Laboratoire 1

DURÉE PRÉVUE : 4H00

OBJECTIFS

Au terme de ce laboratoire, l'étudiant sera capable de :

→ Identifier les éléments d'un programme.

→ Utiliser les commandes de la console/du terminal.

→ Écrire, compiler, exécuter et déboguer un programme Java.

POUR EN SAVOIR PLUS

DOCUMENTATION

Dans ces exercices, il sera essentiellement question d'apprendre à exécuter des commandes dans la console/le terminal de votre système d'exploitation.

Sachez que la console propose de nombreuses **commandes**. Seules quelques-unes d'entre elles vous seront montrées. N'hésitez donc pas à en tester d'autres en consultant la liste suivante (pour les utilisateurs de Windows) :

http://windows.developpez.com/cours/ligne-commande/?page=page_4

En plus de ces commandes préexistantes, d'autres **commandes spécifiques au Java** sont disponibles après l'installation du JDK. Ces dernières sont détaillées sur le site d'*Oracle* :

https://docs.oracle.com/en/java/javase/21/docs/specs/man/index.html

EXERCICE 1 : IDENTIFIER LES ÉLÉMENTS D'UN PROGRAMME

Ci-dessous, un programme qui permet de déterminer le jour de la semaine correspondant à une date donnée. La capture provient de l'éditeur de code *Notepad++*.

```
1  package labo1;
2
3  public class JourDeLaSemaine {
4
5      public static void main(String[] args) {
6          // Variables pour les données de départ
7          int jour = 27, mois = 9, annee = 2024;
8
9          // Variables pour le traitement
10         int anneeEnCours, a, b, m;
11
12         // Variable pour le résultat
13         int jourDeLaSemaine;
14
15         // Traitement
16         anneeEnCours = (14 - mois) / 12;
17         a = annee - anneeEnCours;
18         b = a + a / 4 - a / 100 + a / 400;
19         m = mois + 12 * anneeEnCours - 2;
20         jourDeLaSemaine = (jour + b + 31 * m / 12) % 7;
21
22         // Affichage du résultat
23         System.out.println(jourDeLaSemaine);
24     }
25
26 }
```

Nommez les éléments suivants et donnez-en une brève description :

→ [LIGNE 1] `labo1` packet regroupant tous les fichiers java

→ [LIGNE 3] `JourDeLaSemaine` Nom de la classe & du fichier la contenant

→ [LIGNE 5] `main` fonction static principal de la class JourDeLaSemaine

→ [LIGNE 6] `// Variables pour...` Commentaires du professeur

→ [LIGNE 7] `27` Entier

→ [LIGNE 7] `jour = 27` Attribution de la valeur 27 a la variable jour

→ [LIGNE 13] `int jourDeLaSemaine` Déclaration de la variable "jourDeLaSemaine"

→ [LIGNE 16] `(14 - mois) / 12` Opération mathématique

→ [LIGNE 23] `System.out.println(jourDeLaSemaine)` Sortie console de la valeur de la variable "jourDeLaSemaine"

EXERCICE 2 : CODER UN PROGRAMME, LE COMPILER ET L'EXÉCUTER

AVERTISSEMENT

L'**espace de travail** choisi pour les exemples est le dossier **`C:\Java`**. Si vous placez votre espace de travail à un autre endroit, veillez à adapter les chemins d'accès qui vous seront spécifiés dans cet énoncé.

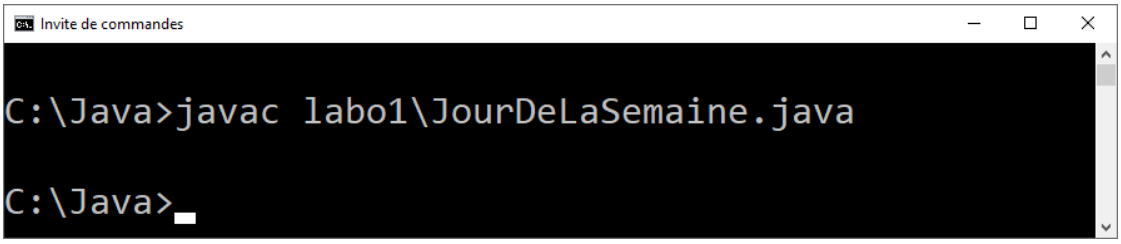
RÉDIGER UN PROGRAMME

1. Créez dans votre espace de travail un dossier nommé **`labo1`** (en minuscules, sans espace et se termine par le chiffre 1), ce qui donne le chemin d'accès **`C:\Java\labo1`**. C'est dans ce dossier que vous enregistrerez tous les programmes de ce 1er labo.
2. Lancez un éditeur de code (par exemple, *Notepad++*).
3. Reproduisez le code source du programme **`JourDeLaSemaine`** donné dans l'exercice précédent.
4. Enregistrez le fichier dans le dossier **`labo1`** en le nommant **`JourDeLaSemaine.java`**.

