```
public static void main (String args[ ]) {

	Tableau obj1 = new Tableau(30) ;
	obj1.affiche();
	System.out.println("max: "+obj1.maximum()+" min: "+obj1.minimum());

	Tableau obj2 = new Tableau(50) ;
	obj2.affiche();
	System.out.println("max: "+obj2.maximum()+" min: "+obj2.minimum()) ;
	}
```

3) Ajouter à la fin de la classe Tableau, **après** les deux méthodes `int maximum()` et `int minimum()`, deux méthodes d'instance `int indiceMaximum()` et `int indiceMinimum()` renvoyant respectivement **l'indice** du plus grand et **l'indice** du plus petit élément du tableau t (algorithme vu en cours) :

```
	int indiceMaximum() {
		...
		}

	int indiceMinimum() {
		...
		}
```

Pour tester les méthodes, compléter **la méthode main** de la classe **ObjetTableau1** de la manière suivante puis compiler et exécuter le programme :

```
public static void main (String args[ ]) {

	Tableau obj1 = new Tableau(30) ;
	obj1.affiche();
	System.out.println("max: "+obj1.maximum()+" min: "+obj1.minimum());
	System.out.println("Indice du maximum : "  + obj1.indiceMaximum()) ;
	System.out.println("Indice du minimum : " + obj1.indiceMinimum()) ;

	Tableau obj2 = new Tableau(50) ;
	obj2.affiche();
	System.out.println("max: "+obj2.maximum()+" min: "+obj2.minimum()) ;
	System.out.println("Indice du maximum : "  + obj2.indiceMaximum()) ;
	System.out.println("Indice du minimum : " + obj2.indiceMinimum()) ;
	}
```

4) Recopier les dossiers **progobjet1**, **progobjet2** et **objettableau1** sur votre clef USB.

TD n°4 : Algorithmique (1)

Exercice 1 : somme, comptages, requêtes

1) Reprendre le fichier **objettableau1.java** de la séance précédente et ajouter à la fin de la classe Tableau une méthode d'instance **`int somme()`** renvoyant **la somme** des éléments du tableau t (algorithme vu en cours).

Pour tester cette méthode, modifier la classe ObjetTableau1 en ajoutant à la fin de la méthode main les lignes suivantes :

```
System.out.println("Somme obj1 : " + obj1.somme()) ;
System.out.println("Somme obj2 : " + obj2.somme()) ;
```

2) Ajouter à la fin de la classe Tableau une méthode d'instance **`int egal(int p)`** comptant **le nombre d'éléments** de t qui sont **égaux à p** (algorithme vu en cours).

Pour tester cette méthode, ajouter à la fin de la méthode main les lignes :

```
System.out.println("obj1 : Nb d'elements egaux a 50 : " + obj1.egal(50));
System.out.println("obj1 : Nb d'elements egaux a 60 : " + obj1.egal(60));
System.out.println("obj2 : Nb d'elements egaux a 20 : " + obj2.egal(20));
System.out.println("obj2 : Nb d'elements egaux a 80 : " + obj2.egal(80));
```

3) Ajouter à la fin de la classe Tableau une méthode d'instance **`int compris(int p, int q)`** comptant **le nombre d'éléments** de t qui sont **strictement compris entre p et q (p < q)**.

Rappel : le « ET » entre clauses logiques s'écrit &&.

Pour tester cette méthode, compter combien d'éléments de obj1 sont strictement compris entre 10 et 30 en ajoutant à la fin de la méthode main la ligne :

```
System.out.println("obj1: entre 10 et 30: " + obj1.compris(10,30));
```

4) Ajoutez à la fin de la classe Tableau une méthode d'instance **`int premier(int p)`** renvoyant le **premier** élément de t **strictement supérieur à p**.

Pour tester cette méthode, déterminez le premier élément de obj1 strictement supérieur à 60 en ajoutant à la fin de la méthode main la ligne :

```
System.out.println("obj1 : premier element > 60: "+obj1.premier(60));
```

5) Ajoutez à la fin de la classe Tableau une méthode d'instance **`int dernier(int p)`** renvoyant le **dernier** élément de t **strictement inférieur à p**.

Pour tester cette méthode, déterminez le dernier élément strictement inférieur à 10 en ajoutant à la fin de la méthode main la ligne :

```
System.out.println("obj1 : dernier element < 10: "+obj1.dernier(10));
```

Exercice 2 : insertion d'un élément dans un tableau déjà rangé

Ouvrir le fichier **ObjetTableau2.java** qui se trouve dans le dossier **objettableau2**.
Son contenu est le suivant :

```
class Tableau {                  // début de la classe
      int nb ;                   // trois attributs : n, nb et t
      int n ;
      int[] t ;

      Tableau(int a) {           // le constructeur
            nb = a;              // n prend la valeur a
            t = new int[nb] ;    // on réserve la place pour le tableau t
            n = 0 ;
            }                    // fin du constructeur
```

```
	void affiche() {
		System.out.println("affichage d'un objet");
		System.out.println("nb vaut " + nb + " et n vaut " + n);
		for (int i = 0; i < n; i++){
			System.out.print(t[i]+" * ");
			}
		System.out.println("");
		}

	void insererange(int x) {
		...
		}
	}

public class ObjetTableau2 {
	public static void main (String args[ ]) {
		Tableau obj1 = new Tableau(8) ;
		obj1.insererange(3);
		obj1.insererange(7);
		obj1.insererange(4);
		obj1.insererange(1);
		obj1.insererange(5);
		obj1.insererange(2);
		obj1.insererange(9);
		obj1.insererange(8);
		obj1.affiche();
		Tableau obj2 = new Tableau(5) ;
		obj2.insererange(8);
		obj2.insererange(6);
		obj2.insererange(9);
		obj2.insererange(3);
		obj2.insererange(2);
		obj2.affiche();
		}
	}
```

Traduire en java dans la méthode `void insererange(int x)` l'algorithme vu en cours afin d'insérer l'entier x à sa place dans le tableau.

Recopier les deux dossiers **objettableau1** et **objettableau2** sur votre clef USB.

