Corrigé des exercices de conversion de types

Exercice 7.1.1 *conversion de String vers int*

Ecrire une méthode appelée `stringToInt` qui réalise la conversion d'une chaîne de caractère de la même façon que la méthode prédéfinie `Integer.parseInt`, c'est-à-dire qui renvoie la valeur numérique de la chaîne si celle-ci représente un nombre entier et qui provoque une erreur dans les autres cas (par exemple si la chaîne contient des lettres). Vous vous assurerez que la méthode fonctionne également pour des nombres négatifs.

```
public class ExoCT1{
    public static int stringToInt(String s){
        int res = 0;
        int deb = 0;
        boolean positif = true;
        char c;
        if (s.charAt(0) == '-'){
            positif = false;
            deb = 1;
        }
        for (int i=deb; i<s.length(); i++){
            c = s.charAt(i);
            if (!Character.isDigit(c)){
                throw new Error("La chaine n'est pas un nombre entier");
            }
            res = res *10 + Character.getNumericValue(c);
        }
        if (positif){
            return res;
        }else{
            return -res;
        }
    }
    public static void main(String[] args){
        Terminal.ecrireString("Entrez une chaine a convertir: ");
        Terminal.ecrireIntln(stringToInt(Terminal.lireString()));
    }
}
```

Exercice 7.1.2 *extraction de mots*

On va considérer qu'un mot est une suite de caractères qui ne comprend que des lettres. Tous les autres caractères seront considérés comme des séparateurs utilisés pour séparer les différents mots. Pour déterminer si un caractère est une lettre, on utilisera la méthode `Character.isLetter`.

Ecrire une méthode qui transforme une chaîne de caractères en un tableau contenant les mots de cette chaîne dans le même ordre.

```
public class ExoCT2{
    public static String[] stringToMots(String s){
        String[] temp = new String[100];
        int nextSt = 0;
        String current = "";
        boolean motEnCours = false;
        if (Character.isLetter(s.charAt(0))){
            motEnCours = true;
            current = current + s.charAt(0);
        }
        for (int i=1; i<s.length(); i++){
            if (Character.isLetter(s.charAt(i))){
                if (motEnCours){ // rajout d'une lettre de plus
                    current = current + s.charAt(i);
                } else{ // debut d'un nouveau mot
                    current = "" + s.charAt(i);
                    motEnCours = true;
                }
            }else{
                if (motEnCours){ // fin d'un mot
                    temp[nextSt] = current;
                    nextSt++;
                    motEnCours = false;
                }
                // sinon on ne fait rien
            }
        }
        if (motEnCours){
            temp[nextSt] = current;
            nextSt++;
        }
        // construction d'un tableau de la bonne taille
        String[] res = new String[nextSt];
        for (int i = 0; i<nextSt; i++){
            res[i] = temp[i];
        }
        return res;
    }
    public static void main(String[] args){
        String[] lesMots;
        Terminal.ecrireString("Entrez␣une␣chaine␣a␣convertir:␣");
        lesMots = stringToMots(Terminal.lireString());
        for (int i=0; i< lesMots.length; i++){
            Terminal.ecrireStringln("Mot␣numero␣" + i + "␣:␣" + lesMots[i]);
        }
```

```
    }
}
```

Exercice 7.1.3 *conversion de chiffres romains*

Les nombres romains sont des nombres écrits au moyen de lettres qui peuvent être considérées comme les chiffres de ce mode de notation. Chaque lettre est associée à un nombre qui s'ajoute ou se retranche selon la position du chiffre dans le nombre.

Question 1 *conversion de chiffre*

Ecrire une méthode qui convertit un chiffre romain donné au moyen d'un caractère du type `char` vers la valeur nuérique correspondante (type `int`). Dans le cas où le caractère n'est pas une des sept lettres utilisée par les nombres romain, la méthode doit provoquer une erreur.

