Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr


# TP n°1

**Objectifs :** Écrire, compiler et exécuter ses premiers programmes.

## Rappels

Afin d'avoir un espace de travail bien organisé, commencez par lancer cette commande :

```
...> ijava Program     -> cela crée automatiquement les répertoires de travail
```

Pour chaque TP, vous devez créer vos programmes dans le répertoire correspond du répertoire `r1.01`. Comme c'est le premier TP, tous les programmes suivant doivent être créés dans le répertoire `r1.01/tp1`.

```
...> cd r1.01          -> nous nous plaçons dans le répertoire 'r1.01'
...> cd tp1            -> nous nous plaçons dans le répertoire 'tp1'
```

Nous voilà maintenant prêts à créer un premier programme simple, le compiler et l'exécuter.

**ATENTION : IL EST OBLIGATOIRE DE RESPECTER STRICTEMENT LES NOMS DE PROGRAMME DONNÉS DANS L'ÉNONCÉ !**

### Structure d'un programme iJava

Pour l'instant, les programmes que vous écrirez en ce début de ce semestre auront la structure suivante (**en gras**, ce qui ne change pas d'un programme à l'autre) :

```
class MonProgramme extends Program{

    // fonction principale appelée lors de l'exécution du programme
    void algorithm (){
      // contient une suite d'instructions
      String msg; //exemple de déclaration
      msg = "Bonjour l'univers"; // exemple d'affectation
      println(msg); // exemple d'appel de fonction
  }
}
```

Le squelette de programme ci-dessus est commenté. La syntaxe des commentaires est `//` pour une ligne unique et `/* ... */` pour un bloc (une suite de lignes). Ce programme minimal peut bien être compilé, il ne produit cependant aucune action à l'exécution, sa fonction principale ne comportant aucune instruction. Dans l'exercice de ce TP, il s'agira pour chaque exercice d'écrire un nouveau programme (au nom différent), avec une suite d'instructions correcte et produisant le résulat demandé dans la consigne.

### Cycle de développement

Voici les étapes que vous suivrez pour écrire un programme; dans l'exemple le programme s'appelle `MonProgramme`. Ce nom est à remplacer par le nom du programme que vous écrivez.

1. Éditer le code source à l'aide d'un éditeur de code source, nous recommandons **vscode**. Le code source doit impérativement se trouver dans un fichier nommé `MonProgramme.java`.
2. Compiler le code source avec la commande

   ```
   ijavac MonProgramme.java
   ```

   Pour cela vous devez vous placer dans le terminal dans le répertoire qui contient le fichier de code source.

3. Exécuter le programme avec la commande

```
ijava MonProgramme
```

Ces étapes sont répétées jusqu'à ce que le programme soit correct (c'est-à-dire qu'il produise le résultat attendu).

**Attention, si vous utilisez un ordinateur hors des salles de TP**, pour que les commandes `ijavac` et `ijava` fonctionnent, il faudra recréer l'alias en précisant le chemin de la librairie `program.jar`. Par exemple, si le fichier `program.jar` se situe dans le répertoire `/dev/`, on modifiera le fichier `.bashrc` en ajoutant :

```
alias ijavac='javac -cp ~/dev/program.jar'
alias ijava='java -cp ~/dev/program.jar:.'
```

### Variables et types

Un type définit la nature d'une information et son intervalle de valeurs. Pendant ce premier semestre, dans un premier temps, vous allez utiliser :

— le type `boolean`
— des types numériques: `int`, `double`, `char`
— le type des chaînes de caractères `String`

Une variable permet de stocker une information. En java, chaque variable a un type et doit être déclarée avant toute utilisation. Pour affecter une valeur à une variable, on utilise l'opérateur =. Une constante stocke une valeur qui ne peut pas être modifiée.