TD n°5 : Algorithmique (2)

Exercice 1 : tri tournoi

1) Ouvrir le fichier **ObjetTableau3.java** qui se trouve dans le dossier **objettableau3**. Dessiner l'allure générale d'un objet de la classe Tableau :

Compiler et exécuter plusieurs fois ce fichier. **Que fait actuellement ce programme ?**
……
……

2) Ajouter la méthode `void tri1()`implémentant le **tri tournoi** (algorithme vu en cours) et modifier le main de la classe ObjetTableau3 afin de tester la méthode tri1() sur obj1 (voir ci-dessous) ; compiler et exécuter le fichier.

```
class Tableau {                  // début de la classe
      int n ;
      int[] t ;

      Tableau(int a) {           // le constructeur
            n = a;       // n prend la valeur a
            t = new int[n] ;    // on réserve la place pour le tableau t
            java.util.Random r = new java.util.Random() ;
            for (int i = 0; i < n; i++){
                  t[i] = r.nextInt(50);
                  }
            }                    // fin du constructeur

      void affiche() {
            for (int i = 0; i < n; i++){
                  System.out.print(t[i]+" * ");
                  }
            System.out.println("\n");
            }

      void tri1() {
            ...
            }
      }

public class ObjetTableau3 {
      public static void main (String args[ ]) {
            System.out.println("Tableau 1");
            Tableau obj1 = new Tableau(80) ;
            obj1.affiche();
            obj1.tri1() ;
            System.out.println("Tableau 1 apres tri1");
            obj1.affiche();
            System.out.println("Tableau 2");
            Tableau obj2 = new Tableau(60) ;
            obj2.affiche();
            }
      }
```

Exercice 2 : tri à bulles

1) Ajouter la méthode `void tri2()`implémentant le **tri à bulles** (algorithme vu en cours) ; modifier le main afin de tester la méthode tri2() sur obj2 (voir ci-dessous), puis compiler et exécuter le fichier.

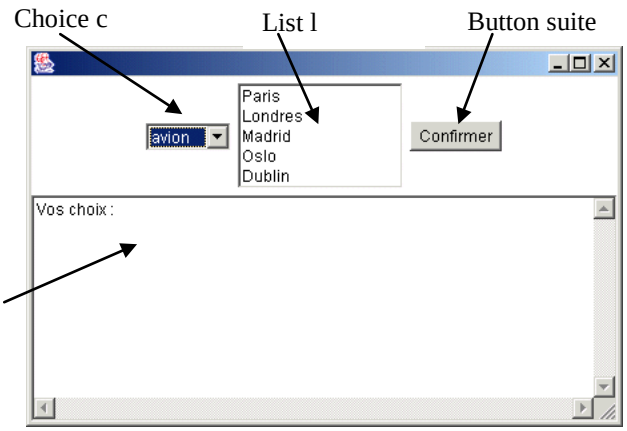
```
class Tableau {                  // début de la classe
	int n ;
	int[] t ;

	Tableau(int a) {           // le constructeur
		n = a;       // n prend la valeur a
		t = new int[n] ;    // on réserve la place pour le tableau t
		java.util.Random r = new java.util.Random() ;
		for (int i = 0; i < n; i++){
			t[i] = r.nextInt(50);
			}
		}                  // fin du constructeur

	void affiche() {
		for (int i = 0; i < n; i++){
			System.out.print(t[i]+" * ");
			}
		System.out.println("\n");
		}

	void tri1() {
		...
		}

	void tri2() {
		...
		}
	}

public class ObjetTableau3 {

	public static void main (String args[ ]) {
		System.out.println("Tableau 1");
		Tableau obj1 = new Tableau(80) ;
		obj1.affiche();
		obj1.tri1() ;
		System.out.println("Tableau 1 apres tri1");
		obj1.affiche();

		System.out.println("Tableau 2");
		Tableau obj2 = new Tableau(60) ;
		obj2.affiche();
		obj2.tri2() ;
		System.out.println("Tableau 2 apres tri2");
		obj2.affiche();
		}
	}
```

2) Recopier le dossier **objettableau3** sur votre clef USB.

TD n°6 : Création d'interfaces

Exercice 1

On souhaite obtenir l'interface ci-contre, comportant :
- un bouton de choix (de la classe **Choice**) appelé c
- une liste (de la classe **List**) appelé l
- un bouton (de la classe **Button**) appelé suite
- un champ de texte (de la classe **TextArea**) appelé texte.



1) Ouvrir le fichier **UtilAwt.java** qui se trouve dans le dossier **utilawt** :

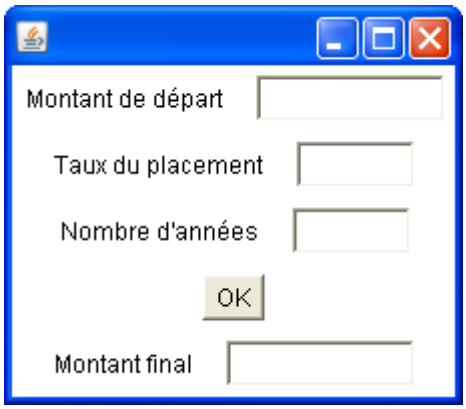
```
import java.awt.* ;

class Fenetre extends Frame {
	protected Panel p ;
	protected Choice c ;
	protected List l ;
	protected Button suite ;
	protected TextArea texte ;

	Fenetre() {		// le constructeur
		p = new Panel() ;
		c = new Choice() ;
		...		// ajout des items dans l'objet Choice
		p.add(c) ;
		l = new List(5) ;
		...		// ajout des items dans l'objet List
		p.add(l) ;
		...		// création du bouton suite
		p.add(suite) ;
		add("North", p) ;
		texte = new TextArea();
		...		// remplissage du champ de texte
		add("South", texte) ;
		}
	}

public class UtilAwt {
	public static void main(String args[]) {
		Fenetre f = new Fenetre() ;
		f.pack() ;
		f.show() ;
		}
	}
```

2) Compléter le constructeur `Fenetre()` afin d'aboutir au résultat escompté en suivant les instructions mises en commentaires.

Note : il est indispensable de regarder la documentation disponible (dans les salles d'Assas) à l'adresse **C:\appserver\sun\docs\doc\docs.** Regarder en particulier, dans le paquetage **java.awt**, les méthodes des classes **Choice**, **List**, **Button** et **TextArea**.

3) Compiler et exécuter cette application. Pour quitter l'application (l'événement de fermeture de la fenêtre n'étant pas pris en compte pour le moment), revenir dans la fenêtre d'invite de commandes et taper **CRTL C**.

