

Java Input



Lab 0C

© A+ Computer Science - www.apluscompsci.com

imports

© A+ Computer Science - www.apluscompsci.com

Scanner Imports

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

**Try to be as specific as possible
when using an import.**

© A+ Computer Science - www.apluscompsci.com

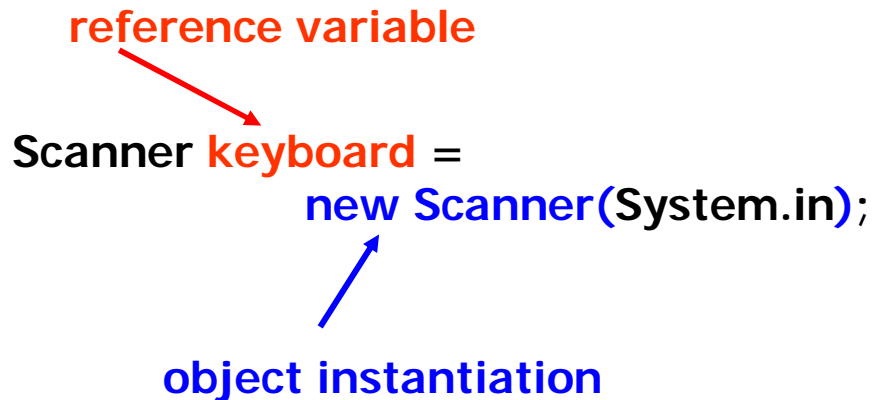
In order to use Scanner, you must import `java.util.Scanner`.

Scanner Creation

reference variable

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

object instantiation



© A+ Computer Science - www.apluscompsci.com

Scanner is a class which must be instantiated before it can be used. In other words, you must make a new Scanner if you want to use Scanner. A reference must be used to store the location in memory of the Scanner object created.

System.in is the parameter passed to the Scanner constructor so that Java will know to connect the new Scanner to the keyboard. keyboard is a reference that will store the location of newly created Scanner object.

Scanner Methods

© A+ Computer Science - www.apluscompsci.com

Scanner frequently used methods	
Name	Use
nextInt()	returns the next int value
nextDouble()	returns the next double value
nextFloat()	returns the next float value
nextLong()	returns the next long value
nextByte()	returns the next byte value
nextShort()	returns the next short value
next()	returns the next one word String
nextLine()	returns the next multi word String

`import java.util.Scanner;`

© A+ Computer Science - www.apluscompsci.com

This chart lists the Scanner methods that will be used most frequently. More Scanner methods will be introduced later.

Reading in Integers

```
Scanner keyboard =  
    new Scanner(System.in);  
  
out.print("Enter an integer :: ");  
int num = keyboard.nextInt();
```



© A+ Computer Science - www.apluscompsci.com

The `nextInt ()` method is used to tell a `Scanner` object to retrieve the next integer value entered.

In the example, the next integer typed in on the keyboard would be read in and placed in the integer variable `num`.

`nextInt ()` will read up to the first whitespace value entered.

Reading in Integers

```
out.print("Enter an integer :: ");  
int num = keyboard.nextInt();  
out.println(num);
```

INPUT

931

OUTPUT

Enter an integer :: 931

931



© A+ Computer Science - www.apluscompsci.com

The `nextInt ()` method is used to tell a `Scanner` object to retrieve the next integer value entered.

In the example, the next integer typed in on the keyboard would be read in and placed in the integer variable `num`.

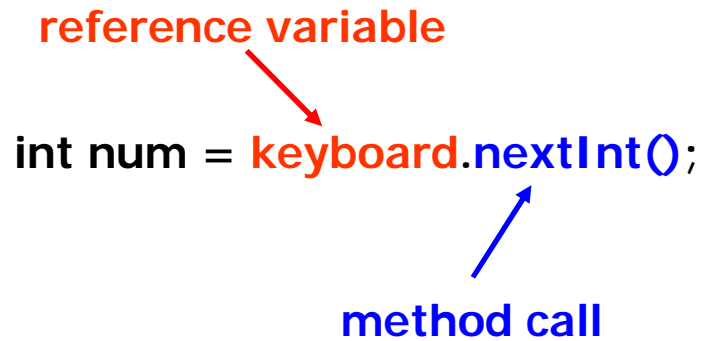
`nextInt ()` will read up to the first whitespace value entered.