Voici quelques exemples de déclarations de variables et de constantes, et des précisions sur les noms de variables qu'on utilise en java.

```
int age; // Un nom de variable commence par une lettre minuscule.
age=19; //On donne à la variable âge la valeur 19
final double PI=3.1415; //On déclare une constante en utilisant le mot réservé final.
//Si un nom de variable est composé, on marque le changement de mot par une majuscule.
String leMotLePlusLong;
```

## Éléments d'organisation

Chaque exercice requiert la création d'un nouveau programme, donc d'un nouveau fichier. Veillez à bien utiliser strictement le même nom de fichier et de programme et surtout que cela corresponde à celui indiqué dans l'énoncé.

## Exercices

### Exercice 1 : Types manquants

Dans le programme ci-dessous, vous devez compléter par le bon type partout où les ... apparaîssent.

```
/**
* Types manquants
* @author yann.secq@univ-lille.fr
*/
class JeuxDeType extends Program {

  void algorithm(){
    ... prenom = "Alan";
    ... nom = "Turing";
    ... naissance = 1912;
    ... annee = 2022;
    ... age = annee - naissance;
    ... initiale = charAt(prenom,0);
    println(initiale + ". " + nom + " aurait eu " + age + "ans en " + annee);
  }

}
```

Une fois le programme complété, compilez-le et exécutez-le. L'exécution attendue consiste en l'affichage suivant :

```
A. Turing aurait eu 110 ans en 2022
```

Après cette première étape, modifiez les valeurs des variables de manière à ce qu'il réalise le même comportement mais cette fois concernant Ada Lovelace née en 1815.

### Amusons-nous avec les chaînes

Voici les fonctions disponibles sur les chaînes de caractère (type `String`) qui vous seront utiles dans cette partie.

— `int length (String uneChaine)`
Retourne le nombre de caractères du paramètre `uneChaine`

— `char charAt(String uneChaine, int indice)`
Retourne le caractère situé à l'indice indice dans la chaîne `uneChaine`.
**ATTENTION**: le premier caractère a pour indice `0` !

— `String substring(String uneChaine, int indiceDebut, int indiceFin)`
Retourne une copie de la chaîne `uneChaine` en commençant la copie à l'indice `indiceDebut` et en s'arrêtant à l'indice `indiceFin-1` (!).
**ATTENTION**: le premier caractère a pour indice **0** !

### Exercice 2 : D'une chaîne à une autre

Étudiez le programme donné ci-dessous :

```
class JeuxDeMots extends Program {

   void algorithm(){
     ... mot = "etat";
     ... resultat;
     ... premiereLettre = charAt(mot, 0);
     ... resteDuMot = substring(mot, 1, length(mot));
     resultat = resteDuMot + premiereLettre;
     println(resultat);
   }
}
```

1. Copier ce programme et compléter-le par les types adéquats partout où les ... apparaîssent.
2. Quel est le calcul effectué par ce programme ?
3. Faites fonctionner ce programme `iJava`. L'exécution attendue consiste en l'affichage suivant :

   ```
   tate
   ```

4. Testez votre programme avec un autre mot de départ.

### Exercice 3 : Affichage verlan

Écrivez un programme `Verlan`, qui consiste à afficher une chaîne en variable mais à l'envers : c'est-à-dire la fin (partie après le milieu) avant le début. Voici une base pour vous aider :

```
class Verlan extends Program {
  void algorithm(){
    ... mot = "cheval";
    ... tailleMot = ;
    ... indiceMilieu = ;
    ... debut = ;
    ... fin = ;
    println(fin+debut);
  }
}
```

1. Compléter ce programme pour qu'il fonctionne comme souhaité. Par exemple, avec la chaîne `"cheval"` stockée dans la variable `mot`, le programme affichera *valche*.
2. Une fois que votre programme marche avec cet exemple, vérifiez qu'il fonctionne bien dans tous les cas de figure. Par exemple, en remplaçant **uniquement** la chaîne `"cheval"` par `"malin"` dans votre programme, après compilation et exécution, celui-ci devrait afficher *linma*. Si tel n'est pas le cas, modifiez votre programme pour qu'il soit suffisamment général pour couvrir à la fois ces 2 cas.

## Conversions de types

Dans certaines situations, il est nécessaire de pouvoir changer le type d'une information. La plupart du temps ce premier semestre, cela concernera la conversion de réels vers des entiers ou des entiers vers les caractères (ou vice-versa).

