Programmation orientée objet

Les bases de Java

Plan du cours

- Présentation de l'informatique en 2ème année
- Présentation du module
- Présentation de Java
- La syntaxe de base de Java
- Les tableaux
- Les chaînes de caractères

L’informatique en 2ème année de DUT R&T

- En 2ème année de DUT R&T il y a 4 modules d’informatique qui vont vous faire découvrir plusieurs langages.
    - I5 et I6 sont basés sur Java
    - IC1 traite de bash et de PowerShell
    - IC3 utilise XHTML, CSS et PHP
- L’informatique représente donc un élément important de la 2ème année.
- Les différents modules sont très denses, donc il faut travailler chaque cours au fur à mesure et ne pas se laisser “enliser” !

Présentation du module

- Le module I5 “Programmation Orientée Objet” fait partie de l’unité d’enseignement 2, il a un coefficient 2 dans cette UE.

- La compréhension de ce module est nécessaire pour le module I6 “Programmation événementielle et réseau”.

- Ces 2 modules sont basés sur le langage Java, bien que de nombreux concepts soit applicables à d’autres langages (PHP dans le module IC3,...).

- Les TP seront évolués au fur et à mesure des séances. Le contrôle écrit final aura lieu le 9 octobre 2008.

- Le module est donc très rapide, il est **impératif** de ne pas se laisser submerger dès le premier cours.

  “*Compared to the time required to learn to play the piano well or to become fluent in a foreign (natural) language, learning a new and different programming language and programming style is easy.*“ - Bjarne Stroustrup (inventeur du C++)

Principales familles de langages

- Plusieurs familles de langages coexistent actuellement.
- Il est bon de les connaître afin de pouvoir choisir le langage le mieux adapté à chaque problème.
- Parmi les plusieurs familles, 4 vont être détaillées, car les langages abordés en 2ème année en font partie :
  - les langages impératifs,
  - les langages fonctionnels,
  - les langages à balises,
  - les langages orientés objets.

Langages impératifs

- Historiquement, ce sont les premiers langages dont le but était de faciliter l'écriture des programmes.

- Les instructions se regroupent en 4 types : manipulation de la mémoire (assignation, lecture et opérations arithmétiques....), branchements sans condition (déplacement dans la mémoire programme), branchements conditionnels (permettent d'effectuer un saut si une condition est vérifiée) et boucles inconditionnelles (nombre de répétitions fixe) ou conditionnelles (sur un test).

- Ces instructions sont proche du code machine, la compilation est très simple.

- Les langages les plus anciens (comme C, Fortran,...) sont impératifs. Les langages modernes (Python, Java,...) supportent la programmation impérative.

Langages fonctionnels

- La gestion de la mémoire est souvent problématique dans la programmation impérative (“effets de bords”, ...)

- La programmation fonctionnelle propose une approche très différente de la programmation. Aucune affectation n’est prévue (d’un point de vue théorique) pour simplifier la gestion de la mémoire par le programmateur.

- Un programme est une application (au sens mathématique), un chaînage de fonctions simples.

- Les principaux langages fonctionnels sont Lisp et Scheme (présent dans Gimp).

- Dans Gnome, la bibliothèque Guile permet de programmer des applications de manière fonctionnelle.

Langages à balisage

- Les langages à balises sont très différents des langages présentés ci-dessus.
- Ces langages sont utilisés pour décrire des éléments et les relations entre eux.
- L’utilisation principale de ces langages est la structuration de données ou de textes.
- Des relations hiérarchiques (par exemple, titre, section, ...) ou des rôles particuliers (image, lien, ...) sont décrits.
- Le représentant le plus connu est HTML et tous ces descendants comme XML, XHTML, CSS, ...

Langages orientés objets

- Actuellement, le *paradigme* de programmation le plus utilisé est la programmation orientée objet (POO).

- Un programme est composé de briques logicielles élémentaires, les objets qui contiennent les données et qui sont capables de les modifier.

- Ces objets simples peuvent être combinés, modifiés, réutilisés ... pour obtenir un logiciel complexe.

