Introduction à la Programmation 1

Séance 2 de cours/TD

Université Paris-Diderot

Objectifs:

- Découverte du type String
- Comprendre qu'il y a des types différents
- Maîtriser les conditions 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 constantes sont entre guillemets doubles
- Attention aux guillemets dans la chaîne
- Opérateur de concaténation : +
- Des caractères spéciaux comme \n pour aller à la ligne
- Des variables de type String
- Les chaînes de caractères peuvent donc être utilisées pour stocker des messages que l'on souhaite afficher à l'écran avec par exemple la procédure void printString(String s) que l'on vous donne

Le niveau de difficulté est ⋆, ⋆⋆ ou ⋆⋆⋆.

Exercice 1 (Des exemples simples, ⋆)
Quelle est la valeur des variables à la suite des instructions ?

```
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 différents types int et String

Les chaînes de caractères et les entiers sont des types différents [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
– On peut utiliser les deux types dans une même suite d'instructions

```
String s = "Ceci est une chaine";
int a = 2;
int b = a * a;
```

Exercice 2 (Des exemples simples avec des variables de différents type, ⋆)

1. *Quelle est la valeur de la variable* ch *à la suite des instructions ?*

```
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);
```

□

3 Fonctions avec le type String

Les fonctions peuvent utiliser le type String [Cours]

– Une fonction peut retourner une valeur de type String
– Les valeurs de type String peuvent être utilisées comme paramètres d'une fonction ;
– Les deux types ne sont pas forcément identiques – on peut avoir une fonction qui prend en paramètre une chaîne de caractères et renvoie un entier ou une fonction prenant en paramètre un entier et retournant une chaîne de caractères.

On considère les deux fonctions données ci-dessous :

```
static String concatenate(String s, String s2) {
    return s + s2;
}

static String neymarOrNot(int x) {
```

```
    String st = "";
    if(x == 10) {
        st = "C'est le numero de Neymar";
    } else {
        st = "Ce n'est pas le numero de Neymar";
    }
    return st;
}
```

Elles peuvent être utilisées de la façon suivante :

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

Les fonctions peuvent être utilisées comme par exemple pour (de telles fonctions vous seront fournies cette année) :
- Calculer la longueur d'une chaîne de caractères `int theLength(String s)`
- Obtenir le (i+1)-ème caractère d'une chaîne `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 `String intToString(int a)`

Exercice 3 (Utilisation de fonctions données, $\star$)

1. *On considère la fonction suivante :*

```
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 longueur(String s) *est la fonction décrite ci-dessus ?*

```
String ch = "Ceci est ";
String ch2 = "une chaine";
ch = ch + ch2;
int a = theLength(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 4 (Modification d'une fonction, ⋆⋆)
On considère la fonction suivante où la fonction String intToString(int x) *est celle décrite ci-dessus.*

```
static String result(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 = result(a);
printString(s);
```

1. *Quelle valeur affichent ces instructions ?*
2. *Est-il possible de modifier la fonction* String result(int x) *de telle sorte que le programme affiche à la fin "Le resultat du calcul est n" (où n est la valeur contenue dans la variable entière* a*) ?*
3. *Définir une fonction* int square(int n) *qui calcule le carré du nombre donné en paramètre et remplacer à l'endroit adéquat le morceau de code dans les instructions précédentes par l'utilisation de cette fonction.*

□

4 Expressions booléennes dans les conditionnelles

Des conditions plus complexes dans les tests [Cours]

- Trois opérateurs booléens :
  1. le **et** (en Java &&). Une condition avec un **et** est vraie si les deux éléments sont vrais ;
  2. le **ou** (en Java ||). Une condition avec un **ou** est vraie si au moins un des éléments est vrai ;
  3. la négation (en Java !). Une condition avec négation est vraie si la condition après la négation est fausse.
- Exemple de conditions 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 (x == 1 || x == 2) && y >5
- Attention, sur les chaînes de caractères on ne peut pas tester l'égalité de deux chaînes avec ==, pour cela on vous donne une fonction boolean stringEquals(String st1, String st2) à utiliser dans les tests

Voilà un exemple :

