

LIRE LES ENTREES CLAVIER

Avant d'entamer les interfaces graphiques, nous travaillerons en mode console. Donc, afin de rendre nos programmes plus ludiques, il est de bon ton de pouvoir interagir avec ceux-ci. Après la lecture de ce chapitre, vous pourrez saisir des informations et les stocker dans des variables afin de pouvoir les utiliser a posteriori.

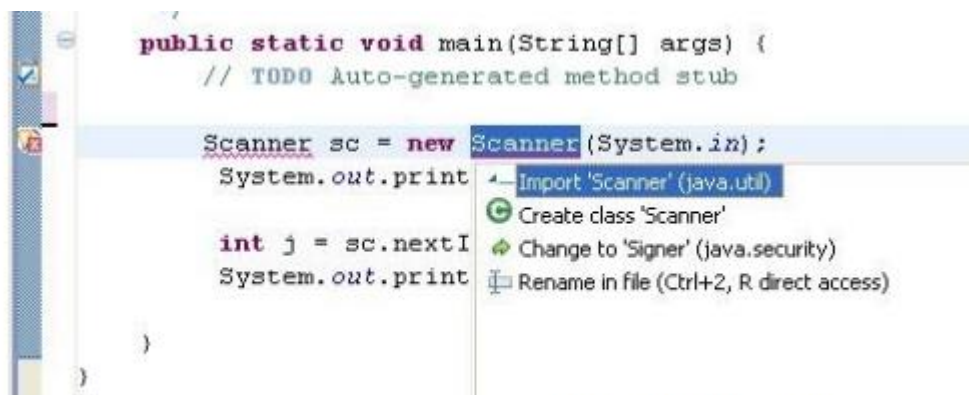
1. La classe Scanner

Nous avons vu que les variables de type *String* sont en réalité des objets de type *String*. Pour que Java puisse lire ce que vous tapez au clavier, vous allez devoir utiliser un objet de type *Scanner*. Cet objet peut prendre différents paramètres, mais ici nous n'en utiliserons qu'un : celui qui correspond à l'entrée standard (le clavier) en Java. Lorsque vous faites *System.out.println()*, je vous rappelle que vous appliquez la méthode *println()* sur la sortie standard (l'écran). Ainsi, nous allons utiliser l'entrée standard *System.in*, comme dans l'exemple suivante :

```
Scanner sc = new Scanner(System.in);
```

Java code

Vous devez avoir une jolie vague rouge sous le mot *Scanner*. Cliquez sur la croix rouge sur la gauche et faites un double-clic sur *Import 'Scanner' java.util* (figure suivante). Et là, l'erreur disparaît !



Importer la classe Scanner

Maintenant, regardez au-dessus de la déclaration de votre classe, vous devriez voir cette ligne :

```
import java.util.Scanner;
```

Java code

Voilà ce que nous avons fait. Je vous ai dit qu'il fallait indiquer à Java où se trouve la classe *Scanner*. Pour faire ceci, nous devons importer la classe *Scanner* grâce à l'instruction **import**. La classe que nous voulons se trouve dans le package `java.util`.

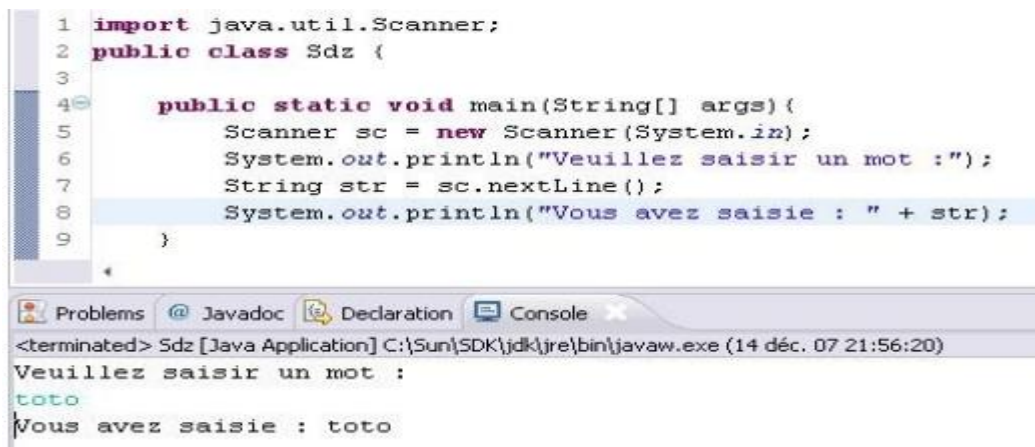
2. Récupérer ce que vous tapez

Voici l'instruction pour permettre à Java de récupérer ce que vous avez saisi pour ensuite l'afficher :

```
Scanner sc = new Scanner(System.in);  
System.out.println("Veuillez saisir un mot :");  
String str = sc.nextLine();  
System.out.println("Vous avez saisi : " + str);
```