- La POO combinée à UML permet une modélisation simple des problèmes informatiques.

- Les principaux langages contemporains supportent la POO.

- Les grands représentants actuels sont C++, Java, Python, Ruby, Smalltalk, ...

Buts de Java

- Java est lancé par Sun Microsystems en 1995
- Plusieurs objectifs avaient été fixés lors de la création de Java :

1. Langage orienté objet
2. Exécution du même programme sur différentes machines sans recompilation (et donc sans distribution des sources).
3. Support natif des réseaux informatiques
4. Sécurité et notamment sécuriser l’exécution à distance de code
5. Facile à utiliser

Bref historique

- Java a mis du temps à s'installer dans le paysage informatique à cause des lenteur de la première version et de démêlés judiciaires (avec Microsoft,...)

- Java est devenu un logiciel libre (GPL) en novembre 2006. La première version libre est prévue pour 2009 (plutôt 2010...) .

- Java est présent sur toutes les plates-formes grand public (Windows, Linux, OS X, ...)

- La compilation *just-in-time* a résolu les problèmes de lenteur.

Popularité de Java

- Le classement TIOBE (http://www.tiobe.com/tiobe_index/index.htm) est un indicateur de popularité des langages de programmation.

- Depuis de nombreuses années, Java fait partie des langages les plus populaires.

Classement	Langage	Delta (2009-2008)
1	Java	0
2	C	0
3	C++	+1
4	PHP	+1
6	Python	0
7	C#	+1
8	Perl	-1
...	...	...

Java et d'autres langages

- Java a de nombreuses caractéristiques avec d'autres langages, il est :
  - Rapide mais pas autant que le C/C++,
  - Sécurisé mais pas autant que ADA,
  - Portable mais pas autant que Perl,
  - Orienté objet mais pas autant que Smalltalk ou Ruby,
  - ...
- Dans tous ces domaines, il existe un langage plus performant que Java, mais Java regroupe toutes ces capacités en un seul langage.
- Peu de langages disposent d'une bibliothèque standard aussi riche que Java.
- Java est le langage le plus présent sur les périphériques mobiles (comme les téléphones) grâce à J2ME et Android (porté par Google).

Principales caractéristique de Java

- Java est un langage compilé.

- Java est un langage typé, chaque élément de mémoire est associé à un type de donné qui est définit lors de l'écriture du programme et qui ne peut pas varier pendant le déroulement du programme.

- Java mélange plusieurs *paradigmes* de programmation :
  - impératif : les instructions sont exécutées les unes après les autres
  - structuré : un programme est composé d'un ensemble de fonctions élémentaires, le résultat d'une fonction est utilisé par une autre et ainsi de suite...
  - orienté objet : le programme manipule des structures de données stockant des types hétéroclites qu'elles sont capables de modifier.

Avantages de Java pour le programmeur

- Langage très sécurisé (voir sécuritaire...) de la conception à l'exécution.
- Syntaxe cohérente, simple et “agréable” (pas d’opérateur surchargé, pas de préprocesseur, ...)
- Gestion automatique de la mémoire (pointeurs remplacés par les références, ramasse miettes, ...)
- Utilisation facile des threads (programmation multiprocessus), des exceptions (gestion des erreurs), ...
- API (“bibliothèque de fonctions”) fournie très riche : plusieurs milliers de classes.
- Le programme est compilé une seule fois et c’est la version compilée qui est distribuée.

Les différentes technologies Java

- La plate-forme Java est distribuée sous plusieurs formes:
    - Java Platform Standart Edition (Java SE ou J2SE) contient les éléments de base de Java, destinée aux ordinateurs de bureau.
    - Java Platform Enterprise Edition (J2EE) reprend les éléments de J2SE plus des éléments orientés entreprise (technologies web, API de communication évoluées,...)
    - Java Platform Micro Edition (J2ME) est spécialement développé pour les appareils mobiles (PDA, téléphones mobiles,...). Plusieurs J2ME coexistent selon la “famille” de périphériques concernés (téléphone mobile, PDA, ...)