Reading in Integers

reference variable

```
int num = keyboard.nextInt();
```

method call



© A+ Computer Science - www.apluscompsci.com

The `nextInt ()` method will read in the next integer. If a non-integer value is encountered such as a decimal value, the result will be run-time exception.

`keyboard` is a reference that refers to a `Scanner` object.

Reading in data

```
out.print("Enter an integer :: ");
```

Prompts are used to tell the user what you want.

© A+ Computer Science - www.apluscompsci.com

When performing input operations, it is a must to use prompts. A prompt is a way of indicating to a user what type of data to enter.

The prompt above indicates that an integer value is expected.

Open scannerints.java

© A+ Computer Science - www.apluscompsci.com

Reading in Doubles

```
Scanner keyboard =  
    new Scanner(System.in);  
  
out.print("Enter a double :: ");  
double num = keyboard.nextDouble();
```



© A+ Computer Science - www.apluscompsci.com

The `nextDouble()` method will read in the next numeric value entered. Any integer or decimal value will be accepted.

In the example, the next numeric value entered on the keyboard would be read in and placed in variable `num`.

`nextDouble()` will read up to the first whitespace value entered.

Reading in Doubles

```
out.print("Enter a double :: ");  
double num = keyboard.nextDouble();  
out.println(num);
```

INPUT

34.33

OUTPUT

Enter a double :: 34.33

34.33



© A+ Computer Science - www.apluscompsci.com

The `nextDouble()` method will read in the next numeric value entered. Any integer or decimal value will be accepted.

In the example, the next numeric value entered on the keyboard would be read in and placed in variable `num`.

`nextDouble()` will read up to the first whitespace value entered.

Reading in Doubles

reference variable

```
double num = keyboard.nextDouble();
```

method call

© A+ Computer Science - www.apluscompsci.com

The `nextDouble()` method will read in the next numeric value. If a non-numeric value is encountered such as a text value or word, the result will be run-time exception.

`keyboard` is a reference that refers to a `Scanner` object.

**Open
scannerreals.java**

© A+ Computer Science - www.apluscompsci.com

Reading in Strings

```
Scanner keyboard =  
    new Scanner(System.in);  
  
out.print("Enter a string :: ");  
String word = keyboard.next();
```

© A+ Computer Science - www.apluscompsci.com

The `next ()` method will read in the next text value entered. A numeric or non-numeric text value will be accepted.

In the example, the next text entered on the keyboard would be read in and placed in variable `word`.

The `next ()` method would read up to the first whitespace encountered. Whitespace would be any space, any tab, or any enter key.

Reading in Strings

```
out.print("Enter a string :: ");  
String word = keyboard.next();  
out.println(word);
```

INPUT

I love java.

OUTPUT

Enter a string :: I love java.

I

© A+ Computer Science - www.apluscompsci.com

The `next ()` method will read in the next text value entered. A numeric or non-numeric text value will be accepted.

In the example, the next text entered on the keyboard would be read in and placed in variable `word`.

The `next ()` method would read up to the first whitespace encountered. Whitespace would be any space, any tab, or any enter key.

Reading in Lines

```
Scanner keyboard =  
    new Scanner(System.in);  
  
out.print("Enter a sentence :: ");  
String sentence = keyboard.nextLine();
```



© A+ Computer Science - www.apluscompsci.com

The `nextLine()` method will read in an entire line of text including whitespace (enter keys, spaces, tabs, etc.). Any text value entered will be accepted, including a line containing spaces.

In the example, the next line of data entered on the keyboard would be read in and placed in variable `sentence`.

Reading in Lines

```
out.print("Enter a line :: ");  
String line = keyboard.nextLine();  
out.println(line);
```

INPUT

I love java.

OUTPUT

Enter a line :: I love java.

I love java.

© A+ Computer Science - www.apluscompsci.com

The `nextLine()` method will read in an entire line of text including whitespace (enter keys, spaces, tabs, etc.). Any text value entered will be accepted, including a line containing spaces.

In the example, the next line of data entered on the keyboard would be read in and placed in variable `sentence`.

Open scannerstrings.java

© A+ Computer Science - www.apluscompsci.com

nextLine() issues

```
out.print("Enter an integer :: ");  
int num = keyboard.nextInt();  
out.print("Enter a sentence :: ");  
String sentence = keyboard.nextLine();  
out.println(num + " " + sentence);
```

OUTPUT

```
Enter an integer :: 34  
Enter a sentence :: 34
```

INPUT

```
34  
picks up \n
```

nextLine() picks up whitespace.