COMPILER UN PROGRAMME

1. Accédez à la console/au terminal (dans Windows : ouvrir la fenêtre *Exécuter* avec le raccourci **`Windows + R`**, puis saisir **`cmd`**).
2. Changez de dossier à l'aide de la commande **`cd`** (*change directory*) pour vous rendre dans votre dossier de travail **`Java`**.

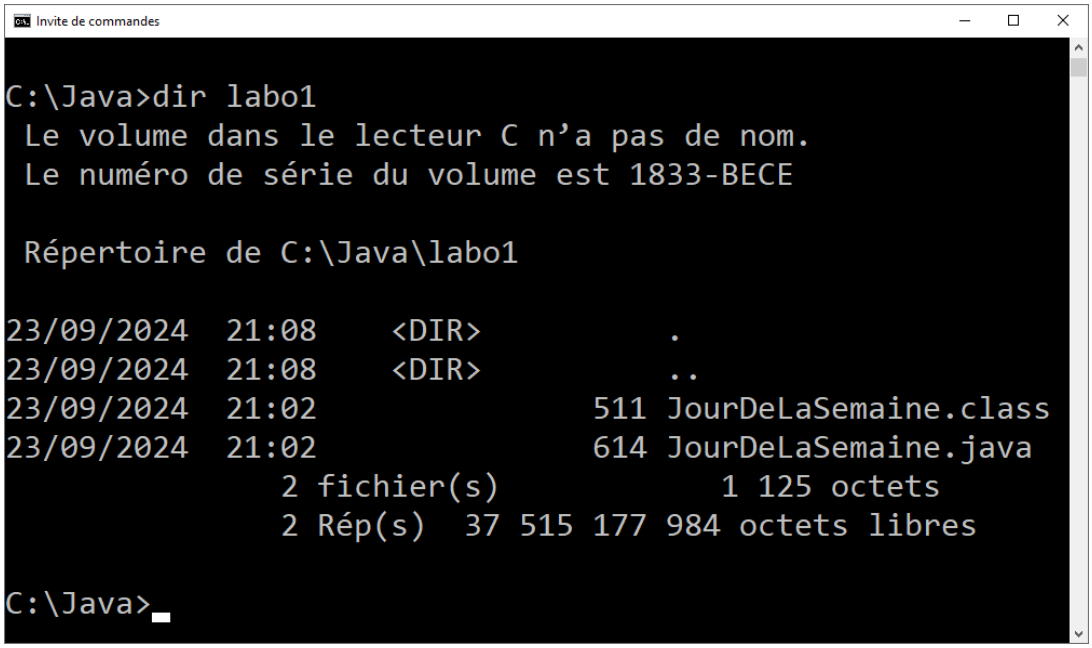
3. Compilez le programme **`JourDeLaSemaine`** avec la commande **`javac`**.



Si la compilation réussit, un fichier nommé **`JourDeLaSemaine.class`** est généré dans le dossier **`labo1`**.

REMARQUE

Pour afficher le contenu du dossier courant (dans ce cas, **`C:\Java\labo1`**), vous pouvez utiliser la commande **`dir`**. Celle-ci permet de vérifier rapidement que le fichier **`.class`** a bien été créé au terme de la compilation :



Si la compilation échoue, lisez attentivement les messages détaillant les erreurs détectées afin de trouver l'origine du problème.

CORRIGER EFFICACEMENT LES ERREURS DE COMPILATION

Voici quelques conseils :

→ Dans la description d'une erreur, **repérez le numéro de la ligne** spécifié à la suite du nom du fichier. Par exemple, l'erreur répertoriée ci-dessous est située à la ligne 10 du fichier **`JourDeLaSemaine.java`**.