Exercice 2



On veut créer l'interface ci-contre qui comporte :

- un Label pour le texte "Montant de départ"
- un TextField appelé dep pour la saisie du montant de départ
- un Label pour le texte "Taux du placement"
- un TextField appelé tx pour la saisie du taux
- un Label pour le texte "Nombre d'années"
- un TextField appelé nb pour la saisie du nombre d'années
- un Button b pour valider
- un Label pour le texte "Montant final"
- un TextField appelé res pour l'affichage du montant final

qui sont rangés dans des Panel pour améliorer la présentation.

Ouvrir le fichier **Fenetre.java** du dossier **fenetre** et remplir les Panel p1, p2, p3, p4, p5 afin d'arriver au résultat souhaité :

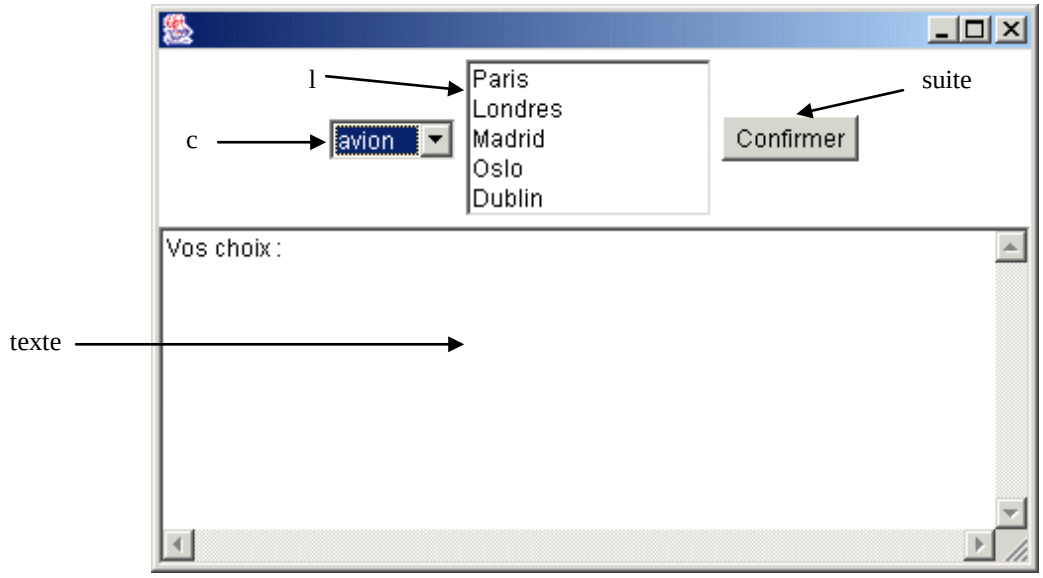
```
import java.awt.* ;
class MaFenetre extends Frame {
	protected Panel p, p1,p2,p3,p4,p5 ;
	protected Button b ;
	protected TextField dep, tx, nb, res ;

	MaFenetre() {
		setLayout(new BorderLayout()) ;
		p = new Panel() ;
		p.setLayout(new GridLayout(5, 1)) ;
		p1=new Panel();
		...
		p.add(p1);
		p2=new Panel();
		...
		p.add(p2) ;
		p3=new Panel();
		...
		p.add(p3);
		p4=new Panel();
		...
		p.add(p4);
		p5=new Panel();
		...
		p.add(p5) ;
		add("Center", p) ;
		}
	}

public class Fenetre {
	public static void main(String[] args) {
		MaFenetre f = new MaFenetre() ;
		f.pack() ;
		f.show() ;
		}
	}
```

Recopier les dossiers **utilawt** et **fenetre** sur votre clef USB.

TD n°7 : La gestion des événements

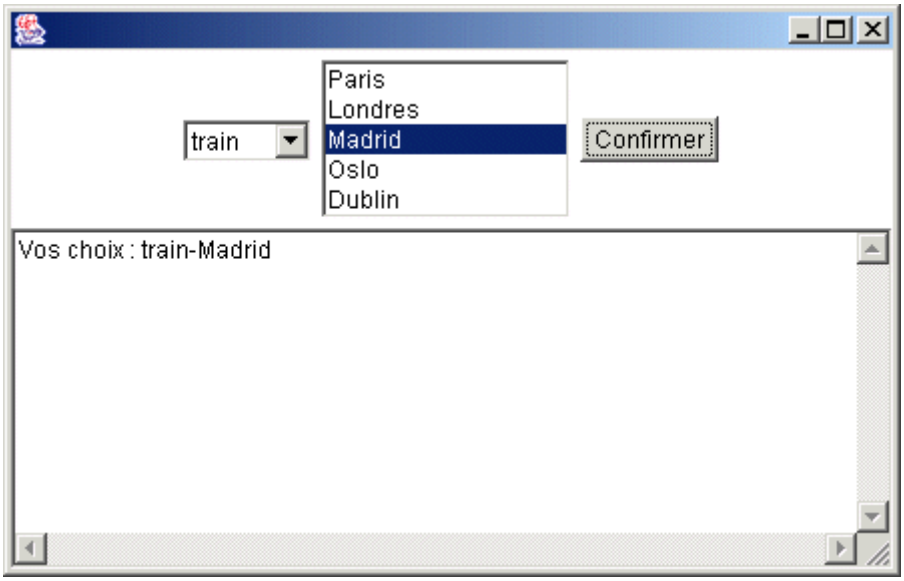
Exercice 1

L'interface graphique ci-dessous a été créée précédemment dans le fichier **UtilAwt.java** :



Compléter le fichier **UtilAwt.java** par un objet d'une classe Delegue et un objet d'une classe Adaptateur de telle sorte que le programme traite les événements suivants :

- cliquer sur la **case de fermeture** de la fenêtre ferme la fenêtre
- cliquer sur le bouton « Confirmer » permet de faire afficher dans la zone de texte les options choisies, comme le montre l'illustration suivante :



<u>La démarche à suivre (décrite complètement en cours)</u>
<u>est rappelée sommairement au verso.</u>

Démarche à suivre

1) Ajouter un deuxième import pour gérer les événements :

```
import java.awt.* ;
import ...
```

2) Ajouter deux attributs d'instance à l'interface :

```
protected ...
protected ...
```

3) Compléter le constructeur Fenetre() en créant le délégué et l'adaptateur par :

```
delegue = ...
adapt = ...
```

4) Ecrire le début des classes Delegue et Adaptateur :

```
class Delegue {
   protected Fenetre fen ;

   Delegue (Fenetre f) {
          ...
          }
   }

class Adaptateur {
   protected Delegue delegue ;

   Adaptateur (Delegue d) {
          ...
          }
   }
```