Quelques applications écrites en Java

- Eclipse : environnement de développement pluri-langages (Java, C/C++, ...)
- Vuze (ex-Azureus) : client BitTorrent
- GWT (Google Web Toolkit) : *framework* de développement AJAX
- TuxGuitar : éditeur de partitions musicales pour les guitaristes
- De nombreuses applications utilisent Java (sans forcement être écrite en Java) :
  - OpenOffice
  - Matlab,...

Outils d'exécution et de développement

- Pour exécuter un programme Java, une machine virtuelle doit être présente. Sun la fournit avec le JRE (Java Runtime Environment) qui doit être choisi en fonction de son système.

- Le JRE est installé sur la majorité des systèmes d'exploitation en standard (Windows, Linux, OS X, Solaris ...).

- Pour programmer en Java, il faut installer le JDK (Java Developement Kit, qui contient le JRE) adapté à sa machine distribué gratuitement par Sun. Plusieurs applications sont alors installées :

  - `javac` le compilateur qui transforme un fichier texte (le programme source) en un fichier compilé.

  - `javadoc` qui permet de construire automatiquement les documentations.

  - `jar` construit une archive "exécutable" des programmes comportant plusieurs fichiers.

Premier programme en Java

Fichier PremierProgramme.java

```
import java.util.Scanner;

public class PremierProgramme {

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String nom;
        System.out.println("Donnez votre nom ?");
        nom=sc.nextLine();
        System.out.println("Bonjour "+nom);
    }
}
```

Les imports permettent d'ajouter de nouvelles fonctionnalités aux programmes (ici la lecture des entrées)

Un fichier doit toujours contenir une classe de même nom.

Un programme doit avoir une méthode `main` pour être exécuté.

Ce programme demande le nom de l'utilisateur et affiche "Bonjour "suivie du nom.

- Les instructions sont terminées par un point-virgule.
- Les blocs d'instructions sont matérialisés par des accolades.
- Le langage est sensible à la case (majuscules, minuscules).

Premier programme en Java

- Pour compiler un fichier source il faut utiliser l'utilitaire `javac` en passant en paramètre le fichier source avec l'extension `.java`
- Après la compilation, un fichier `.class` est crée. C'est ce fichier qui peut être redistribué sur d'autres ordinateurs.
- Ce fichier peut être exécuté en utilisant l'utilitaire `java` en ligne de commande ; la majorité des interfaces graphique automatisent cette tâche.

```
new-host:Cours jb$ javac PremierProgramme.java
new-host:Cours jb$ ls
PremierProgramme.java    PremierProgramme.class
new-host:Cours jb$ java PremierProgramme
Donnez votre nom ?
Jb
Bonjour Jb
new-host:Cours jb$
```

Un programme est compilé avec la commande `javac PremierProgramme.java`

Un fichier `PremierProgramme.class` est crée après la compilation (si tout c'est bien passé !)

Le programme est exécuté avec la commande `java PremierProgramme` (on ne met pas l'extension `.class`).

Types de base

- Java est un langage typé : chaque variable doit être associée à un type de donnée qui ne peut pas varier au cours du programme.

- Ces types ne sont pas des classes ; il existe des classes équivalentes si nécessaire (par exemple pour les collections).

Type	Description
`boolean`	Booléen valant `true` ou `false`
`char`	Caractère Unicode sur 16 bits
`byte`	Entier sur 8 bits signé
`short`	Entier sur 16 bits signé
`int`	Entier sur 32 bits signé
`long`	Entier sur 64 bits signé
`float`	Flottant IEEE 754 sur 32 bits
`double`	Flottant IEEE 754 sur 64 bits

Déclaration de variables

- Les variables sont déclarées comme en C : un type suivie d'un nom (la ligne se terminant par un point virgule).

- Les déclarations peuvent se faire n'importe où dans le programme, mais avant la première utilisation. Il est conseillé de déclarer toutes les variables en début de bloc.