Java code

Une fois l'application lancée, le message que vous avez écrit auparavant s'affiche dans la console, en bas d'Eclipse. Pensez à cliquer dans la console afin que ce que vous saisissez y soit écrit et que Java puisse récupérer ce que vous avez inscrit (figure suivante) !



```
1 import java.util.Scanner;  
2 public class Sdz {  
3  
4     public static void main(String[] args){  
5         Scanner sc = new Scanner(System.in);  
6         System.out.println("Veuillez saisir un mot :");  
7         String str = sc.nextLine();  
8         System.out.println("Vous avez saisie : " + str);  
9     }  
}
```

The screenshot also shows the Eclipse console output:

```
<terminated> Sdz [Java Application] C:\Sun\SDK\jdk\jre\bin\javaw.exe (14 déc. 07 21:56:20)  
Veuillez saisir un mot :  
toto  
Vous avez saisie : toto
```

En plus des chaînes de caractères on peut lire d'autre type de données :

```
Scanner sc = new Scanner(System.in);  
System.out.println("Veuillez saisir un mot :");  
int str = sc.nextInt();  
System.out.println("Vous avez saisi : " + str);
```

Java code

...vous devriez constater que lorsque vous introduisez votre variable de type Scanner (sc) et que vous introduisez le point permettant d'appeler des méthodes de l'objet, Eclipse vous propose une liste de méthodes associées à cet objet (ceci s'appelle l'autocomplétion) ; de plus, lorsque vous commencez à taper le début de la méthode nextInt(), le choix se restreint jusqu'à ne laisser que cette seule méthode.

Exécutez et testez ce programme : vous verrez qu'il fonctionne à la perfection. Sauf...si vous saisissez autre chose qu'un nombre entier !

Vous savez maintenant que pour lire un int, vous devez utiliser nextInt(). De façon générale, dites-vous que pour récupérer un type de variable, il vous suffit d'appeler next<Type de variable commençant par une majuscule>

```
Scanner sc = new Scanner(System.in);  
int i = sc.nextInt();  
double d = sc.nextDouble();  
long l = sc.nextLong();  
byte b = sc.nextByte();  
//Etc...
```

Java code

✚ Il y a un type de variables primitives qui n'est pas pris en compte par la classe Scanner : il s'agit du type **char**. Voici comment on pourrait récupérer un caractère :

```
System.out.println("Saisissez une lettre :");  
Scanner sc = new Scanner(System.in);  
String str = sc.nextLine(); //en gros on récupère un String(chaine de caractère)  
char caract = str.charAt(0); //et on accède au 1er caractère de la chaine récupérée  
/*une chaine de caractère est perçue comme un tableau de caractère. Et en java, le  
premier élément d'un tableau a pour indice : 0*/  
System.out.println("Vous avez saisi le caractère : " + caract);
```

Java code

✚ Une précision s'impose, toutefois : la méthode nextLine() récupère le contenu de toute la ligne saisie et remplace la « tête de lecture » au début d'une autre ligne. Par contre, si vous avez invoqué une méthode comme nextInt(), nextDouble() et que vous invoquez directement après la méthode nextLine(), celle-ci ne vous invitera pas à saisir une chaîne de caractères : elle videra la ligne commencée par les autres instructions. En effet, celles-ci ne repositionnent pas la tête de lecture, l'instruction nextLine() le fait à leur place. Pour faire simple, ceci :

```
Scanner sc = new Scanner(System.in);
System.out.println("Saisissez un entier : ");
int i = sc.nextInt();
System.out.println("Saisissez une chaîne : ");
String str = sc.nextLine();
System.out.println("FIN ! ");
```

Java code

...ne vous demandera pas de saisir une chaîne et affichera directement « Fin ». Pour pallier ce problème, il suffit de vider la ligne après les instructions ne le faisant pas automatiquement :

```
Scanner sc = new Scanner(System.in);
System.out.println("Saisissez un entier : ");
int i = sc.nextInt();
System.out.println("Saisissez une chaîne : ");
sc.nextLine(); //On vide la ligne avant d'en lire une autre
String str = sc.nextLine();
System.out.println("FIN ! ");
```

Java code