COMPILER POUR VERIFIER QU'IL N'Y A PAS D'ERREUR JUSQUE LA

5) Modifier la déclaration de la classe Adaptateur en implémentant les deux listeners :

```
class Adaptateur implements WindowListener, ActionListener {
```

6) Enregistrer l'adaptateur dans le constructeur Fenetre() sur les éléments à surveiller :

```
this.add...
suite.add...
```

7) Ajouter à la classe Adaptateur les méthodes obligatoires comme Listeners :

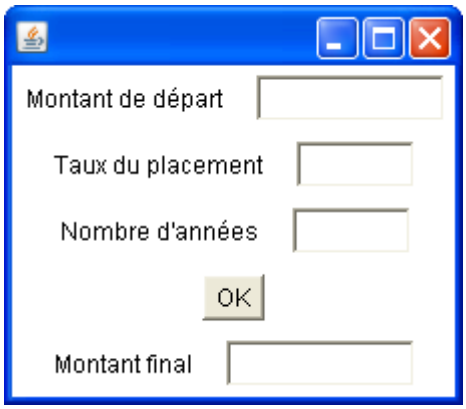
```
   public void windowOpened(WindowEvent e){}           // WindowListener
   public void windowClosing(WindowEvent e) {          // 7 méthodes
          delegue.quitter() ;
          }
   public void windowClosed(WindowEvent e){}
   public void windowIconified(WindowEvent e){}
   public void windowDeiconified(WindowEvent e){}
   public void windowActivated(WindowEvent e){}
   public void windowDeactivated(WindowEvent e){}

   public void actionPerformed(ActionEvent e) {        // ActionListener
          delegue.afficheres() ;                       // 1 méthode
          }
```

8) Ecrire les méthodes de la classe Delegue réalisant les actions souhaitées :

```
   public void quitter() {
          ...
          }

   public void afficheres() {
          ...
          }
```

COMPILER ET EXECUTER

Exercice 2



On a créé précédemment l'interface ci-contre dans le dossier **fenetre**. Cette interface comporte :

- un TextField appelé **dep** pour la saisie du montant de départ
- un TextField appelé **tx** pour la saisie du taux
- un TextField appelé **nb** pour la saisie du nombre d'années
- un Button **b** pour valider
- un TextField appelé **res** pour l'affichage du montant final.

Ouvrir le fichier **Fenetre.java** du dossier **fenetre**.

Compléter le fichier **Fenetre.java** par un objet d'une classe Delegue et un objet d'une classe Adaptateur (suivre la même démarche qu'à l'exercice précédent) de telle sorte que le programme traite les événements suivants :

- cliquer sur la **case de fermeture** de la fenêtre ferme la fenêtre
- cliquer sur le bouton « OK » permet de faire afficher le montant final du placement (regarder les conversions ci-dessous).

Recopier les dossiers **utilawt** et **fenetre** sur votre clef USB.

Rappels sur les conversions

Les contenus des champs de texte des **TextField** ou **TextArea** sont des **chaînes de caractères** de la classe String. Or pour effectuer des calculs il faut travailler sur des **nombres**.

La **conversion** d'une chaîne de caractères notée s :

- en un objet a de la classe Double se fait par : `Double a = new Double(s) ;`
- en un double b se fait par : `double b = new Double(s).doubleValue() ;(*)`
  OU BIEN par : `double b = Double.parseDouble(s) ;  (**)`

- en un objet c de la classe Float se fait par : `Float c = new Float(s) ;`
- en un float d se fait par : `float d = new Float(s).floatValue() ;  (*)`
  OU BIEN par : `float d = Float.parseFloat(s) ;  (**)`

- en un objet e de la classe Integer par : `Integer e =  new Integer(s) ;`
- en un int f par : `int f = new Integer(s).intValue() ;  (*)`
  OU BIEN par : `int f = Integer.ParseInt(s) ;  (**)`

(*) : **méthode d'instance** appliquée à un **objet de la classe enveloppante**
(**) : **méthode de classe** appliquée à **la classe enveloppante**.

Inversement, pour afficher un **résultat numérique** dans un champ de texte, il faut le convertir en une **chaîne de caractères**.

La conversion d'un double g en une chaîne s se fait par : `String s = new Double(g).toString());`
OU BIEN (moins élégant mais autorisé) par : `String s = "" + g ;`
La conversion d'un float h en une chaîne s se fait par : `String s = new Float(h).toString());`
OU BIEN (moins élégant mais autorisé) par : `String s = "" + h ;`
La conversion d'un int i en une chaîne s se fait par : `String s = new Integer(i).toString());`
OU BIEN (moins élégant mais autorisé) par : `String s = "" + i ;`

