## Exercice 1 : Passage de l'iJava au Java

Voici le code d'un programme iJava tel que vous auriez pu l'écrire.

```
1     void algorithm() {
          final char SEP = ';';
3         final char SUB = ' ';
          String[] lignesCSV = new String[] {
5             "2-207-3040-3;Arthur;Dent;42", "2-207-30351-9;Ford;Prefect;224",
              "2-207-30369-1;Zaphod;Beeblebrox;224", "2-207-30549-X;Marvin;Trilian;256"
7         };
          for(int i=0; i<length(lignesCSV); i=i+1) {
9             int deb=-1, fin=-1, idx=0;
              String maChaîne = lignesCSV[i];
11            while(idx<length(maChaîne) && charAt(maChaîne, idx)!=SEP) idx=idx+1;
              deb = idx;
13            idx=length(maChaîne)-1;
              while(idx>=0 && charAt(maChaîne, idx )!=SEP) idx=idx-1;
15            fin = idx;
              String aTraiter = substring(maChaîne, deb, fin);
17            String res = "";
              for(idx=0; idx<length(aTraiter); idx=idx+1)
19                if(charAt(aTraiter,idx)==SEP) res=res+SUB;
                  else res=res+charAt(aTraiter,idx);
21            println("Bonjour" + res);
          }
23    }
}
```

**Q1.** Analysez ce code pour comprendre sa fonction. Qu'affiche-t-il ?

**Q2.** À l'aide de la documentation de la classe `String` fournie, réécrivez ce code en Java (à partir de la ligne 8).

**Class `String`**

| | |
|---|---|
| `char` | `charAt(int index)`<br>Returns the char value at the specified index. |
| `int` | `compareTo(String anotherString)`<br>Compares two strings lexicographically. |
| `int` | `indexOf(char ch)`<br>Returns the index within this string of the first occurrence of the specified character. |
| `boolean` | `isEmpty()`<br>Returns `true` if, and only if, `length()` is 0. |
| `int` | `lastIndexOf(char ch)`<br>Returns the index within this string of the last occurrence of the specified character. |
| `int` | `length()`<br>Returns the length of this string. |
| `String` | `replace(char oldChar, char newChar)`<br>Returns a string resulting from replacing all occurrences of `oldChar` in this string with `newChar`. |
| `String` | `substring(int beginIndex, int endIndex)`<br>Returns a string that is a substring of this string. |

## Exercice 2 : Écriture d'une première classe

On souhaite disposer d'un objet `NetworkIP`, permettant de représenter les adresses réseau en sachant qu'elles sont composées de 4 nombres entre 0 et 255 séparés par des points à l'affichage. Les deux premiers représentent le domaine (`dom1` et `dom2` par exemple) et les deux suivants représentent l'hôte (`host1` et `host2` par exemple).

**Q1.** Écrire une classe `NetworkIP` répondant aux spécifications suivantes :

**Class `NetworkIP`**

| | |
|---|---|
| | `NetworkIP(int a, int b, int c, int d)`<br>Crée une adresse IP initialisée avec les nombres fournis. |
| `String` | `strDomaine()`<br>Retourne l'adresse IP sous forme : `xx.xx`. |
| `String` | `strHost()`<br>Retourne l'adresse IP sous forme international : `xx.xx`. |

**Q2.** Donnez la représentation UML de cette classe.

**Q3.** Que faut-il ajouter pour que l'instruction suivante affiche l'adresse réseau complète ?

```
System.out.println(new NetworkIP(127, 0, 0, 1));
```

**Q4.** Que faudrait-il modifier à votre classe si l'on désirait stocker les différents nombres dans un tableau ?

**Q5.** En utilisant la classe `Random` du package `java.util`, écrire une classe `UseNetworkIP`, munie uniquement d'une méthode principale (`main`), qui génère aléatoirement 5 adresses et les affichent.

```
244.228.232.130
184.239.243.237
176.17.149.172
24.75.87.105
154.201.54.213
```

**Class `Random`**

| | |
|---|---|
| | `Random()`<br>Creates a new random number generator. |
| `int` | `nextInt()`<br>Returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence. |
| `int` | `nextInt(int bound)`<br>Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence. |