Introduction à la Programmation 1

Objectifs:

- Découverte du type `String`.
- Comprendre qu'il y a des types di□érents.
- Maîtriser les expressions booléennes dans les conditions.
- Comprendre l'utilisation des boucles avec dépendances.

1 Les chaînes de caractères : le type `String`

Les chaînes de caractères [Cours]

- Les chaînes de caractères représentent du texte.
- Les constantes de chaînes de caractères sont écrites entre guillemets doubles.
- `"Mr Robot"` est un exemple de chaîne.
- Pour introduire un guillemet dans une chaîne, on écrit `\"`.
- La chaîne vide s'écrit `""`.
- L'opérateur de concaténation de deux chaînes est +.
- Dans une chaîne, la séquence `\n` représente le passage à une nouvelle ligne.

Exercice 1 (De premiers exemples, ⋆)
Quelle est la valeur des variables après l'exécution des instructions suivantes ?

```
String s = " rentre chez lui.";
String s2 = "Paul";
String s3 = "Michel";
String s4 = s2 + s;
String s5 = s3 + s;
String s6 = s2 + " et " + s3 + " rentrent chez eux.\n";
```

□

2 Utilisation des di□érents types `int` et `String`

Les chaînes de caractères et les entiers sont des types di□érents [Cours]

— Une variable `x` de type `int` ne peut pas contenir de chaîne de caractères.
— Une variable `s` de type `String` ne peut pas contenir d'entier.
— En général, une fonction ou une procédure attend des arguments de types spécifiés lors de sa déclaration.
— Par exemple, la procédure « `void printInt (int a)` » ne peut pas être utilisée avec un argument de type `String`. Ainsi, écrire "`printInt ("Yoda")` provoquera une **erreur de typage** au moment de la compilation.

Exercice 2 (Des exemples simples avec des variables de di□érents type, ⋆)

1. *Quelle est la valeur de la variable* `ch` *après l'exécution des instructions suivantes ?*

```
int x = 3;
String ch = "Salut";
if (x <= 2) {
    ch = "Hallo";
} else {
    ch = "Hi";
}
```

2. *Que fait la suite d'instructions suivantes ?*

```
int x = 3;
String st = "Ca va ?";
String ch = "";
x = x / 2;
if (x == 1) {
    ch = "How are you ?";
} else {
    ch = "Comment allez-vous ?";
}
ch = ch + "\n";
printString (ch);
```

□

Exercice 3 (Deux représentations très di□érentes de la même chose, ⋆⋆)

— *Que vaut l'expression* `"41" + "1"` *?*
— *Que vaut l'expression* `41 + 1` *?*
— *Quelle di érence faites-vous donc entre la chaîne de caractères* `"42"` *et l'entier* 42 *? Dans quelles situations utiliser l'une ou l'autre de ces représentations de l'entier* 42 *?*

□

Exercice 4 (Trouver le bon type, ⋆⋆⋆)

Dans les fonctions suivantes, remplacez les `[...]` *par les bons types :*

```

public static [...] f ([...] x) {
    [...] y = x % 2 ;
    if (y == 0) {
        printString ("Argument pair");
    } else {
        printString ("Argument impair");
    }
}
```

```
10
11 public static [...] g ([...] x) {
12     [...] y = x % 2 ;
13     if (y == 0) {
14         return "Argument pair";
15     } else {
16         return "Argument impair";
17     }
18 }
19
20 public static [...] h ( [...] x, [...] y) {
21     return (x + y);
22 }
23
24 public static [...] id ( [...] x) {
25     return x;
26 }
```

code/ExoInference.java

□

3 Fonctions avec le type `String`

Les fonctions peuvent utiliser le type `String` [Cours]

— Une fonction peut renvoyer une valeur de type `String`.
— Les valeurs de type `String` peuvent être utilisées comme paramètres d'une fonction.