TD n°8 : Calculs à partir d'interfaces

Exercice 1

Le dossier **calculette** contient un fichier **Calculette.java**. Compiler et faire exécuter cette application. Le but est de faire exécuter les opérations prévues sur les deux nombres donnés (entiers ou réels).

```
import java.awt.*;
import java.awt.event.*;

class Calc extends Frame {
	protected Panel p1, p2, p3, p10,p11,p12,p13,p20,p21,p22,p23,p24,p31,p32;
	protected TextField f1, f2, f3;
	protected Button b1, b2, b3, b4, b5;
	protected Delegue d1;
	protected Adaptateur a1;

	Calc () {
		setLayout(new BorderLayout());
		p1=new Panel(); p1.setLayout(new GridLayout(1,4));
		p10=new Panel(); p10.add(new Label (" a ")); p1.add(p10);
		p11=new Panel(); f1 = new TextField(8); p11.add(f1); p1.add(p11);
		p12=new Panel(); p12.add(new Label (" b ")); p1.add(p12);
		p13=new Panel(); f2 = new TextField(8); p13.add(f2); p1.add(p13);
		add("North", p1);

		p2=new Panel(); p2.setLayout(new GridLayout(1,5));
		p20=new Panel(); b1 = new Button(" + "); p20.add(b1); p2.add(p20);
		p21=new Panel(); b2 = new Button(" - "); p21.add(b2); p2.add(p21);
		p22=new Panel(); b3 = new Button(" * "); p22.add(b3); p2.add(p22);
		p23=new Panel(); b4 = new Button(" / "); p23.add(b4); p2.add(p23);
		p24=new Panel(); b5 = new Button(" a^b "); p24.add(b5); p2.add(p24);
		add("Center", p2);

		p3=new Panel(); p3.setLayout(new GridLayout(1,2));
		p31 = new Panel(); p31.add(new Label (" result ")); p3.add(p31);
		p32 = new Panel(); f3 = new TextField (20); p32.add(f3); p3.add(p32);
		add("South", p3);

		d1 = new Delegue(this);
		a1 = new Adaptateur (d1);
		this.addWindowListener (a1);
		...
		}
	}

public class Calculette {
	public static void main (String args [] ) {
		Calc f = new Calc();
		f.pack();
		f.show();
		}
	}

class Delegue {
	protected Calc c1;

	Delegue (Calc c) {
		c1=c;
		}
	public void exit() {
		System.exit(0);
		}
	public void somme() {
		}
```

```
	public void diff() {
		}
	public void mult() {
		}
	public void divis() {
		}
	public void puis() {
		}
	}

class Adaptateur implements ActionListener, WindowListener {
	protected Delegue d1;

	Adaptateur (Delegue d) {
		d1=d;
		}

	public void actionPerformed (ActionEvent e) {
		// clic sur boutton
		Object src = e.getSource();
		String param = ((Button)src).getLabel();
		if (param.equals(" + ")) {d1.somme();}
		if (param.equals(" - ")) {d1.diff();}
		if (param.equals(" * ")) {d1.mult();}
		if (param.equals(" / ")) {d1.divis();}
		if (param.equals(" a^b ")) {d1.puis();}
		}
	public void windowOpened (WindowEvent e) {}
	public void windowClosing (WindowEvent e) {
		// fermer la fenetre
		d1.exit();
		}
	public void windowClosed (WindowEvent e) {}
	public void windowIconified (WindowEvent e) {}
	public void windowDeiconified (WindowEvent e) {}
	public void windowActivated (WindowEvent e) {}
	public void windowDeactivated (WindowEvent e) {}
	}
```

Attachez adapt à tous les boutons, puis complétez les 5 méthodes de calcul `somme()`, `diff()`, `mult()`, `divis()` et `puis()` pour que les calculs se fassent correctement..

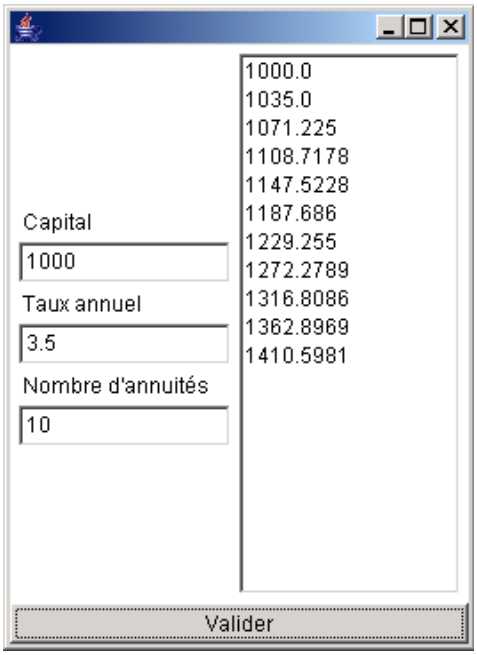
Exercice 2

Le dossier **utilplan** contient un fichier **UtilPlan.java**. Compiler et faire exécuter cette application.
Le programme est le suivant :

```
import java.awt.* ;
import java.awt.event.* ;

class Plan extends Frame {
	protected Panel p, p1 ;
	protected TextField capital ;
	protected TextField taux ;
	protected TextField nbannuites ;
	protected List resultat ;
	protected Button ok ;
	protected Delegue delegue ;
	protected Adaptateur adapt ;

	Plan() {
		setLayout(new BorderLayout()) ;
		p = new Panel() ;
		p1 = new Panel() ;
		p1.setLayout(new GridLayout(6,1)) ;
		p1.add(new Label("Capital"));
```

```
              capital=new TextField() ;
              p1.add(capital) ;
              p1.add(new Label("Taux annuel"));
              taux=new TextField() ;
              p1.add(taux) ;
              p1.add(new Label("Nombre d'annuités"));
              nbannuites=new TextField() ;
              p1.add(nbannuites) ;
              p.add(p1) ;
              resultat = new List(20) ;
              p.add(resultat) ;
              add("Center", p) ;
              ok = new Button("Valider") ;
              add("South", ok) ;
              delegue = new Delegue(this) ;
              adapt=new Adaptateur(delegue) ;
              ok.addActionListener(adapt) ;
              this.addWindowListener(adapt);
              }
       }

class Delegue {
       protected Plan p ;

       Delegue(Plan f) {
              p = f ;
              }
       public void quitter() {
              System.exit(0) ;
              }
       void calcule() {
              ...
              ...
              }
       }

class Adaptateur implements ActionListener, WindowListener {
       protected Delegue delegue ;

       public Adaptateur(Delegue d) {
              delegue = d ;
              }

       public void actionPerformed(ActionEvent e) {
              Object src = e.getSource();
              String param = ((Button)src).getLabel();
              if (param == "Valider") delegue.calcule() ;
              }

       public void windowClosing(WindowEvent e) {
              delegue.quitter() ;
              }
       public void windowClosed(WindowEvent e) { }
       public void windowIconified(WindowEvent e) { }
       public void windowDeiconified(WindowEvent e) {
       public void windowActivated(WindowEvent e) {
       public void windowDeactivated(WindowEvent e) {
       public void windowOpened(WindowEvent e) {
       }

public class UtilPlan {
              public static void main(String args[]) {
              Plan p = new Plan() ;
              p.pack() ;
              p.show() ;
              }
       }
```

Le but de l'exercice est de faire afficher les valeurs acquises par le capital, placé à un taux fixe pendant un certain nombre d'années. Ces trois données sont entrées par l'utilisateur dans la partie gauche de la fenêtre. Un clic sur le bouton « Valider » doit provoquer le calcul des valeurs acquises (voir illustration ci-dessous).



Complétez la méthode `void calcule()` de la classe Delegue pour qu'elle fasse afficher les résultats du calcul, considérés comme des réels flottants, donc de type primitif float.

A noter : le capital initial doit figurer en première ligne.

On veillera à ce qu'à chaque appui sur le bouton « Valider », les résultats précédents soient effacés.

Recopier les dossiers **calculette** et **utilplan** sur votre clef USB.