© A+ Computer Science - www.apluscompsci.com

The `nextLine()` method will read in an entire line of text including the enter key. Any text value entered will be accepted, including a line containing spaces.

After 34 is typed in, enter must be pressed to get the system to register the 34.

`nextInt()` reads in the 34 and stores it in `num`.

`nextInt()` reads up to the enter key(`\n`) typed in after the 34.

`nextLine()` reads in the enter(`\n`) and stores it in `sentence`.

This is a problem.

nextLine() issues

```
out.print("Enter an integer :: ");
int num = keyboard.nextInt();
keyboard.nextLine(); //pick up whitespace
out.print("Enter a sentence :: ");
String sentence = keyboard.nextLine();
out.println(num + " " + sentence);
```

OUTPUT

```
Enter an integer :: 34
Enter a sentence :: picks up \n
34 picks up \n
```

INPUT

```
34
picks up \n
```

nextLine() picks up whitespace.

© A+ Computer Science - www.apluscompsci.com

The `nextLine()` method will read in an entire line of text including the enter key. Any text value entered will be accepted, including a line containing spaces.

After 34 is typed in, enter must be pressed to get the system to register the 34.

`nextInt()` reads in the 34 and stores it in `num`.

`nextInt()` reads up to the enter key(`\n`) typed in after the 34.

A `nextLine()` is placed after the `nextInt()` to read in the enter(`\n`). The additional `nextLine()` picks up the enter(`\n`) left behind by `nextInt()`;

Now, `nextLine()` can read in the line and store it in `sentence`. The problem has been solved.

Open nextlineissues.java

© A+ Computer Science - www.apluscompsci.com

Multiple Inputs

INPUT

1 2 3 4 5

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

```
out.println(keyboard.nextInt());  
out.println(keyboard.nextInt());  
out.println(keyboard.nextInt());
```

OUTPUT

1
2
3

© A+ Computer Science - www.apluscompsci.com

Scanner can be used to read in multiple values on one line as long as whitespace is entered in between each value on the line. If whitespace is not used to separate the values, the values would be considered one value.

For the example, if 1 2 3 4 5 is entered. Only values 1 2 3 are read in because the code only had 3 `nextInt()` method calls.

If 12345, was entered with no spaces, then 12345 would be the first and only value read in.

Open multiread.java

© A+ Computer Science - www.apluscompsci.com

Old School Input

```
BufferedReader keyboard =  
    new BufferedReader(  
        new InputStreamReader( System.in ) );  
  
System.out.print("Enter a word :: ");  
String s = keyboard.readLine();  
  
System.out.println(s + '\n' );
```

© A+ Computer Science - www.apluscompsci.com

Back in the day, `BufferedReader` was used to perform basic input operations. You can still use it today, but with the introduction of the `Scanner` class, there is really no reason to use `BufferedReader`.

Old School Input

`readLine()` reads in all data as text / string data.
The text you read in must be converted over to the appropriate type before it can be stored.

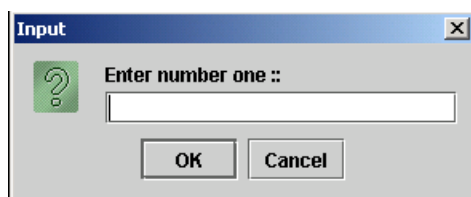
```
System.out.print("Enter an integer :: ");  
one = Integer.parseInt(keyboard.readLine());
```

```
System.out.print("Enter a double :: ");  
two = Double.parseDouble(keyboard.readLine());
```

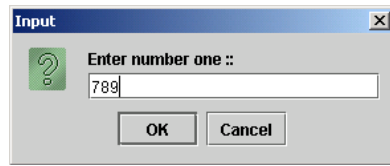
Open
oldschoolone.java
oldschooltwo.java

© A+ Computer Science - www.apluscompsci.com

GUI



© A+ Computer Science - www.apluscompsci.com



//GUI INPUT BOX

```
input= JOptionPane.showInputDialog("Enter an integer :: ");  
one = Integer.parseInt(input);
```

//GUI OUTPUT BOX

```
JOptionPane.showMessageDialog(null, "Integer value :: " + one);
```

**Open
guihelp.java**

© A+ Computer Science - www.apluscompsci.com

GUI boxes for input and output are very fun and look really cool. These boxes can be used to perform the same input operations as performed with Scanner.

Start work on Lab 0c

© A+ Computer Science - www.apluscompsci.com