L'opérateur permettant de forcer un changement de type, ou opérateur de *cast* en anglais, utilise une syntaxe singulière: un type entouré de parenthèses.

Exemple: `int note = (int) 15.6;`

Dans l'exemple ci-dessus, un nombre réel ( 15.6 ) est converti vers un entier grace à l'opérateur de cast `(int)`.

**ATTENTION** la conversion dans ce cas correspond à la partie entière du nombre réel (troncature) et non pas un arrondi mathématique. Ainsi, la variable `note` vaudra `15` une fois l'expression évaluée.

**ATTENTION** l'opérateur de forçage de type est plus prioritaire que les opérations arithmétiques, soyez donc attentifs au parenthésage de vos expressions !

La conversion de types est possible uniquement entre types compatibles. Tous les types numériques sont compatibles entre eux. En iJava, `char` est un type numérique et donc il est possible de convertir une valeur de type `char` en valeur de type `int` et vice versa. Par contre, le type `boolean` et le type `String` ne sont pas compatibles avec les types numériques. Ainsi, même si ça peut paraître étrange pour l'instant, l'opérateur de forçage de type ne permet pas de convertir une valeur de type `char` en valeur de type `String`.

## Exercice 4 : D'un type à l'autre

Cet exercice vous aidera à comprendre le fonctionnement de l'opérateur de conversion. Dans un premier temps, ne copiez pas et ne cherchez pas à compiler ce programme.

```
class Conversions extends Program {

  void algorithm(){

    print( "(int) 4.6 ->" );
    println( (int) 4.6 );

    print( "(double) 4 ->" );
    println( (double) 4 );

    print("2.1 + 3 -> ");
    println(2.1 + 3);

    print("(int) 'A' -> ");
    println( (int) 'A' );

    print(" (char) 66 -> ");
    println( (char) 66);


    print(" (int) 3.7 * 2 -> ");
    println( (int) 3.7 * 2);

    print("(int) (3.7 * 2) -> ");
    println((int) (3.7 * 2));

    print(" \"ABC\" + (char) 65 ");
    println( "ABC" + (char) 65 );

    }
}
```

1. Essayez de prédire le résultat du programme Conversions ci-dessus en écrivant dans un fichier texte les affichages qu'il produira à l'exécution.
2. Appeler votre enseignant.e une fois que c'est terminé pour vérification.

**Dans les exercices qui suivent, vous verrez des applications concrètes qui requièrent la conversion de types.**

### Exercice 5 : Suivant !

On souhaite disposer d'un programme qui nous indique quel est le caractère suivant dans la table ascii pour un caractère donné. Cette fois, le caractère de départ ne sera pas directement initialisé "en dur" dans une variable mais résultera d'une saisie utilisateur. Pour ce faire, on fait un appel à la fonction `readChar()`, qui ne prend pas de paramètre et retourne le premier caractère saisi par l'utilisateur au clavier.

Copiez le programme ci-dessous :

```
class CaractereSuivant extends Program {

  void algorithm(){
    print("Entrez un caractère");
    char c = readChar();

    char suivant = ; // <- À COMPLÉTER
    println("Le caractère après " + c + " est : " + suivant );
  }

}
```

1. Complétez ce programme afin d'obtenir l'affichage suivant à l'exécution pour les deux scénarios qui suivent :

```
Entrez un caractère : a
Le caractère après a est : b
```

```
Entrez un caractère : Z
Le caractère après Z est : [
```

   En gras, les entrées de l'utilisateur au cours de l'exécution du programme.

2. Essayez avec au moins un autre caractère que les deux demandés.

### Exercice 6 : Des minuscules aux majuscules et vice-versa

On souhaite disposer d'un programme qui transforme une lettre de l'alphabet des minuscules vers les majuscules et réciproquement. N'oubliez pas qu'avec le code ASCII les différents caractères sont représentés par des nombres. Il est donc possible en appliquant un certain décalage sur un caractère (en lui ajoutant un entier) de le transformer en un autre caractère. Pas besoin de connaître la table ASCII pour réaliser un tel exercice, juste de savoir comment elle est strucutrée [...,A-Z,...,a-z,...].

