

## Module 3 – Aide à la Réussite : EP 2 Soutien en Programmation

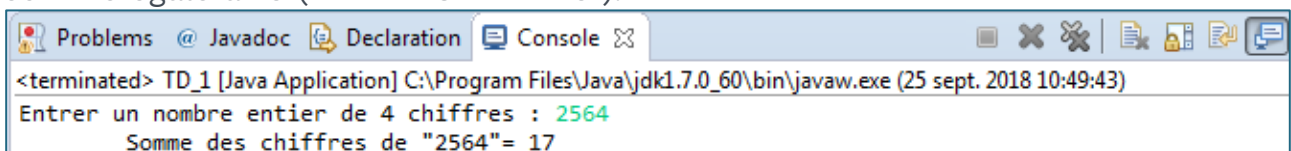
### Travaux Dirigés (1), Licence 1ère Année

### Instructions Simples et Types Élémentaires

#### Exercice 1 Somme des chiffres d'un entier

- 1.1. En utilisant les instructions `/` et `%`, écrivez un programme qui demande à l'utilisateur d'introduire un entier de 4 chiffres et qui affiche la somme de ces derniers.

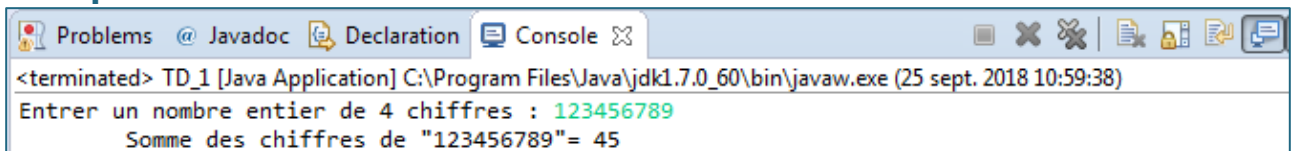
**Exemple :** Si le nombre introduit est **1234**, le programme devrait afficher une somme égale à **10** ( $1 + 2 + 3 + 4 = 10$ ).



```
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 10:49:43)
Entrer un nombre entier de 4 chiffres : 2564
Somme des chiffres de "2564" = 17
```

- 1.2. Généralisez le programme de façon qu'il fasse la même chose pour un entier d'un quelconque nombre de chiffres.

**Exemple :**

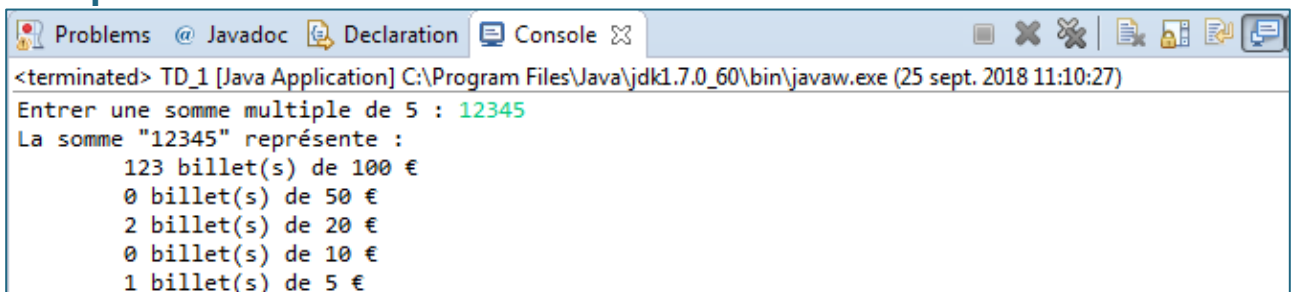


```
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 10:59:38)
Entrer un nombre entier de 4 chiffres : 123456789
Somme des chiffres de "123456789" = 45
```

#### Exercice 2 Quantité de billets représentant une somme d'argent

Écrivez un programme qui demande à l'utilisateur d'introduire une valeur (multiple de 5) représentant une somme d'argent et qui calcule et affiche le nombre de billets de 100, 50, 20, 10 et 5 euros qu'elle représente (en privilégiant le minimum de billets).

**Exemple**



```
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 11:10:27)
Entrer une somme multiple de 5 : 12345
La somme "12345" représente :
    123 billet(s) de 100 €
     0 billet(s) de 50 €
     2 billet(s) de 20 €
     0 billet(s) de 10 €
     1 billet(s) de 5 €
```

#### Exercice 3 Génération aléatoire d'un nombre

La méthode **Math.random()** donne un réel aléatoire positif, strictement inférieur à 1 (**[0, 1[**).

En utilisant la méthode **random()** de la classe **Math**, générez les nombres suivants :

**3.1.** Un réel compris dans l'intervalle **[min, max[**, où **min** et **max** sont des réels introduits par l'utilisateur.

**Exemple**

```

Problems @ Javadoc Declaration Console
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 11:22:31)
Génération d'un nombre réel.
    Entrer la borne min (comprise) : 12,362
    Entrer la borne max (non comprise) : +14,524
Le nombre réel généré dans l'intervalle [ 12.362 , 14.524 [ est : 12.723119842027003
    
```

**3.2.** Un entier compris dans l'intervalle **[10, 15]**.

**Exemple**

```

Problems @ Javadoc Declaration Console
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 11:32:14)
Le nombre entier généré dans l'intervalle [ 10 , 15 ] est 14
    
```

**3.3.** Un entier compris dans l'intervalle **[-10, 15]**.

**Exemple**

```

Problems @ Javadoc Declaration Console
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 11:38:20)
Le nombre entier généré dans l'intervalle [ -10 , 15 ] est -7
    
```

**3.4.** Un entier compris dans l'intervalle **[min, max]**, où **min** et **max** sont des entiers introduits par l'utilisateur.