On considère les deux fonctions données ci-dessous :

```
1  public static String sandwich (String s, String s2) {
2      return s + s2 + s;
3  }
4
5  public static String neymarOrNot (int x) {
6      String st = "";
7      if (x == 10) {
8          st = "C'est le numéro de Neymar.";
9      } else {
10         st = "Ce n'est pas le numéro de Neymar.";
11     }
12     return st;
13 }
```

Elles peuvent être utilisées de la façon suivante :

```
1 String s = sandwich ("Hello", "World!\n");
2 String s2 = sandwich (s, "Ca va ?");
3 String s3 = neymarOrNot (1);
```

Voici quelques fonctions et procédures que nous vous fournirons couramment pour écrire des programmes manipulant des chaînes de caractères :

— Calculer la longueur d'une chaîne de caractères avec "`int stringLength (String s)`".
— Obtenir le $(i+1)$-ème caractère d'une chaîne avec "`String characterAtPos (String s, int i)`" (Le premier caractère est à la position 0 et si i est plus grand que la longueur, la fonction renvoie la chaîne vide "".) ;

— Transformer un entier en chaîne de caractères avec "`String intToString (int a)`" ou (certaines) chaînes de caractères en entiers avec "`int stringToInt (String s)`"
— Afficher une chaîne de caractères sur la sortie standard avec "`void printString (String s)`".
(De telles fonctions vous seront fournies ce semestre.)

Exercice 5 (Utilisation de fonctions données, ⋆)

1. *On considère la fonction suivante :*

```
public static String addA (String ch) {
    return (ch + "A");
}
```

Que vaut la variable `st` *après exécution du code suivant ?*

```
String s = "Ma lettre finale est ";
String st = addA (s);
```

2. *Quelle est la valeur de la variable* `a` *après les instructions suivantes où "*`int stringLength(String s)`*" est la fonction décrite ci-dessus ?*

```
String ch = "Ceci est ";
String ch2 = "une chaine";
ch = ch + ch2;
int a = stringLength (ch);
```

3. *Que vaut la variable* `cha` *dans l'exemple suivant où "*`String characterAtPos (String s, int i)`*" est la fonction décrite ci-dessus ?*

```
String ch = "Avec consonnes";
String cha = characterAtPos (ch, 0);
cha = cha + characterAtPos (ch, 2);
cha = cha + characterAtPos (ch, 6);
cha = cha + characterAtPos (ch, 9);
cha = cha + characterAtPos (ch, 12);
```

□

Exercice 6 (Modification d'une fonction, ⋆⋆)

On considère la fonction suivante où la fonction "`String intToString(int x)`*" est celle décrite ci-dessus.*

```
public static String message (int x) {
    return intToString (x);
}
```

Et on considère la liste d'instructions suivante :

```
int a = 4;
int b = a * a;
a = b - a;
String s = message (a);
printString (s);
```

1. *Quelles sont les valeurs des variables de ce programme à la fin de son exécution ?*

2. *Qu'a chent ces instructions ?*
3. *Est-il possible de modifier la fonction "*`String message (int x)`*" de telle sorte que le programme a che à la fin "Le résultat du calcul est n." (où n est la valeur contenue dans l'argument transmis à* `message`*) ?*
4. *Définissez une fonction "*`int square (int n)`*" qui calcule le carré du nombre donné en paramètre et utilisez-la pour a cher une ligne du type $n * n = n^2$ (où n sera remplacé par sa valeur).*

□

4 Expressions booléennes dans les conditionnelles

Des expressions booléennes plus complexes dans les tests [Cours]