Copiez le programme ci-dessous :

```
class MajMin extends Program {

  void algorithm(){
    print("Entrez une lettre en minuscule");
    char lettreMin = readChar();

    char enMaj = ; // <- À COMPLÉTER
    println("La lettre " + lettreMin + " en majuscule donne : " + enMaj );

    print("Entrez une lettre en majuscule");
    char lettreMaj = readChar();

    char enMin = ; // <- À COMPLÉTER
    println("La lettre " + lettreMaj + " en minuscule donne : " + enMin );

  }

}
```

1. Complétez ce programme afin d'obtenir l'affichage suivant à l'exécution :

```
Entrez une minuscule : a
La lettre a en majuscule donne : A
Entrez une majuscule : Z
La lettre Z en minuscule donne : z
```

   En gras, les entrées de l'utilisateur au cours de l'exécution du programme.

2. Essayez avec au moins une autre minuscule et une autre majuscule.

### Exercice 7 : Rendre efficacement la monnaaie

Sachant que l'on dispose de coupures de 20, 10, 5, 2 et 1 euro, comment procéder pour rendre la monnaie de la manière la plus efficace possible (ie. en rendant le moins de coupure possible) ?

On supposera que l'on a en entrée un nombre représentant la somme à rendre et que l'on affiche en sortie le nombre minimum de billets de 20 euros, de 10 euros, de 5 euros et de pièces de 2 et de 1 euro à rendre. Programme à réaliser sans le mot-clé `if`.

Voici le squelette qu'il vous faut compléter :

```
/**
* Ce programme détermine le nombre minimal de coupures
* à restituer pour une somme donnée. Les coupures utilisables
* sont les billets de 20, 10, 5 et les pièces de 2 et 1 euros.
*
* @author yann.secq@univ-lille1.fr
*/
class RenduMonnaie extends Program {

   void algorithm(){
     int somme, nb20, nb10, nb5, nb2, nb1, reste;
     print("Quelle est le montant que vous souhaitez rendre en monnaie ?");
     somme = readInt();
     // à vous de compléter ce qui suit par les calculs permettant le nombre de chaque coupure
nécessaire.




     println("Nombre de billets de 20 : " + nb20);
     println("Nombre de billets de 10 : " + nb10);
     println("Nombre de billets de 5 : " + nb5);
     println("Nombre de billets de 2 : " + nb2);
     println("Nombre de billets de 1 : " + nb1);
   }
}
```

1. Ecrivez le programme permettant de rendre la monnaie en un minimum de coupure. Voici 2 exemples d'exécutions attendues :

```
 Quelle est le montant que vous souhaitez rendre en monnaie ? 4
Nombre de billets de 20 : 0
Nombre de billets de 10 : 0
Nombre de billets de 5 : 0
Nombre de billets de 2 : 2
Nombre de billets de 1 : 0
```

```
 Quelle est le montant que vous souhaitez rendre en monnaie ? 38
Nombre de billets de 20 : 1
Nombre de billets de 10 : 1
Nombre de billets de 5 : 1
Nombre de billets de 2 : 1
Nombre de billets de 1 : 1
```

2. Combien de variables avez-vous utilisé ? Est-il possible d'utiliser moins de variables ? Si c'est le cas, quel est le nombre minimal de variables nécessaire ! :)
3. Essayer votre programme avec au moins 2 autres exemples de votre choix.

### Un peu d'aléatoire...

Dans ces trois exercices, nous aurons besoin de programmes qui se comportent aléatoirement. Autrement dit, des exécutions avec les mêmes entrées pourront produire des résultats différents. Pour ce faire, on va recourir à la fonction

`random()` qui renvoie un nombre réel aléatoire entre 0 inclus et 1 exclus, de type `double`. Par exemple, après l'instruction `double alea = random();`, la variable `alea` contient une valeur aléatoire entre 0 et 1, qui peut aussi bien être `0.34522876893747156` que `0.7983732526723374` ou n'importe quel autre nombre de l'intervalle $[0, 1[$.