  - Pour les types, les variables **contiennent** la valeur (caractères, booléen, entier, ...).

  - Les variables peuvent aussi **référencer** des objets : un nom de classe suivi du nom de la variable.

- Que ce soit pour les types ou les objets, on peut déclarer plusieurs variables sur une même ligne (en les séparant par des virgules).

```
int a;
char b;
String uneChaine;
float x,y,z;
```

Initialisation des variables

- L’affectation de valeur se fait comme en C avec l’opérateur =
- Les valeurs peuvent être initialisée lors de la déclaration.
- L’initialisation peut contenir des opérations quelconques (arithmétiques, ...)
- Une variable doit être initialisée avant d’être utilisée en lecture sinon le compilateur provoquera une erreur.
- Dans 2 bloc imbriqués il ne pas avoir deux variables ayant le même nom;

```
{
    int a;
    {
        int a; // Provoque une erreur de compilation
    }   ...
}
```

Ecritures des différents types

- Les entiers (`byte`, `short`, `int`, `long`) en base 10, sont écrit de manière “naturelle”. Pour les valeurs hexadécimales, on utilise le préfixe 0X (ou 0x).

- Les caractères (`char`) sont placés entre `''` (touche 4 du clavier). Le caractère \ permet d’écrire les saut de lignes (\n), les tabulations (\t),...

- Les nombres flottants (`float`, `double`) sont notés avec le caractère . comme séparateur décimal. Par défaut, les valeurs sont des doubles, elles doivent être suivies de F pour spécifier une valeur en simple précision.

- La notation scientifique est possible en utilisant e comme exposant.

```
int a,b;
a=100;
b=0xF00;
```

```
char A='A';
char retourChariot='\n';
```

```
float euro=6.55957F;
double c=3e+8;
```

Conversion entre les différents types

- Il est possible de convertir une variable en un autre type en plaçant le type de transformation entre parenthèse lors de l'affectation. Cette opération porte le nom de *transtypage* (ou *casting*).

- Une conversion flottant vers entiers entraîne un arrondi.

- Dans le cas d'un entier vers un caractère, on utilise le point Unicode.

- Certaines conversions ne sont pas nécessaires, elles sont implicites dans le langage (par exemple d'un type `byte` vers un `int`).

- Il est conseiller de **toujours** forcer la conversion pour éviter les "mauvaises surprises".

```
double euro = 6.55957;
int euroArrondi;
euroArrondi = (int) euro;
System.out.println(euroArrondi);  // Affichage de 6
```

Principaux opérateurs

Priorité	Opérateur	Description
1	`++ --`	Incrémentation et décrémentation
1	`!`	Complément logique
1	`(type)`	Transtypage
2	`* / %`	Multiplication, division et modulo (reste de la division entière)
3	`+ -`	Addition et soustraction
4	`<< >>`	Décalage à gauche, à droite
5	`< <= => >`	Compariason
5	`instanceof`	Comparaison de type
6	`== !=`	Egalité et inégalité
7	`&`	ET binaire
8	`^`	OU exclusif binaire
9	`|`	OU binaire
10	`&&`	ET conditionnel
11	`||`	OU conditionnel
13	`=`	Affectation
13	`*= /= %= += -= <<= >>=`	Affectation avec opération

Les boucles `for`, `while` et `do...while`

- Les trois boucles usuelles existent en Java.

- La boucle `for` est utilisée quand le nombre d'itérations est connu et constant (parcours d'un tableau,...).

- Les deux autres sont utilisées pour répéter un bloc de programme en fonction d'une condition booléenne.

- La boucle `do...while` impose une exécution des instructions (contrairement à la boucle `while`) même si la condition est toujours fausse.

La boucle `for`

- Les instructions placées dans le bloc délimité par les accolades sont exécutées tant que le test de continuité est vrai.

- Une post incrémentation (ou décrémentation) est souvent utilisée pour modifier le compteur de boucle.