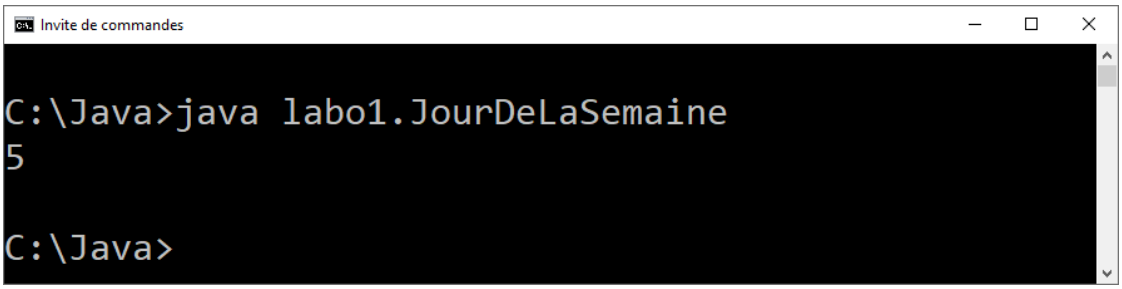
```
labo1\JourDeLaSemaine.java:10: error: ...
```

Ainsi, vous localiserez rapidement l'instruction problématique.

→ **Corrigez les erreurs par ordre d'apparition**, autrement dit, en les parcourant de haut en bas. En effet, les erreurs situées « plus loin » dans le code sont parfois causées par celles qui les précèdent.

EXÉCUTER UN PROGRAMME

Si la compilation a réussi, exécutez le programme **JourDeLaSemaine** avec la commande **java**. Le nombre « 5 » doit s'afficher en console.



COMPRENDRE LES ERREURS DE COMPILATION

1. Modifiez le code source du programme **JourDeLaSemaine** afin de voir comment réagit le compilateur dans les 4 cas de figure spécifiés après les encadrés suivants.

> **MÉTHODOLOGIE À SUIVRE**
>
> Après chaque modification du code source, n'oubliez pas d'enregistrer le fichier en cliquant sur l'icône ou en utilisant le raccourci **Ctrl + S**.
>
> Compilez le programme avec la commande **javac**.
>
> Après chaque compilation, n'oubliez pas d'annuler la modification en cliquant sur l'icône ou en utilisant le raccourci **Ctrl + Z**.

> **MIEUX MAÎTRISER LA CONSOLE**
>
> Voici quelques techniques facilitant l'usage de la console :
>
> → Utilisez la flèche du **haut/bas** pour retrouver les commandes précédentes/suivantes.
>
> → Appuyez sur **F7** pour afficher une fenêtre listant les dernières commandes exécutées.
>
> → Utilisez l'auto-complétion avec la touche **tabulation** pour accélérer la saisie d'un chemin (des pressions répétées de la touche **tabulation** permettent de parcourir les résultats correspondant aux premiers caractères saisis).

Traduisez les messages d'erreurs affichés par le compilateur dans chacun des cas ci-dessous. Comprenez-vous le sens du message au vu de la modification effectuée ?

→ [LIGNE 7] Renommez la variable **annee** en **année**.

Si toutes renommées --> aucun soucis sinon, une des variables est inconnu

→ [LIGNE 3] Modifiez le nom de la classe en remplaçant la lettre **J** majuscule par son équivalent minuscule.

erreur car nom class == nom fichier

→ [LIGNE 16] Placez en commentaire l'instruction `//anneeEnCours = (14 -...`

Annee En cours ce sera non initalisée donc erreur

→ [LIGNE 5] Supprimez l'accolade ouvrante **{**.

Problème synthaxique

[QUESTION FACULTATIVE] Voici une question beaucoup plus difficile. Quelle est l'explication de l'erreur détectée en effectuant la modification ci-dessous (*indice : la base 8 n'y est pas étrangère*) ?

→ [LIGNE 7] ajoutez un zéro en préfixe du littéral entier 9.

09 ne correspond pas a la structure binaire d'un entier

2. Modifiez le code source du programme **`JourDeLaSemaine`** de manière à reproduire l'erreur affichée par le compilateur :

→ Erreur n°1

retirer le ";" de la ligne 7

```
labo1\JourDeLaSemaine.java:10: error: ';' expected
                int anneeEnCours, a, b, m
                                         ^
1 error
error: compilation failed
```

→ Erreur n°2

rajouter "int jour;" dans le code peut importe la position a considération qu'il soit après la ligne 5

```
labo1\JourDeLaSemaine.java:13: error: variable jour is alr
eady defined in method main(String[])
                int jour;
                    ^
1 error
error: compilation failed
```

→ Erreur n°3

Remplacer System par system

```
labo1\JourDeLaSemaine.java:23: error: package system does
not exist
                system.out.println(jourDeLaSemaine);
                      ^
1 error
error: compilation failed
```

→ Erreur n°4

supprimer une accolade de fin

```
labo1\JourDeLaSemaine.java:24: error: reached end of file
while parsing
        }
         ^
1 error
error: compilation failed
```

COMPRENDRE LES ERREURS D'EXÉCUTION

Indiquez ce qu'il se passe lors de l'exécution du programme dans les 3 cas de figure spécifiés après l'encadré suivant.