- Une expression booléenne a le type `boolean`.
- Une expression booléenne peut s'évaluer en l'une des deux valeurs `true` ou `false`.
- Une expression avec un **et** (en Java `&&`) est vraie si les expressions à gauche et à droite du **et** sont vraies.
- Une expression avec un **ou** (en Java `||`) est vraie si au moins une des expressions à gauche et à droite du **ou** est vraie.
- Une expression avec une négation (en Java `!`) est vraie si l'expression après la négation est fausse.
- L'opérateur unaire `!` est prioritaire sur l'opérateur `&&` qui est lui-même prioritaire sur l'opérateur `||`.
- Exemples d'expressions utilisant ces opérateurs :
  - `(2 < x && x <= 5)`
  - `(x == 1 || x == 2 )`
  - `!(x == 0)` (qui est équivalent à "`x != 0`")
- On peut combiner ces opérateurs, par exemple en écrivant "`(x == 1 || x == 2) && y > 5`"
- Attention, on ne peut pas tester l'égalité de deux chaînes de caractères avec `==`. Pour cela, on vous donne une fonction "`boolean stringEquals (String s1, String s2)`" à utiliser dans les tests.
- Voici un exemple :

```
public static void profiler (String name) {
  if (stringEquals (name, "Paul") || stringEquals (name, "Pierre")) {
      printString ("On a trouve Paul ou Pierre.");
  } else {
      printString ("Ce n'est ni Paul, ni Pierre.");
  }
}
```

Exercice 7 (Savoir évaluer une condition, ⋆)
Quelle est la valeur de la variable `x` *après la suite d'instructions suivante ?*

```
int a = 2;
a = a * a * a * a + 1;
int b = a / 2;
int c = a - 1;
int x = 0;
if ((b == 0 ) && (c == 16)) {
    x = 1;
} else {
    x = 2;
}
```

□

Exercice 8 (Équivalence d'expressions, ⋆⋆)
On dit que deux expressions booléennes sont équivalentes *quand ces deux expressions s'évaluent toujours vers la même valeur booléenne pour toutes les valeurs possibles des variables qui apparaissent dans ces expressions.*

Par exemple, "`x > 10 && x < 12`*" est équivalente à "*`x == 11`*". Par contre, "*`x > y`*" et "*`x != y`*" ne sont pas équivalentes.*

Dire si les expressions suivantes sont équivalentes :

1. "`x > y || x < y`" *et* "`x != y`";
2. "`x != 3 && x != 4 && x != 5`" *et* "`x <= 2 || x >= 6`";
3. "`x ==  y && x == z`" *et* "`x == z`";
4. "`x ==  y && x == z`" *et* "`x == y && y == z`".

□

Exercice 9 (Simplification des expressions booléennes, ⋆⋆)

Simplifier *une expression booléenne consiste à la remplacer par une autre expression booléenne équivalente qui est plus courte. Simplifier les expressions suivantes :*

— `x > 5 && x > 7`
— `x == y && x == z && y == z`
— `x == 17 || (x != 17 && x == 42)`
— `x > 5 || (x <= 5 && y > 5)`

□

5 Instructions conditionnelles imbriquées

Instructions composées [Cours]

— Certaines des instructions présentées sont des instructions *composées* : elles peuvent contenir d'autres instructions.
— Les instructions conditionnelles et les boucles sont des instructions composées.
— Par exemple :

```
int x = 1;
int y = 2;
int z = 0;
if (x == 1) {
    if (y == 2) {
        z = 3;
    } else {
        z = 5;
    }
} else {
    z = 7;
}
```

L'exécution du programme précédent a pour effet d'affecter la valeur 3 à la variable z.
— On a le droit d'imbriquer des instructions dans d'autres instructions à volonté : des conditionnelles dans des boucles dans des conditionnelles, etc ...
— Avec des instructions imbriquées sur plusieurs niveaux, il est absolument indispensable (dans l'intérêt des lecteurs humains) de suivre strictement une discipline d'indentation du code source.
— On parlera des boucles imbriquées lors de la séance 3.