- Logiquement si le compteur est incrémenté, le test est inférieur (ou égal). Réciproquement, si le compteur est décrémenté, le test est supérieur (ou égal).

- Il est possible d'effectuer une déclaration et l'initialisation en même temps.

```
for (int i=0; i<10; i++) {
  System.out.println("Bonjour !");
}
```

La boucle `do...while`

- Les instructions placées dans le bloc délimité par les accolades sont exécutées tant que la condition de continuité est vraie.

- Cette condition peut être :
    - soit un test,
    - soit la valeur de retour d'un méthode (qui renvoie alors un booléen).

- Les instructions placées dans le bloc sont exécutées au moins une fois.

```
int a=0;
Random ran = new Random();

do{
  a  = ran.nextInt();
}while ( (a%2) != 0 );
```

La boucle `while`

- Les instructions placées dans le bloc délimité par les accolades sont exécutées tant que la condition de continuité est vraie.

- Cette condition peut être :
    - soit un test,
    - soit la valeur de retour d'une méthode (qui renvoie alors un booléen).

- Il est possible de trouver une affectation en même temps que le test dans la condition

```
Scanner sc = new Scanner(new File("Truc.txt"));

while (sc.hasNext()) {
  System.out.println(sc.nextLine());
}
```

Les tests

- Les structures `if` et `if...else` permettent d'exécuter un bloc d'instructions en fonction d'une condition.

- Cette condition peut être :

    - soit un test,
    - soit la valeur de retour d'une méthode (qui renvoie alors un booléen).

- Les structures `if...else` peuvent être imbriquées les unes dans les autres ; attention à l'écriture du programme dans ce cas.

```
if (condition) {
    // Instruction a exécuter si la condition est vraie
} else {
    // Instruction a exécuter si la condition est fausse
}
```

Les tests

- Les instructions placées dans un bloc toujours faux conduisent à une erreur de compilation.

- Les opérateurs && et || peuvent être utilisés comme des opérateurs court-circuits dans les tests entre deux opérations :
  - Si l'opération située à gauche de l'opérateur && est fausse (ou renvoie `false`), l'opération de droite n'est pas évaluée.
  - Si l'opération située à gauche de l'opérateur || est vraie (ou renvoie `true`), l'opération de droite n'est pas évaluée.

- Comme de nombreux langages, Java supporte l'opérateur ternaire `?:` qui est toutefois peu lisible.

```
max=(a>b)?a:b;
```

Les structures conditionnelles

- La structure `switch...case` permet d'obtenir un (ou plusieurs) choix parmi plusieurs selon une valeur entière. La variable est placée dans les parenthèses suivant l'instruction `switch`. Le cas correspondant à la valeur est traité, sinon c'est le cas `default` qui est traité.

- L'instruction `break` permet de quitter le bloc sélectionné, les cas suivants ne sont alors pas traités.

```
switch (variable) {
    case valeur1:
       // Instructions à traiter dans ce cas
    break;
    case valeur2:
        // Instructions à traiter dans ce cas
    break;
    default:
        // Instructions à traiter dans tous les autres cas
    break;
}
```

Les tableaux

- Un tableau doit d'abord être déclaré (comme une variable). La déclaration se compose d'un type (ou d'une classe), d'autant de paires de crochets que de dimensions au tableau et du nom du tableau.

- Le tableau est crée en utilisant l'instruction new suivie du type du tableau et des dimensions entre crochets.

- Les dimensions peuvent être des constantes ou des variables.

- La déclaration et la création peuvent être regroupées sur une seule ligne.

```
int[] tab;
tab = new int[100];
boolean[][] autre;
autre = new boolean[10][20];
```

```
int taille = 50;
int[] tab = new int[taille];
boolean[][] autre = new boolean[10][20];
```

Les tableaux

- Comme en C les éléments sont accessible individuellement à l'aide des crochets.

- Le premier élément a l'indice 0.

- Les tableaux sont des objets, ils ont une propriété (cette notion sera détaillée au prochain cours), `length` qui permet de connaître le nombre d'éléments stockés.