Question 2 *conversion de nombre*

Ecrire une méthode qui convertit un nombre romain écrit dans une String en une valeur numérique de type `int`. Il sera utile de faire appel à la méthode écrite à la question précédente.

```
public class ExoCT3{
    public static int chiffreToInt(char c){
        int res;
        if (c == 'I'){
            res = 1;
        }else if (c == 'V'){
            res = 5;
        }else if (c == 'X'){
            res = 10;
        }else if (c == 'L'){
            res = 50;
        }else if (c == 'C'){
            res = 100;
        }else if (c == 'D'){
            res = 500;
        }else if (c == 'M'){
            res = 1000;
        }else{
            throw new Error("ce n'est pas un chiffre romain: " + c);
        }
        return res;
    }
    public static int nombreToInt(String s){
        int res = chiffreToInt(s.charAt(s.length()-1));
        int valprec = res;
        int valcour;
        for (int i=s.length()-2; i>=0; i=i-1){
            valcour = chiffreToInt(s.charAt(i));
            if (valcour >= valprec){
                res = res + valcour;
```

```
            }else{
                res = res -valcour;
            }
            valprec = valcour;
        }
        return res;
    }
    public static void main(String[] args){
        Terminal.ecrireString("Entrez␣un␣nombre␣a␣convertir:␣");
        Terminal.ecrireIntln(nombreToInt(Terminal.lireString()));
    }
}
```

Exercice 7.1.4 *cryptographie élémentaire*

Question 1 *code de Jules César*

Pour crypter ses messages, Jules César a utilisé un code qui consistait à remplacer chaque lettre par la lettre située trois crans plus loin dans l'alphabet. Par exemple, le a est codé par un d, le b par un e, etc. Les trois dernières lettres de l'alphabet sont codées respectivement par a, b et c.

Ecrivez les fonctions d'encodage et de décodage pour ce code. Ces fonctions ne seront définies que pour les lettres, les autres caractères restant inchangés. Ces fonctions permettront d'encoder et de décoder des chaînes de caractère.

Notez qu'en Java, le type `char` est numérique : on peut lui appliquer les opération arithmétiques et notamment l'addition et la soustraction utiles dans cet exercice.

```
class Exo12_2{
    static String cesarEncode(String s){
        String res = "";
        for (int i=0; i<s.length(); i++){
            res = res + charCesarEncode(s.charAt(i));
        }
        return res;
    }
    static String cesarDecode(String s){
        String res = "";
        for (int i=0; i<s.length(); i++){
            res = res + charCesarDecode(s.charAt(i));
        }
        return res;
    }
    static char charCesarDecode(char c){
        if ((c>= 'A') && (c<='C')){
            return (char) (c+ 23);
        }
        if ((c>= 'a') && (c<='c')){
            return (char) (c+23);
        }
        if ((c>= 'd') && (c<='z')){
            return (char) (c-3);
        }
```

```
        if ((c>= 'D') && (c<='Z')){
            return (char) (c-3);
        }
        return c;
    }
    static char charCesarEncode(char c){
        if ((c>= 'A') && (c<='W')){
            return (char) (c+ 3);
        }
        if ((c>= 'a') && (c<='w')){
            return (char) (c+3);
        }
        if ((c>= 'x') && (c<='z')){
            return (char) (c-23);
        }
        if ((c>= 'X') && (c<='Z')){
            return (char) (c-23);
        }
        return c;
    }
    public static void main(String[] args){
        String st = "abcdwxyz";
        Terminal.ecrireStringln(st + "␣=>␣" + cesarEncode(st));
        Terminal.ecrireStringln(cesarDecode(cesarEncode(st)));
   }
}
```

Question 2 *un code à clé*

On veut garder le principe de remplacement d'une lettre par une autre caractérisée par son décalage dans l'alphabet, mais on veut maintenant que ce décalage varie d'une lettre à l'autre selon une clé numérique. Chaque chiffre de la clé donne le décalage d'une lettre.