Exercice 10 (Suppression des opérateurs booléens, ⋆⋆)
Réécrire les suites d'instructions suivantes en n'utilisant que des conditions sans opérateur booléen :

1.

```
public static void f (int l) {
  int x = 0;
  if (l != 0 && l <= 10) {
      x = 1;
  } else {
      x = 2;
  }
}
```

Quelle est la valeur finale de `x` *pour de valeurs de* `l` *dans* $\{0, 5, 15\}$ *? Vérifier que votre réécriture du code est compatible.*

2.

```
public static void g (int a) {
  int b = 2 * a;
  b = a - a;
  if (b == a || b > 43) {
      b = 0;
  } else {
      b = 1;
  }
}
```

Quelle est la valeur finale de `b` *pour de valeurs de* `a` *dans* $\{0, 25, 50\}$ *? Vérifier que votre réécriture du code est compatible.*

3.

```
public static void h (int c, int d) {
  int e = 0;
  if (!(d == 6) && (c >= 2)) {
      e = 2;
  } else {
      e = 3;
  }
}
```

Quelle est la valeur finale de `e` *pour de valeurs de* `(c,d)` *dans* $\{(0,0), (0,6), (6,0), (6,6)\}$ *? Vérifier que votre réécriture du code est compatible.*

□

6 Boucles

Exercice 11 (Pendu à l'envers, ⋆⋆)

1. *Écrire une fonction qui prend en argument une chaîne de caractères et a che une suite de tirets (-) de même longueur.*

   Contrat:
   argument = "amphitheatre" → *a chage : - - - - - - - - - - -*

2. *Écrire une fonction qui prend en argument une chaîne de caractères et l'a che à l'envers.*

Contrat:
argument = "amphitheatre" → *a chage : ertaehtihpma*

□

Boucles et conditionnelles [Cours]

Le corps d'une boucle peut contenir n'importe quelle instruction. En particulier, il peut contenir une conditonnelle. Par exemple, le programme suivant affiche `"Pair."` pour les valeurs de `i` qui sont des entiers pairs et `"Impair."` pour les valeurs de `i` qui sont des entiers impairs.

```
for (int i = 0; i < 100; i++) {
    if (i % 2 == 0) {
        printString ("Pair.");
    } else {
        printString ("Impair.");
    }
}
```

Dans le programme précédent les instructions qui sont effectuées à chaque itération *dépendent de la valeur du compteur de boucle* `i`. On ne se contente donc pas de répéter toujours la même chose : ce que l'on fait à chaque itération dépend du nombre d'itérations déjà effectuées, c'est-à-dire de l'indice de l'itération courante, représenté par la valeur du compteur `i`.

7 Fonctions utilisées

Liste des fonctions [Cours]

```
/*
 * Affiche un entier sur l'écran.
 * x est l'entier à afficher.
 */
/* Affiche un entier. */
public static void printInt (int x) {
    System.out.print (x);
}

/* Affiche une chaîne de caractères. */
public static void printString (String s) {
    System.out.print (s);
}

/* Renvoie la longueur d'une chaîne de caractères. */
public static int stringLength (String s) {
    return s.length ();
}

/* Transforme un entier x en une chaîne de caractères.
 * Renvoie la chaîne de caractères représentant la valeur de x.
 */
public static String intToString(int x) {
    return String.valueOf (x);
}

```

```java
/*
 * Renvoie une chaîne constituée du caractère se trouvant à la
 * position $i$ de la chaîne s.
 * Si i est négatif ou en dehors de la chaîne, renvoie la chaîne vide.
 */
public static String characterAtPos(String s,int i) {
    String res = "";
    if(i >= 0 && i < s.length ()) {
        res = String.valueOf (s.charAt (i));
    }
    return res;
}

/*
 * Teste si deux chaînes de caractères st1 et st2 ont le même contenu.
 * Renvoie vrai si st1 et st2 sont égales.
 */
public static boolean stringEquals (String st1, String st2) {
    return st1.equals(st2);
}
```

8 Do it yourself

Pour certains exercices, un "contrat" est proposé : ce sont des tests que nous vous fournissons pour vérifier votre réponse. Si votre réponse ne passe pas ces tests, il y a une erreur ; si votre réponse passe les tests, rien n'est garanti, mais c'est vraisemblablement proche de la réponse. Quand aucun contrat n'est spécifié, à vous de trouver des tests à faire.

