

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Amir
 */
import java.io.*;

/**
 * Le principal but de la classe Clavier est de proposer des
 * entrées clavier sans exception. On propose ainsi
 * une opération élémentaire de lecture sur l'entrée standard. Cette
 * classe permet d'éviter de rentrer dans les détails complexes des
 * entrées-sorties.
 */

public class Clavier {
    /**
     * Cette méthode lit un entier dans une chaîne de caractères.
     * Elle s'occupe des éventuelles exceptions. Si une exception
     * (NumberFormatException) est attrapée, la méthode renvoie l'entier
     * nul et affiche un message d'erreur.
     * @param s la chaîne de caractères dans laquelle on lit
     * @return l'entier lu (0 en cas d'erreur) */
    public static int toInt(String s) {
        int result=0;
        try {
            result=Integer.valueOf(s).intValue();
        } catch (NumberFormatException e) {
            System.err.println("Erreur de format: "+e.getMessage());
        }
        return result;
    }

    /**
     * Cette méthode lit un réel (double) dans une chaîne de caractères.
     * Elle s'occupe des éventuelles exceptions. Si une exception
     * (NumberFormatException) est attrapée, la méthode renvoie le réel
     * nul et affiche un message d'erreur.
     * @param s la chaîne de caractères dans laquelle on lit
     * @return le réel lu (0 en cas d'erreur)
     */
    public static double toDouble(String s) {
        double result=0;
        try {
            result=Double.valueOf(s).doubleValue();
        } catch (NumberFormatException e) {
            System.err.println("Erreur de format: "+e.getMessage());
        }
        return result;
    }

    public static float toFloat(String s) {
        float result=0;
        try {
            result=Float.valueOf(s).floatValue();
        } catch (NumberFormatException e) {
            System.err.println("Erreur de format: "+e.getMessage());
        }
        return result;
    }

    /**
     * Cette méthode lit une ligne sur l'entrée standard et s'occupe des
     * éventuelles exceptions. Si une exception (IOException) est
     * attrapée, la méthode renvoie une chaîne vide et affiche un
     * message d'erreur.
     * @return la chaîne lue (vide en cas d'erreur)
     */
}
```

```
public static String lireString() {
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    String result;
    try {
        result=br.readLine();
    } catch (IOException e) {
        System.err.println("Erreur de lecture: "+e.getMessage());
        result="";
    }
    return result;
}

/**
 * Cette méthode lit un entier sur l'entrée standard.
 * Elle s'occupe des éventuelles exceptions et utilise la méthode
 * toInt pour obtenir un entier. S'il y a un problème de lecture,
 * la méthode affiche un message d'erreur et renvoie 0.
 * @return l'entier lu (0 en cas d'erreur)
 */
public static int lireInt() {
    return toInt(lireString());
}

/**
 * Cette méthode lit un réel sur l'entrée standard.
 * Elle s'occupe des éventuelles exceptions et utilise la méthode
 * toDouble pour obtenir un réel. S'il y a un problème de lecture,
 * la méthode affiche un message d'erreur et renvoie 0.
 * @return le réel lu (0 en cas d'erreur)
 */
public static double lireDouble() {
    return toDouble(lireString());
}
public static float lireFloat() {
    return toFloat(lireString());
}

/**
 * Cette méthode lit un booléen sur l'entrée standard.
 * Elle s'occupe des éventuelles exceptions et renvoie false
 * s'il y a un problème de lecture.
 * @return le booléen lu (false en cas d'erreur)
 */
public static boolean lireBoolean() {
    return Boolean.valueOf(lireString()).booleanValue();
}

/**
 * Cette méthode lit un caractère sur l'entrée standard.
 * Elle s'occupe des éventuelles exceptions et renvoie NULL
 * s'il y a un problème de lecture.
 * N.B.: si plusieurs caractères sont entrés, c'est le premier
 * qui est pris en compte.
 * @return le (premier) caractère lu (caractère spécial NULL en cas d'erreur)
 */
public static char lireChar() {
    char result='\u0000';
    try {
        result=lireString().charAt(0);
    } catch (IndexOutOfBoundsException e) {
        System.err.println("Erreur de lecture: aucun caractère entré");
    }
    return result;
}
}
```