```
String nom="Michel";
if(stringEquals(nom, "Paul") || stringEquals(nom, "Pierre"){
    printString("On a trouve Paul ou Pierre");
} else {
    printString("Ce n'est ni Paul, ni Pierre");
}
```

Exercice 5 (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 6 (Équivalence d'expressions, ⋆⋆)
On dit que deux expressions booléennes sont équivalentes *quand les deux s'évaluent toujours vers la même valeur booléenne, peut importe les valeurs des variables. 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 7 (Simplification des expressions booléennes, ⋆⋆)
Simplifier *une expression booléenne signifie 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 à la Session 1 sont des instructions *composées* : elles peuvent contenir d'autres instructions.
– Exemples d'instructions composées déjà vus : les conditionnelles, et les boucles.
– 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 a comme effet d'affecter la valeur 3 à la variable z :

– On a le droit d'imbriquer arbitrairement : des conditionnelles dans des conditionnelles dans des conditionnelles ...
– Avec des instructions imbriquées sur plusieurs niveaux il est absolument indispensable (dans l'intérêt des lecteurs humains) de suivre strictement la politique d'incrémentation de lignes.
– On parlera des boucles imbriquées à la Session 3.

Exercice 8 (Suppression des opérateurs booléens, ⋆⋆)
Réécrire les suites d'instructions suivantes en n'utilisant que des conditions sans opérateur booléen :

1.

```
String text = "Ceci est une phrase.\n";
int l = theLength(texte);
int x = 0;
if(l != 0 && l <= 10){
    x=1;
} else {
    x=2;
}
```

2.

```
int a = 26;
int b = 2 * a;
b = a - a;
if(b == a || b > 43){
    b = 0;
} else {
 b = 1;
}
```

3.

```
int c = 45;
int d = c / 9;
int e = 0;
if(!(d == 6) && (c >= 2){
    e = 2;
} else {
    e = 3;
}
```

□

6 Boucles

Boucles et conditionnelles [Cours]

Le corps d'une boucle peut contenir n'importe quelle instruction ; en particulier, il peut contenir une conditonnelle.

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

7 Fonctions utilisées

Liste des fonctions [Cours]

```
/* 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);
}

/* Retourne la longueur d'une chaîne de caractères. */

public static int theLength(String s) {
    return s.length();
}

/* Transforme un entier x en une chaine de caracteres. */
 * Retourne la chaine de caracteres représentant la valeur de x.
 */

public static String intToString(int x) {
    return String.valueOf(x);
}

/*
 * Retourne 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, retourne la chaîne vide.
 */

public static String characterAtPos(String s,int i) {
    String res = "";
    char a;
    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.
 * Retourne vrai si st1 et st2 sont égales.
 */

public static boolean stringEquals(String st1, String st2){
    return String.equals(st1, st2);
}
```

8 Do it yourself

Exercice 9 (Concaténation, ⋆)
Ecrire une fonction qui prend en paramètre une chaîne de caractères toto *et affiche une ligne commençant par* bonjour, *suivi du contenu de* toto. □

Exercice 10 (Concaténation et somme, ⋆⋆)
Que valent les variables x *et* s *après les instructions suivantes ?*

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

□

Exercice 11 (Condition et division entière, ⋆)
Écrire une condition permettant de tester si le tiers de l'entier x *appartient à l'intervalle* $[2; 8]$. □

Exercice 12 (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.* □

Exercice 13 (Somme des cubes, ⋆)
Écrire une boucle permettant de calculer la somme des cubes des entiers de 1 à 100. □

Exercice 14 (Ordinaux anglais, ⋆⋆⋆)
On s'intéresse aux ordinaux anglais abrégés, *où le nombre (le cardinal) est écrit en chiffres :*

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

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

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

Exercice 15 (Simplification de code, ⋆)

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

```
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 16 (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 ; assez bien *entre 12 inclus et 14 exclu ;* bien *entre 14 inclus et 16 exclu, et* très bien *au-delà de 16.* □

Exercice 17 (Lignes et pointillés, ⋆⋆)

1. *Écrivez une fonction qui prend en paramètre un entier n et qui affiche une ligne d'astérisques « * » de longueur n puis va à la ligne. Par exemple, si n vaut 7, votre fonction affichera :*
   ****
2. *Modifiez votre fonction pour qu'elle affiche une ligne pointillée alternant astérisques et espaces. Par exemple, si n vaut 7, votre fonction affichera :*
   * * * *
   Qu'affiche votre fonction si n est pair ?

□

Exercice 18 (Nombres parfaits, ⋆⋆⋆)

Un entier $n > 1$ est dit parfait *s'il est égal à la somme de ses diviseurs propres (c'est-à-dire autres que lui-même). Par exemple, 6 est parfait car ses diviseurs propres sont 1, 2 et 3, et on a $6 = 1 + 2 + 3$.*

1. *Écrire une fonction* `divisors` *qui prend en entrée un entier n et affiche tous les entiers x qui divisent n.*
2. *Écrire une procédure* `void isPerfect(int n)` *qui prend en entrée un entier n et qui affiche "n est parfait" si n est parfait.*
3. *Écrire une procédure* `void enumPerfect(int m)` *qui prend en paramètre un entier m et affiche pour tous les nombres parfaits tels que $n \leq m$, "m est parfait".*

□