- Pour accéder à une propriété (que l'on utilise comme une variable), après le nom de la variable, on ajout un point puis le nom de la propriété.

```
for (int i = 0; i < tab.length; i++) {
    System.out.println("La case "+i+" du tableau contient : "+tab[i]);
}
```

Les boucles dans les tableaux

- Pour le parcours des tableau une notation simplifiée de `for` est possible (sans aucune conséquence sur la vitesse d'exécution).

- Le mot clé `for` est suivi d'une paire de parenthèses contenant la déclaration d'une variable (type puis nom) et du nom du tableau précédé de deux points.

- A chaque itération de la boucle, la variable prend une valeur possible dans le tableau, la boucle s'arrête au dernier élément du tableau.

- Cette notation ne permet pas d'accéder à l'indice de l'élément.

```
int[] tab;
int somme = 0;

tab = new int[10];
// Remplissage du tableau ...
for (int elt : tab) {
    somme += elt;
}

System.out.println("La somme des éléments vaut : " + somme);
```

Les tableaux : exemple

- Un exemple simple de manipulation de tableau est le calcul de la suite de Fibonacci.

```
import java.util.Scanner;

public class Fibonnacci {
    public static void main(String[] args) {
        int[] tab;
        int taille;
        Scanner sc = new Scanner(System.in);
        System.out.println("Combien de terme voulez-vous ?");
        taille=sc.nextInt();
        tab = new int[taille];
        tab[0]=0;
        tab[1]=1;
        for (int i = 2; i < tab.length; i++) {
            tab[i] = tab[i-1] + tab[i-2];
            System.out.println("Le terme "+i+" de la suite de
Fibonacci est : "+tab[i]);
        }
    }
}
```

Déclaration d'un tableau contenant des entiers.

Demande d'un entier à l'utilisateur et stockage dans `taille`.

Construction du tableau contenant le nombre d'élément demandé.

Remplissage des 2 premiers éléments.

Utilisation de la propriété `lenght` pour connaître la taille du tableau.

Boucle sur tout le tableau pour calculer et afficher les éléments.

Les chaînes de caractères

- La manipulation des chaînes de caractères est assurée par plusieurs classes en Java.

- La classe `String` est la plus utilisée pour manipuler les chaînes de caractères. Pour déclarer une chaîne de caractères, on doit donc utiliser `String` suivi du nom de la variable.

- Cette classe étant très utilisée, quelques simplifications d'écriture ("*sucre syntaxique*") sont présentes dans le langages afin d'en simplifier l'usage.

- En Java, les chaînes de caractères s'écrivent entre guillemets. Dans ce cas (et **exclusivement** dans ce cas) on n'utilise pas le constructeur.

- Les `String` sont proches des tableaux de caractères du C.

Les chaînes de caractères

- Les chaînes de caractères peuvent être déclarées et initialisées en plusieurs lignes ou en une seule ligne.

- La classe `String` a un comportement unique très différent des autres classes de Java pour des raisons de sécurité.

- A chaque fois qu'une chaîne est modifiée dans le programme, un nouvel objet est crée et associé à la variable. Les chaînes sont **immuables**. Ce comportement est **unique** aux chaînes.

```
String uneChaine;
uneChaine = "Bonjour !";
```

```
String uneChaine = "Bonjour !";
```

```
public static void main(String[] args) {
        String uneChaine;
        uneChaine = "Bonjour !";
        System.out.println("Adresse de la chaine " + Integer.toHexString(uneChaine.hashCode()) );
        uneChaine = "Au revoir !";
        System.out.println("Adresse de la chaine " + Integer.toHexString(uneChaine.hashCode()) );
}
```

```
Adresse de la chaine ba694b04
Adresse de la chaine ecf4c762
```

Les chaînes de caractères

- Les chaînes de caractères supportent naturellement l'encodage Unicode, on utilise la notation \uxxxx avec xxxx le point de code (valeur Unicode). La même notation est possible pour les `char`.