TD n°9 : Applications et applets

Exercice 1 : utilisation de conditions dans une application

Le dossier **impot** contient le fichier **Impot.java**. Compiler ce fichier. Les boutons « Calculer » et « Annuler » sont pour l'instant inactifs. Le but de l'exercice est de les rendre actifs : « Calculer » doit faire afficher le nombre de parts, la valeur de la part et le montant de l'impôt ; « Annuler » doit annuler toutes les données saisies et affichées.

Calcul du nombre de parts :

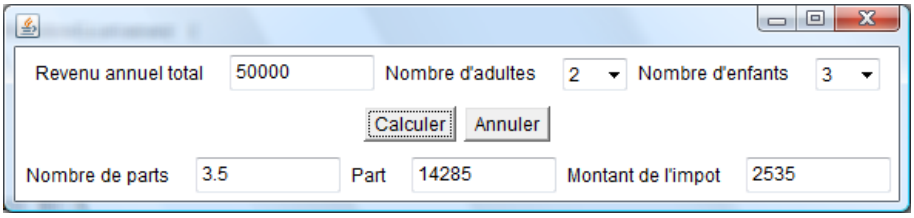
- si plus de 6 enfants : nombre de parts = nombre d'adultes + 0.5*nombre d'enfants + 1
- sinon, si plus de 3 enfants : nombre de parts = nombre d'adultes + 0.5*nombre d'enfants + 0.5
- sinon, si au moins un enfant : nombre de parts = nombre d'adultes + 0.5*nombre d'enfants
- sinon, nombre de parts = nombre d'adultes.

La part est égale au revenu annuel total divisé par le nombre de parts.

Calcul de l'impôt : six tranches sont définies correspondant aux intervalles suivants, associés aux taux spécifiés :

- tranche 5 : partie au-dessus de 25000 — taux d'imposition : 45 %
- tranche 4 : partie comprise entre 20000 et 25000 — taux d'imposition : 40 %
- tranche 3 : partie comprise entre 15000 et 20000 — taux d'imposition : 35 %
- tranche 2 : partie comprise entre 10000 et 15000 — taux d'imposition : 30 %
- tranche 1 : partie comprise entre 5000 et 10000 — taux d'imposition : 25 %
- tranche 0 : partie en-dessous de 5000 — taux d'imposition : 0 %

Ouvrir le fichier **Impot.java**. Compléter la méthode calcul de la classe Delegue pour qu'elle calcule et fasse afficher le résultat demandé. Exemple :




```
import java.awt.*;
import java.awt.event.*;

class FenetreImpot extends Frame {
       protected Panel p1, p2, p3 ;
       protected TextField rev, nbpart, part, impot ;
       protected Choice nba, nbe ;
       protected Button confirmer, annuler ;
       protected Delegue delegue;
       protected Adaptateur adapt;

       FenetreImpot() {
              this.setLayout(new GridLayout(3,1));
              p1 = new Panel();
              p1.add(new Label("Revenu annuel total"));
              rev= new TextField(10); p1.add(rev);
              p1.add(new Label("Nombre d'adultes"));
              nba = new Choice();
              for (int i = 1 ; i <=10 ; i++) nba.add(new Integer(i).toString());
              p1.add(nba);
              p1.add(new Label("Nombre d'enfants"));
              nbe = new Choice();
              for (int i = 0 ; i <=10 ; i++) nbe.add(new Integer(i).toString());
              p1.add(nbe);
              add(p1);
```

```
              p2 = new Panel();
              confirmer = new Button("Calculer"); p2.add(confirmer);
              annuler = new Button("Annuler"); p2.add(annuler);
              add(p2);
              p3 = new Panel();
              p3.add(new Label("Nombre de parts"));
              nbpart= new TextField(10); p3.add(nbpart);
              p3.add(new Label("Part"));
              part= new TextField(10); p3.add(part);
              p3.add(new Label("Montant de l'impot"));
              impot= new TextField(10); p3.add(impot);
              add(p3);
              delegue = new Delegue(this);
              adapt = new Adaptateur (delegue);
              this.addWindowListener (adapt);
              confirmer.addActionListener(adapt);
              annuler.addActionListener(adapt);
              }
       }

public class Impot {
       public static void main(String[] args) {
              FenetreImpot f = new FenetreImpot();
              f.show(); f.pack();
              }
       }

class Delegue {
       protected FenetreImpot fif;

       Delegue (FenetreImpot c) {
              fif=c;
              }

       public void exit() {
              System.exit(0);
              }

       public void calcule() {
              // récupération des données et conversion en numérique
              // double revenu = ...
              // int nba = ...
              // int nbe = ...
              double nbpart ;
              // calcul et affichage de nbpart
              ...
              double part = 0 ;
              // calcul et affichage de part
              ...
              boolean tranche5 = (part >= 25000) ;
              boolean tranche4 = ...
              boolean tranche3 = ...
              boolean tranche2 = ...
              boolean tranche1 = ...
              boolean tranche0 = (part < 5000);
              double taux5 = 0.45 ;
              double taux4 = 0.40 ;
              double taux3 = 0.35 ;
              double taux2 = 0.30 ;
              double taux1 = 0.25 ;
              double impot ;
              // calcul de l'impot
              if (tranche5) { ... }
              else if (tranche4) { ... }
              else if (tranche3) { ... }
              else if (tranche2) { ... }
              else if (tranche1) { ... }
              else { ... }
```

```
            // affichage de impot
            ...
            }

      public void annulle() {
            ...
            }
      }

class Adaptateur implements ActionListener, WindowListener {
      protected Delegue delegue;

      Adaptateur (Delegue d) {
            delegue=d;
            }

      public void actionPerformed (ActionEvent e) {
            // clic sur boutton
            Object src = e.getSource();
            String param = ((Button)src).getLabel();
            if (param.equals("Calculer")) { delegue.calcule(); }
            if (param.equals("Annuler")) { delegue.annulle(); }
            }
      public void windowOpened (WindowEvent e) {}
      public void windowClosing (WindowEvent e) { delegue.exit(); }
      public void windowClosed (WindowEvent e) {}
      public void windowIconified (WindowEvent e) {}
      public void windowDeiconified (WindowEvent e) {}
      public void windowActivated (WindowEvent e) {}
      public void windowDeactivated (WindowEvent e) {}
      }
```

Exercice 2 : transformation d'une application en applet

Le dossier **utilplan** contient le fichier **UtilPlan.java**. On va le transformer en applet.

Créer un dossier **plan** dans le dossier **java**. Y recopier le fichier **UtilPlan.java**.