MÉTHODOLOGIE À SUIVRE

Après chaque modification du code source, n'oubliez pas d'enregistrer le fichier !

Après chaque exécution, n'oubliez pas d'annuler la modification.

→ [LIGNE 5] Supprimez le contenu des parenthèses `(String[] args)`.

→ [LIGNE 7] Remplacez le littéral `2024` par `2000000000` (2 milliards).

→ [LIGNE 16] Remplacez le littéral `12` par `0`.

COMPRENDRE LE RÉSULTAT PRODUIT PAR LE PROGRAMME

Modifiez les données de départ (plus précisément, celles affectées aux variables **`jour`**, **`mois`** et **`annee`**) afin de répertorier les résultats affichés par le programme. En procédant de la sorte, déduisez la signification des nombres entiers ainsi obtenus.

MÉTHODOLOGIE À SUIVRE

Après chaque modification du code source, n'oubliez pas d'enregistrer le fichier !

DATE	NOMBRE ENTIER AFFICHÉ
26/09/2024	4
27/09/2024	5
28/09/2024	6
29/09/2024	0
30/09/2024	1
01/10/2024	2
02/10/2024	3
03/10/2024	4

EXERCICE 3 : DÉBOGUER UN PROGRAMME

LES ERREURS DE PROGRAMMATION

Nous l'avons constaté précédemment, il existe différents types d'erreurs :

- Les **erreurs de compilation** qui sont détectées au moment de la compilation d'un programme car elles empêchent le compilateur d'effectuer la traduction du code source en *bytecode* (par exemple, l'oubli d'un point-virgule).
- Les **erreurs d'exécution** qui sont détectées au moment de l'exécution d'un programme car ce dernier effectue une opération non valide (par exemple, une division par zéro) ou ne parvient pas à effectuer l'une des opérations spécifiées (par exemple, la fonction `main` est introuvable).
- Les **erreurs logiques** qui ne peuvent être détectées que par le programmeur lui-même lorsqu'il exécute son programme en comparant le résultat produit par ce dernier avec le résultat attendu (par exemple, un dépassement d'entier ou une retranscription erronée de calculs mathématiques). Ces *bugs* sont les plus complexes à identifier et à corriger !

Dès lors, **déboguer un programme** consiste à localiser une erreur logique dans son code source, puis à la corriger.

A force de programmer, l'une des premières compétences que vous développerez sera d'identifier rapidement vos erreurs. L'une des suivantes sera de devenir suffisamment rigoureux lorsque vous coderez pour éviter de commettre ces erreurs.

UTILISER UN DÉBOGUEUR

La méthode de débogage recommandée est l'**exécution pas-à-pas**. Elle nécessite l'emploi d'un outil appelé **débogueur** qui permet de guider l'exécution d'un programme. De cette manière, la partie du code source suspectée d'être à l'origine du problème est exécutée instruction par instruction. Durant ce processus, toujours grâce au débogueur, le programmeur a le loisir de consulter l'état des variables (et bien davantage). Il devient ainsi plus simple de déceler l'erreur dès l'apparition d'un résultat erroné.

Lorsque nous travaillerons avec le logiciel *Eclipse*, nous apprendrons les manipulations de base d'un débogueur. En attendant…

UTILISER LA FONCTION « PRINTLN »

Une autre méthode consiste simplement à afficher les informations souhaitées en cours d'exécution du programme.

Par exemple, dans le cas du programme **`JourDeLaSemaine`** avec lequel nous avons travaillé précédemment, l'instruction « **`System.out.println(…);`** » peut être utilisée comme ci-dessous afin d'afficher le résultat intermédiaire enregistré dans la variable **`anneeEnCours`** :

```
15          // Traitement
16          anneeEnCours = (14 - mois) / 12;
17          System.out.println(anneeEnCours);
18          a = annee - anneeEnCours;
```

Cette approche a l'avantage d'être simple et immédiate (elle ne nécessite pas la maîtrise d'un outil de débogage) pour un débutant.

Cependant, il n'est pas évident de cibler directement les bonnes informations à afficher. Le programmeur doit fréquemment ajouter de nombreuses instructions de débogage, ce qui rend le **processus fastidieux**. De plus, une fois l'erreur corrigée, il se doit de supprimer toutes ces instructions devenues inutiles.