- La concaténation de chaînes (mise "bout-à-bout") se fait avec l'opérateur + (il s'agit d'une simplification d'écriture évitant le recours à la méthode `concat(String str)` de la classe `String`).

- Les types primitifs sont automatiquement convertit en chaîne au moment de la concaténation. Lors de la concaténation d'objet, c'est la méthode `toString()` qui est appelée. Par un mécanisme d'*héritage*, il est possible de concaténer **tous** les objets avec des chaînes.

```
String symboleEuro="\u20AC";
float prix=1456.87;
System.out.println("Le prix est de "+prix+" "+symboleEuro);
```

Conversion des chaînes de caractères

- La conversion d'une chaîne en un type (entier, flottant, ...) est une opération usuelle : lecture d'une valeur saisie au clavier, dans un fichier texte, ...

- Des *méthodes statiques* permettent de réaliser cette opération. Pour l'instant, il vous suffit de connaître la notation "par coeur" sans la comprendre pour l'utiliser.

- Pour transformer une chaîne de caractères en une valeur entière on utilise la méthode statique `parseInt(String s)` de la classe `Integer`

- Les méthodes `parseFloat(String s)` et `parseDouble(String s)` des classes `Float` et `Double` permettent de convertir une chaîne en `float` ou en `double`.

```
String uneValeur="3e+8";
double c;
c = Double.parseDouble(uneValeur);
```

Méthodes utiles des chaînes de caractères

- La classe `String` fournit de très nombreuses méthodes pour faciliter le travail du programmeur.

- Une méthode peut être assimilée à une fonction qui s'applique à un objet. Pour appeler une méthode on ajoute un point au nom de l'objet auquel on veut appliquer la méthode puis le nom de la méthode suivit des éventuels paramètres entre parenthèses (s'il n'y a pas de paramètre on place une paire de parenthèses vides).

- La méthode `indexOf(String str)` permet de rechercher une chaîne de caractères dans une autre. Elle renvoie la position de la première occurrence de la chaîne recherchée.

```
String chaine, motif;
int position;
chaine = "J'aime Java";
motif = "Java";
position = chaine.indexOf(motif);
System.out.println("Le mot recherché est situé en " + position);
```

Méthodes utiles des chaînes de caractères

- Les méthodes `toLowerCase()` et `toUpperCase()` permettent de convertir une chaîne en minuscules/majuscules.

- La méthode `split(String regex)` permet de découper une chaîne en fonction d'un caractère (ou d'une expression régulière) passé en paramètre. Elle renvoie un tableau de chaînes de caractères.

```
public static void main(String[] args){
    String chaine="Bill,Bob,Steve,Mick";
    String[] prenoms;
    prenoms = chaine.split(",");
    for (int i = 0; i < prenoms.length; i++) {
        System.out.println(prenoms[i]);
    }
}
```

Déclaration et initialisation d'une chaîne contenant des noms séparés par des virgules.

Déclaration d'un tableau de `String`.

Utilisation de la méthode split sur la `chaine` avec " , " comme argument.

Parcours du tableau et affichage de chaque élément.

Les chaînes de caractères : exemple

- Le programme suivant demande une chaîne de caractères à l'utilisateur et l'affiche en majuscules.

```
import java.util.Scanner;

public class Majuscules {
    public static void main(String[] args) {
        String uneChaine, uneAutreChaine;
        Scanner sc = new Scanner(System.in);
        System.out.println("Donnez un mot ?");
        uneChaine=sc.nextLine();
        uneAutreChaine=uneChaine.toUpperCase();
        System.out.println("Votre chaine en majuscule :
"+uneAutreChaine);
    }
}
```

Déclaration de 2 chaînes de caractères.

Déclaration et création d'un objet Scanner pour lire le clavier

Demande d'une chaîne à l'utilisateur et stockage dans uneChaine grâce à la méthode nextLine() de la classe Scanner.

Utilisation de la méthode toUpperCase() pour construire une nouvelle chaine placée dans uneAutreChaine.

Affichage de la chaîne uneAutreChaine.