Modifier le fichier de la manière suivante :

- importer le paquetage java.applet
- faire dériver la classe Plan de la classe Applet au lieu de la faire dériver de la classe Frame
- supprimer la classe UtilPlan
- la classe Plan étant devenue la classe maître, la déclarer public, et transformer son constructeur en la méthode : `public void init()`
- enregistrer le fichier **sous le nom de la nouvelle classe principale**, c'est-à-dire **Plan.java**.
- modifier la classe Adaptateur pour qu'elle n'implémente plus l'interface WindowListener
- supprimer dans la classe Plan l'enregistrement de l'adaptateur adapt comme WindowListener
- enlever la méthode quitter de la classe Delegue

Ecrire ensuite une page **plan.html** chargeant l'applet Plan. Vérifier en faisant exécuter l'applet.

Rappel : une applet ne s'exécute pas par un appel direct à l'interpréteur java, contrairement à une application, mais via une page HTML chargée par un navigateur disposant d'un interpréteur java

Il y a deux manières d'exécuter des applets. Faire successivement les deux :

- un appel à un visualiseur (l'appletViewer) simulant le navigateur, à qui on donne le nom de la page HTML contenant l'applet ou les applets que l'on veut visualiser. Taper, dans la fenêtre MS-DOS, la commande : **appletViewer plan.html.** La page HTML n'est pas visible, seule l'applet est visualisée.

- un appel à un navigateur à qui on donne le nom de la page HTML contenant l'applet. Le navigateur se lance, charge la page et l'applet.

Exercice 3 : compléter une applet (2)

Le dossier **convert** contient les fichiers **Convert.java** et **convert.html**. Compiler le fichier **Convert.java** puis faire charger la page **convert.html** par un navigateur. Le bouton = est pour l'instant inactif. Le but de l'exercice est de le rendre actif et de faire afficher dans le dernier champ de texte le résultat de la conversion en euros des monnaies proposées. Illustration :

2) Ouvrir le fichier **Convert.java** dont le code est le suivant :

```
import java.awt.* ;
import java.applet.* ;
import java.awt.event.* ;

public class Convert extends Applet {
       static int nb = 4 ;
       static String listemonnaies[] ;
       static double listevaleurs[] ;
       protected Panel p ;
       protected Button bouton ;
       protected TextField res ;
       protected TextField dep ;
       protected List monnaie ;
       protected Delegue d ;
       protected Adaptateur adapt ;

       public static void initmonnaies() {
              listemonnaies = new String[nb] ;
              listevaleurs = new double[nb] ;

              listemonnaies[0]="dollars US" ;
              listevaleurs[0]=1.30 ;    // 1 euro = 1.30 dollar US

              listemonnaies[1]="francs suisses" ;
              listevaleurs[1]=1.4737 ;

              listemonnaies[2]="livres sterling" ;
              listevaleurs[2]=0.6288 ;

              listemonnaies[3]="yen" ;
              listevaleurs[3]=110.49 ;
              }
```

```
	public void init() {
		initmonnaies() ;
		setLayout(new BorderLayout()) ;
		p = new Panel() ;
		dep = new TextField(10) ;
		p.add(dep) ;
		monnaie = new List(4) ;
		for (int i = 0 ; i < nb ; i++) monnaie.add(listemonnaies[i]) ;
		p.add(monnaie) ;
		bouton = new Button("=") ;
		p.add(bouton) ;
		res = new TextField(10) ;
		p.add(res) ;
		add("Center", p) ;
		d = new Delegue(this) ;
		adapt = new Adaptateur(d);
		bouton.addActionListener(adapt) ;
		}
	}

class Delegue {
	protected Convert ap ;

	Delegue(Convert a) {
		ap = a ;
		}
	void conversion() {
		double valEnEuros;
		// récupère le nombre tapé dans le premier champ
		// et le convertit en double
		// String s =
		// double quantite =

		// puis l'unité monétaire sélectionnée
		// int choix =

		// convertit le tout en euros
		// valEnEuros =

		// puis en chaine de caractères
		// s =

		// et demande à l'applet d'afficher le résultat en euros

		}
	}

class Adaptateur implements ActionListener {
	protected Delegue delegue ;

	Adaptateur(Delegue dd) {
		delegue = dd ;
		}

	public void actionPerformed(ActionEvent e) {
		delegue.conversion() ;
		}
	}
```

3) En exploitant les méthodes des classes List et TextField, ainsi que les méthodes de conversion de chaînes en nombres (et réciproquement), compléter la méthode conversion de la classe Delegue pour qu'elle fasse afficher dans le deuxième champ texte de l'applet le résultat demandé.

Recopier les dossiers **impot, plan** et **convert** sur votre clef USB.

TD n°10 : Applets et images

Exercice 1 : insertion d'image

Dans le dossier **convert** se trouve une image **euro.jpg** que l'on va insérer dans l'applet **Convert.java.** Pour cela :

- agrandir la fenêtre de l'applet afin de laisser de la place pour l'image : dans la page HTML, modifier la **hauteur** de l'applet de la manière suivante :
```
<APPLET CODE="Convert.class" WIDTH="400" HEIGHT="180">
</APPLET>
```

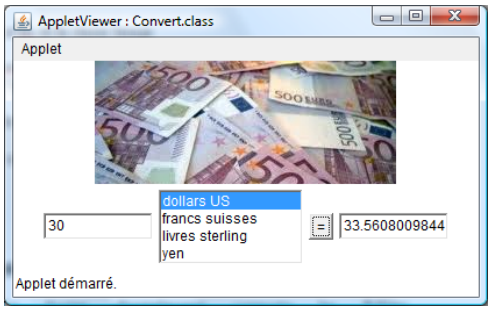
- déclarer dans l'applet Convert un attribut im de la classe Image

- dans la methode init() de l'applet Convert, laisser de la place pour l'image en mettant les éléments de l'interface **au sud** de la fenêtre plutôt qu'**au centre** : au lieu d'écrire : `add("Center",p);` écrire : `add("South",p);`

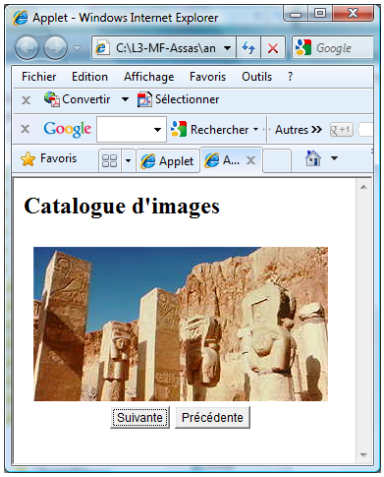
- toujours dans la methode init() de l'applet Convert, charger l'image **euro.jpg** en mémoire dans l'attribut im

- ajouter à l'applet Convert la méthode **`public void paint(Graphics g)`** faisant afficher l'image im au point de coordonnées 70, 0

Compiler et tester. Résultat attendu :



Exercice 2 : défilement d'images



Le dossier **chargeimages** comporte les fichiers **ChargeImages.java** et **ExempleChargeImages.html**, ainsi que 19 fichiers images **im1.jpg**, **im2.jpg**, …, **im19.jpg**.