### Exercice 8 : Des chiffres...

On souhaite créer le programme `ChiffreAuHasard` qui affiche un nombre à un chiffre (une valeur entre 0 et 9) au hasard. Sachant que l'appel à `random()` nous renvoie une valeur dans l'intervalle de réels $[0; 1[$, trouver les opérations à faire pour arriver dans l'intervalle d'entiers $[0; 9]$, programmez ensuite l'algorithme `iJava` correspondant dans un programme nommé `ChiffreAuHasard`.

### Exercice 9 : ... et des lettres !

De manière similaire à l'exercice précédent, on souhaite désormais créer un programme `LettreAuHasard` qui affiche une lettre au hasard parmi les majuscules (par exemple pour jouer au mot le plus long en l'appelant autant de fois que de lettres souhaitées).

### Exercice 10 : Des dés !

1. Créer un programme `De` simulant un dé à 6 faces. Il affichera une valeur aléatoire entre 1 et 6 à son exécution.
2. Assurez-vous avec une dizaine d'essais que votre programme produit bien toutes valeurs possibles et rien que celles-ci.
3. Modifiez ce programme de manière à ce qu'il demande à l'utilisateur un dé à combien de faces il veut lancer et affiche le résultat. Pour ce faire, on pourra utiliser la fonction `readInt` fonctionnant sur le même principe que `readChar` vue précédemment mais renvoyant un entier, comme son nom l'indique.

## Prolongement

Les exercices qui suivent sont proposés pour celles et ceux qui veulent s'entraîner ou aller plus loin.

### Exercice 11 : Par ici la monnaie

De retour de vacances passées en partie en Angleterre, Alice et Bob ont dans leur bourse commune 59 livres Sterling et 42 euros. Afin de partager en deux la somme restante, ils souhaitent savoir la valeur totale en euros qu'ils détiennent.

**Q1.** Écrire le programme `Conversion` dont la fonction principale `algorithm` :

1. stocke dans une variable `bourse` le résultat de la multiplication de 59 par une constante `COURS_LIVRE` égale à 1,09 (car 1 livre sterling = 1.09 euros).
2. lui ajoute 42
3. affiche à l'écran le résultat total puis la part qui revient à chacun.

**Q2.** Quelques mois plus tard, Alice et Bob partent voir des amis en Suisse et en profitent pour aller en Pologne. Ils se retrouvent désormais avec des francs suisses et des zlotys (la monnaie polonaise). Écrivez un programme qui permet de convertir n'importe quelle somme d'une monnaie en une autre étant donné le taux de conversion entre ces monnaies. Pour saisir une somme et un taux, on pourra faire appel à la fonction `readDouble` qui, à la manière de `readChar`, ne prend aucun paramètre et retourne le double entré par l'utilisateur au clavier (par exemple `0,123`, attention à bien utiliser une virgule).

### Exercice 12 : Additionnatrice

Réalisez un programme `Additionnatrice` qui demande à l'utilisateur d'entrer deux réels et en affiche la somme.

### Exercice 13 : Heure aléatoire

Réalisez un programme `HeureAleatoire` qui affiche une heure aléatoire au format HH:MM entre 00:00 et 23:59. Dans le cas d'un nombre d'heure ou de minutes inférieur à 10, on pourra omettre les zéros. Par exemple, 8:5 sera accepté pour 08h05

### Exercice 14 : Caractère aléatoire dans un intervalle

Réalisez un programme `CaractereAuHasard` qui demande à l'utilisateur d'entrer deux caractères de la table ASCII et en retourne un au hasard entre ceux-ci, le premier inclus et le dernier exclus.

### Exercice 15 : Chrono

À l'aide de la documentation de la fonction `getTime`, réalisez un programme `Chrono` qui indique combien de secondes et millisecondes se sont écoulées entre le démarrage du programme et le moment où l'utilisateur entre un caractère.