**Exemple**

```

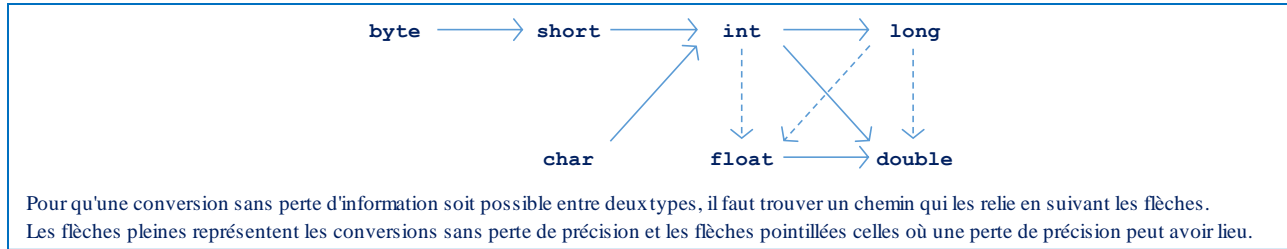
Problems @ Javadoc Declaration Console
<terminated> TD_1 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (25 sept. 2018 11:44:43)
Génération aléatoire d'un nombre entier.
    Entrer la borne min : -33
    Entrer la borne max : +66
Le nombre entier généré dans l'intervalle [ -33 , 66 ] est : -19
    
```

**Exercice 4 Déclarations, Types et Expressions**

Rappel des types de base de Java :

Type	Désignation	Longueur	Défaut	Valeurs
boolean	valeur logique : true ou false	1 bit	false	true ou false
byte	octet signé	8 bits	0	-128 à 127
short	entier court signé	16 bits	0	-32768 à 32767
char	caractère Unicode	16 bits	\u0000	\u0000 à \uFFFF
int	entier signé	32 bits	0	-2147483648 à 2147483647
float	virgule flottante simple précision (IEEE754)	32 bits	0.0	1.401e-045 à 3.40282e+038
double	virgule flottante double précision (IEEE754)	64 bits	0.0	2.22507e-308 à 1.79769e+308
long	entier long	64 bits	0	-9223372036854775808 à 9223372036854775807

Rappel sur les conversions :



**5.1.** Parmi les déclarations de variables suivantes, indiquez celles qui sont valides et celles qui ne le sont pas. Vous pourrez les vérifier dans un programme.

- |                                      |                                    |                                  |
|--------------------------------------|------------------------------------|----------------------------------|
| 1. <code>int i = 0;</code>           | 7. <code>char c = a;</code>        | 13. <code>float f = 0.1;</code>  |
| 2. <code>short j;</code>             | 8. <code>char c = 0x41;</code>     | 14. <code>double d = 0.1;</code> |
| 3. <code>long l1, l2 = 0, l3;</code> | 9. <code>char c = '\u20AC';</code> | 15. <code>double d = 0;</code>   |
| 4. <code>short j = 60000;</code>     | 10. <code>boolean b = true;</code> | 16. <code>float f = 0x10;</code> |
| 5. <code>int i = 0x10;</code>        | 11. <code>boolean b = 0;</code>    | 17. <code>double d = .1;</code>  |
| 6. <code>char c = 'a';</code>        | 12. <code>real r = 0.1;</code>     | 18. <code>int i = 'a';</code>    |

Soient les déclarations suivantes :

```
int i, j, k ;
double x, y, z ;
char c ;
boolean b ;
```

**5.2.** Indiquez le type de chacune des expressions suivantes (vous pouvez les tester dans un programme Java, en reprenant les déclarations ci-dessus et en affectant une valeur initiale à chacune des variables) :

- |                         |                          |                                      |   |
|-------------------------|--------------------------|--------------------------------------|---|
| 1. <code>x</code>       | 7. <code>i + 2</code>    | 13. <code>x &lt; y</code>            | 19. <code>i &gt; j &amp;&amp; k &gt; j</code> |
| 2. <code>2</code>       | 8. <code>x + i</code>    | 14. <code>i % j + y</code>           | 20. <code>x + y * i</code>                    |
| 3. <code>i = j</code>   | 9. <code>x / 2</code>    | 15. <code>i / j + y</code>           | 21. <code>i = c</code>                        |
| 4. <code>i == j</code>  | 10. <code>x / 2.0</code> | 16. <code>i &gt; j &gt; k</code>     | 22. <code>x = (int) y</code>                  |
| 5. <code>x + 2.0</code> | 11. <code>i / 2</code>   | 17. <code>i &amp;&amp; b</code>      | 23. <code>c = (char)((int)c+1)</code>         |
| 6. <code>x + 2</code>   | 12. <code>i / 2.0</code> | 18. <code>i == j &amp;&amp; b</code> | 24. <code>i++</code>                          |

**5.3.** Écrivez en Java les expressions suivantes et testez-les en vérifiant que vous avez les mêmes résultats pour les mêmes valeurs de variables que dans l'exemple ci-dessous :

(1)	(2)	(3)
$a^2 - c + \frac{a}{bc + \frac{c}{d + \frac{e}{f}}}$	$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$	$\frac{\frac{1}{a} + \frac{1}{b}}{c + d}$