Compiler le fichier **ChargeImages.java** puis faire charger la page **ExempleChargeImages.html** par le navigateur.

Les boutons « Suivante » et « Précédente » sont pour le moment inactifs. Le but de l'exercice est de les rendre actifs.

Ouvrir le fichier **ChargeImages.java** :

```
import java.awt.* ;
import java.applet.* ;
import java.awt.image.* ;
import java.awt.event.* ;

public class ChargeImages extends Applet {
	protected Panel p ;
	protected Image image ;
	protected Button suivant, precedent ;
	protected int rang ;
	...
	...

	public void init() {
		setLayout(new BorderLayout()) ;
		rang = 1 ;
		p = new Panel() ;
		suivant = new Button("Suivante") ; p.add(suivant) ;
		precedent = new Button("Précédente") ; p.add(precedent) ;
		add("South", p) ;
		...
		...
		}
	public void paint(Graphics g) {
		image = getImage(getDocumentBase(), "im"+rang+".JPG") ;
		g.drawImage(image, 10, 10, this) ;
		}
	}

class Delegue {
	protected ChargeImages ci ;

	Delegue(ChargeImages c) { ci = c ; }

	public void avancer() {
		...
		...
		ci.repaint() ;
		}
	public void reculer() {
		...
		...
		ci.repaint() ;
		}
	}

class Adaptateur implements ActionListener {
	protected Delegue delegue ;

	public Adaptateur(Delegue d) {
		delegue = d ;
		}
	public void actionPerformed(ActionEvent e) {
		Object src = e.getSource();
		String param = ((Button)src).getLabel();
		if (param.equals("Suivante")) delegue.avancer() ;
			else delegue.reculer() ;
		}
	}
```

Compléter les lignes manquantes pour associer l'adaptateur aux boutons « Suivante » et « Précédente » et faire en sorte que les clics sur ces boutons permettent d'afficher respectivement l'image suivante et l'image précédente. On surveillera les numéros des images afin de ne pas demander d'afficher une image inexistante. Quand on aura parcouru toutes les images, on bouclera sur les images déjà vues.

Exercice 3 : changement d'image

Le dossier **changeimages** comporte les fichiers **ChangeImages.java** et **ExempleChangeImages.html**, ainsi que les 19 fichiers images **im1.jpg**, **im2.jpg**, …, **im19.jpg**.

Compiler le fichier **ChangeImages.java** puis faire charger la page **ExempleChangeImages.html** par le navigateur.

Le but est d'associer un listener à la liste de choix - de telle sorte que dès qu'un nom d'image est sélectionné, l'image correspondante s'affiche instantanément sans avoir besoin d'un quelconque bouton de confirmation..

Cet exercice nécessite un Listener sur un objet de type List : il s'agit d'un **ItemListener** dont la seule méthode obligatoire est **itemStateChanged(ItemEvent e)**.

Compléter le fichier **ChangeImages.java** ci-dessous afin d'implémenter ce Listener et cette fonctionnalité.

```
import java.awt.* ;
import java.applet.* ;
import java.awt.image.* ;
import java.awt.event.* ;

public class ChangeImages extends Applet {
	protected Panel p ;
	protected Image image ;
	protected List l;
	protected String nomimage ;

	public void init() {
		setLayout(new BorderLayout()) ;
		p = new Panel() ;
		l = new List(10);
		for (int i = 1 ; i < 20; i++) l.add("im"+i+".jpg");
		p.add(l) ;
		add("North", p) ;
		nomimage = "im1.jpg"; // image initiale
		}

	public void paint(Graphics g) {
		image = getImage(getDocumentBase(), nomimage) ;
		g.drawImage(image, 10, 200, this) ;
		}
	}

class Delegue {
	protected ChangeImages ci ;

	Delegue(ChangeImages c) { ci = c ; }
	}

class Adaptateur {
	protected Delegue delegue ;

	public Adaptateur(Delegue d) {
		delegue = d ;
		}
	}
```

Recopier les dossiers **chargeimages** et **changeimages** sur votre clef USB.

Annexes

Noms des variables

Le nom d'une variable est composé **d'un seul mot** et commence obligatoirement par une **lettre**. Il peut être constitué de lettres, de chiffres ou du caractère souligné _ et peut contenir un nombre quelconque de caractères (se limiter pour la commodité à une taille raisonnable).

Casse

Il y a une distinction entre **minuscules** et **majuscules** (java est toujours sensible à la CASSE). Par convention les noms de variable sont orthographiés en lettres minuscules sauf la première lettre de chaque mot si le nom de la variable en comporte plusieurs : nomDeFamille par exemple.

Mots clés

Les **mots clés** du langage ne peuvent en aucun cas être utilisés comme noms de variable.
Ce sont les mots :

```
abstract        assert          boolean         break           byte
case            catch           char            class           const
continue        default         do              double          else
extends         false           final           finally         float
for             goto            if              implements      import
instanceof      int             interface       long            native
new             null            package         private         protected
public          return          short           static          strictfp
super           switch          synchronized    this            throw
throws          transient       true            try             void
volatile        while
```

Portée des variables

La **portée d'une variable** est la **portion de code** dans laquelle on peut manipuler cette variable. Elle est fonction de l'emplacement où est située la déclaration de la variable.

La déclaration d'une variable peut être faite dans le bloc de code d'une classe, le bloc de code d'une fonction ou un bloc de code à l'intérieur d'une fonction (boucle for, while, do while par exemple). **Seul le code du bloc où est déclarée la variable peut utiliser la variable**.

Il ne peut pas y avoir deux variables portant le même nom avec la même portée. On peut cependant déclarer une variable interne à une fonction ou un paramètre d'une fonction ayant le même nom qu'une variable déclarée au niveau de la classe. La variable déclarée au niveau de la classe est dans ce cas **masquée** par la variable interne à la fonction.