Exercice 12 (Concaténation, ⋆)
Écrire une fonction qui prend en paramètre une chaîne de caractères `toto` *et a che une ligne commençant par* `bonjour`, *suivi du contenu de* `toto`.

> ***Contrat:***
> *si* `toto` *a la valeur* `"toi"`*, l'appel à la fonction doit a cher*
> `bonjour, toi`

□

Exercice 13 (Concaténation et somme, ⋆⋆)
Que valent les variables `x` *et* `s` *après les instructions suivantes ?*

```java
int x = 0;
String s = "";
for (int n = 0; n < 10; n++) {
    x = x + 1;
    s = s + "1";
}
```

□

Exercice 14 (Condition et division entière, ⋆)
Écrire une condition permettant de tester si le tiers de l'entier `x` *appartient à l'intervalle* $[2; 8]$.

> ***Contrat:***

x = 0 $\rightarrow$ *false*
x = 6 $\rightarrow$ *true*
x = 9 $\rightarrow$ *true*
x = 25 $\rightarrow$ *false*

□

Exercice 15 (Condition, ⋆)
Écrire une condition permettant de tester si les entiers `a`, `b` *et* `h` *peuvent correspondre aux longueurs des côtés d'un triangle rectangle, où* `h` *serait la longueur de l'hypothénuse.*

Contrat:
(a,b,h) = (3,4,5) $\rightarrow$ *true*
(a,b,h) = (1,2,3) $\rightarrow$ *false*

□

Exercice 16 (Somme des cubes, ⋆)
Écrire une boucle permettant de calculer la somme des cubes des entiers de 1 à 100. □

Exercice 17 (Ordinaux anglais, ⋆⋆⋆)
On s'intéresse aux ordinaux anglais abrégés*, où le nombre (le cardinal) est écrit en chi res :*

1st, 2nd, 3rd, 4th,. . ., 9th, 10th, 11th, 12th,. . ., 19th, 20th, 21st, 22nd, 23rd,. . .

Pour déterminer le su xe, on regarde le dernier chi re du nombre : si c'est 1, on ajoute le su xe "st"; si c'est 2, le su xe est "nd"; si c'est 3, le su xe est "rd"; sinon le su xe est "th". Il y a cependant une exception : si l'avant-dernier chi re du nombre est 1, le su xe est toujours "th".

Écrire une fonction `printOrdinal` *qui prend un entier de type* `int` *en paramètre et a che l'ordinal anglais abrégé correspondant.*

Contrat:
1 $\rightarrow$ *1st*
12 $\rightarrow$ *12th*
23 $\rightarrow$ *23rd*
32 $\rightarrow$ *32nd*
44 $\rightarrow$ *44th*

□

Exercice 18 (Simplification de code, ⋆)

1. *Dire, en justifiant, ce que fait la fonction suivante selon les valeurs de* x *et* y*.*

```
public static int f (int x, int y) {
    if (((x <= y) || (x >= y + 1))) {
        if ((x <= y) && (x < y)) {
            if (x < -x) {
                int a = -x;
                return a;
            } else {
                return x;
            }
        } else {
            if (y * y * y < 0) {
                return -y;
            }
        }
    }
    return y;
}
```

2. *Écrire la même fonction de manière plus simple.*

□

Exercice 19 (Mention, ⋆)

Écrire une conditionnelle afin de stocker dans une chaîne de caractères `s` *la mention correspondant à la note (entière) contenue dans la variable* `n` *de type* `int`*.*

Rappel : entre 10 inclus et 12 exclu, la mention est passable*; entre 12 inclus et 14 exclu;* assez bien *entre 14 inclus et 16 exclu, et* très bien *au-delà de 16.* □

Exercice 20 (Lignes et pointillés, ⋆⋆)

1. *Écrivez une fonction qui prend en paramètre un entier supposé positif* n *et qui a che une ligne d'astérisques « ∗ » de longueur* n *puis va à la ligne. Par exemple, si* n *vaut 7, votre fonction a chera :*