À VOUS DE JOUER

Téléchargez sur la page du cours le fichier nommé **`Inconnu.java`** et placez-le dans votre dossier **`C:\Java\labo1`**.

Voici le code source de ce programme :

```
1  package labo1;
2
3  public class Inconnu {
4
5      public static void main(String[] args) {
6          int nombre = 7805;
7          int somme = 0;
8          int chiffre;
9
10         chiffre = nombre % 10;
11         somme = somme + chiffre;
12         nombre = nombre / 10;
13
14         chiffre = nombre % 10;
15         somme = somme + chiffre;
16         nombre = nombre / 10;
17
18         chiffre = nombre % 10;
19         somme = somme + chiffre;
20         nombre = nombre / 10;
21
22         chiffre = nombre % 10;
23         somme = somme + chiffre;
24         nombre = nombre / 10;
25
26         System.out.println(somme);
27     }
28
29 }
```

Ajoutez des appels de la fonction **`println`** aux lignes 13, 17, 21 et 25 pour afficher l'état des variables **`nombre`**, **`chiffre`** et **`somme`** après chaque groupe de 3 instructions.

Idéalement, faites en sorte que les informations affichées respectent la forme suivante :

```
nombre=780, chiffre=5, somme=5
```

Sur base des résultats obtenus, répondez aux questions suivantes :

→ Que représentent les valeurs de la variable **`chiffre`** par rapport à la valeur initiale de la variable **`nombre`** ?

"chiffre" est le dernier chiffre de "nombre"

→ Que permet de calculer la variable **`somme`** ?

"somme" est le total de chaque "chiffre" de nombre

REMARQUE

La technique employée ici est très fréquemment utilisée lorsqu'il s'agit de vérifier la validité d'un numéro (registre national, code-barres, carte bancaire …).

EXERCICE 4 : DÉTECTER ET CORRIGER LES ERREURS D'UN PROGRAMME

Téléchargez sur la page du cours le fichier nommé **ConversionsTemperature.java** et placez-le dans votre dossier **C:\Java\labo1**.

Voici le code source de ce programme :

```
1  package labo1;
2
3  public class ConversionsTemperature {
4
5      public static void main(String[] args) {
6          double tCelsius = 19;
7          double tKelvin;
8          double tFarhenheit;
9
10         tKelvin = tCelsius + 273,15;
11         tFarhenheit = 9 / 5 * tCelsius + 32;
12
13         System.out.print(kelvin);
14         System.out.println("K");
15         System.out.print(farhenheit);
16         System.out.println("°F");
17     }
18 }
```

Le programme **ConversionsTemperature** doit afficher la conversion de 19 degrés Celsius en Kelvin et en degrés Fahrenheit.

Les formules de conversion sont les suivantes :

→ K = °C + 273,15

→ °F = 9 / 5 × °C + 32

Avant de compiler et exécuter ce programme, effectuez mathématiquement les calculs ci-dessous afin de déterminer les résultats que le programme est censé produire pour une température de 19°C. Pour ce faire, aidez-vous d'une calculatrice (par exemple, celle de votre système d'exploitation) :

→ 19 + 273,15 = ... K 292, 15 K

→ 9 / 5 × 19 + 32 = ... °F 35,9 °F

Compilez le programme. Plusieurs erreurs de compilation sont alors détectées, corrigez-les sur base des informations affichées par le compilateur. N'oubliez pas de sauvegarder votre fichier !

Exécutez le programme. A la lecture du résultat affiché pour les degrés Fahrenheit, vous devriez constater la présence d'une erreur logique. En effet, la formule mathématique n'a pas été retranscrite correctement dans le code.

MÉTHODOLOGIE À SUIVRE

Ajoutez plusieurs appels de la fonction **println** à la ligne 12 pour afficher les résultats obtenus aux différentes étapes du calcul. Par exemple, sous la forme suivante :

Lorsque vous avez identifié l'opération problématique, corrigez-la (*indice : souvenez-vous du résultat obtenu en Java lors de la division de deux nombres entiers*).

EXERCICE 5 : UN AVANT-GOÛT DE PROGRAMMATION

Adaptez le programme **ConversionsTemperature** de sorte qu'il affiche les résultats manquants du tableau ci-dessous. Dans certains cas, il est nécessaire de retravailler les formules données.

Degrés Celsius	Kelvin	Degrés Fahrenheit
19,00	*292,15*	*66,20*
20,00	293.15	68
21,00	294..15	69.8
-173.16	*100,00*	-141.15
-17.7°C	255.37	*0,00*