Réponse : pour ce codage, on peut adapter le principe de notre programme précédent ou préférer passer par un calcul sur les entiers représentant l'ordre des lettres dans l'alphabet. Dans les deux cas, il faut utiliser l'opérateur modulo (%) qui calcule le reste de la division entière.

```
class Exo12_2_2{
    static String aCleEncode(String s, int[] cle){
        String res = "";
        for (int i=0; i<s.length(); i++){
            res = res + charACleEncode(s.charAt(i),cle[i % cle.length]);
        }
        return res;
    }
    static String aCleDecode(String s, int[] cle){
        String res = "";
        for (int i=0; i<s.length(); i++){
            res = res + charACleDecode(s.charAt(i),cle[i % cle.length]);
        }
        return res;
```

```
    }
    static char charACleDecode(char c, int n){
        if ((c>= 'A') && (c<(char) 'A'+n)){
            return (char) (c+ 26 -n);
        }
        if ((c>= 'a') && (c<(char) 'a'+n)){
            return (char) (c+ 26 -n);
        }
        if ((c>=(char) 'A'+n) && (c<='z')){
            return (char) (c-n);
        }
        if ((c>=(char) 'a'+n) && (c<='Z')){
            return (char) (c-n);
        }
        return c;
    }
    static char charACleEncode(char c, int n){
        if ((c>= 'A') && (c<=(char) 'Z'-n)){
            return (char) (c+ n);
        }
        if ((c>= 'a') && (c<=(char) 'z'-n)){
            return (char) (c+n);
        }
        if ((c>(char) 'z'-n) && (c<='z')){
            return (char) (c-26+n);
        }
        if ((c> (char) 'Z'-n) && (c<='Z')){
            return (char) (c-26+n);
        }
        return c;
    }
    public static void main(String[] args){
        String st = "bonjourcommentva";
        int[] cle = {2,3,7,8,1};
        Terminal.ecrireStringln(st + "␣=>␣" + aCleEncode(st,cle));
        st = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        Terminal.ecrireStringln(st + "␣=>␣" + aCleEncode(st,cle));
        Terminal.ecrireStringln("237812378123781237812378123781237812378123781");
        Terminal.ecrireStringln(aCleDecode(aCleEncode(st,cle),cle));
   }
}
```

Voici la version avec calcul sur les entiers plutôt que sur les caractères :

```
class Exo12_2_2bis{
    static String aCleEncode(String s, int[] cle){
        String res = "";
        for (int i=0; i<s.length(); i++){
            res = res + charACleEncode(s.charAt(i),cle[i % cle.length]);
        }
        return res;
    }
    static String aCleDecode(String s, int[] cle){
```

```
        String res = "";
        for (int i=0; i<s.length(); i++){
            res = res + charACleDecode(s.charAt(i),cle[i % cle.length]);
        }
        return res;
    }
    static char charACleDecode(char c, int n){
        int x, diff;
        if ((c>= 'A') && (c<= 'Z')){
            x = c -'A';
            diff = 'A';
        }else if ((c>= 'a') && (c<= 'z')){
            x = c -'a';
            diff = 'a';
        }else{
            return c;
        }
        return (char) (((x+26-n) % 26) + diff);
    }
    static char charACleEncode(char c, int n){
        int x, diff;
        if ((c>= 'A') && (c<= 'Z')){
            x = c -'A';
            diff = 'A';
        }else if ((c>= 'a') && (c<= 'z')){
            x = c -'a';
            diff = 'a';
        }else{
            return c;
        }
        return (char) (((x+n) % 26) + diff);
    }
    public static void main(String[] args){
        String st = "bonjourcommentva";
        int[] cle = {2,3,7,8,1};
        Terminal.ecrireStringln(st + "␣=>␣" + aCleEncode(st,cle));
        st = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        Terminal.ecrireStringln(st + "␣=>␣" + aCleEncode(st,cle));
        Terminal.ecrireStringln("2378123781237812378123781237812378123781");
        Terminal.ecrireStringln(aCleDecode(aCleEncode(st,cle),cle));
    }
}
```