Scanner

A Scanner object reads from an input source (keyboard, file, String, etc)

— `next()` returns the next token as a String

— `nextInt()` returns the next token as an `int`

— `nextDouble()` returns the next token as a `double`

— `nextLine()` returns the next line as a `String`

A token is a sequence of characters not including any whitespace.

Scanner tests

You can test whether a read will succeed.

— `hasNext()` is true if `next()` will succeed

— `hasNextInt()` is true if `nextInt()` will succeed

— `hasNextDouble()` is true if `nextDouble()` will succeed

— `hasNextLine()` is true if `nextLine()` will succeed

Each will pause your program if there is not enough information (eg, waiting for <return> on keyboard).

Scanner sources

Common sources for a Scanner are the keyboard, a file, or just a String.

```
Scanner a = new Scanner(System.in);            // "standard input"
Scanner b = new Scanner(new File("foo.txt"));  // File "foo.txt"
Scanner c = new Scanner("Hello\nWorld!");      // A String
```

Input buffer

String and File Scanners read entire source into input buffer (along with end-of-input indicator).

```
new Scanner("Hello\nWorld!");
```

Resulting input buffer:

```
Hello\nWorld!<EOF>
^
```

Input buffer

```
foo.txt:
----------
A
AB
ABC            <- may end with newline or not
----------
```

```
new Scanner(new File("foo.txt"));
```

Resulting input buffer:

```
A\nAB\nABC<EOF>      OR (depending if newline at end of file)
^
A\nAB\nABC\n<EOF>
^
```

Input buffer

`System.in` Scanner adds to buffer each <return> press.

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

Resulting input buffer after user types

`Hello` <return> `World!` <return> `Boo`

```
Hello\nWorld!\n        <== No EOF, No Boo
^
```

What methods do

`next()`: skip whitespace, build token, return token
`nextInt()`: skip whitespace, build token, convert to `int`
`nextLine()`: return everything up to next `\n` or EOF

```
A\nAB\nABC\n<EOF>  nextLine()  A\nAB\nABC\n<EOF>
^                  "A"            ^


A\nAB\nABC\n<EOF>  nextLine()  A\nAB\nABC\n<EOF>
   ^               "AB"                ^


A\nAB\nABC\n<EOF>  nextLine()  A\nAB\nABC\n<EOF>
       ^           "ABC"                   ^
```

What happens next?

```java
public static void main(String[] args) {
    Scanner in = new Scanner("A\nAB\nABC\n");
    System.out.println(":"+in.nextLine()+":");
    System.out.println(":"+in.nextLine()+":");
    System.out.println(":"+in.nextLine()+":");
    System.out.println(":"+in.nextLine()+":");
}
```

```
:A:
:AB:
:ABC:
Exception in thread "main" java.util.NoSuchElementException: No line found
    at java.base/java.util.Scanner.nextLine(Scanner.java:1651)
    at Untitled.main(Untitled.java:11)
```

EOF by itself is not a line. Write small programs to answer "what if" questions.

More examples

```
A\nAB\nABC\n<EOF>  nextInt()   A\nAB\nABC\n<EOF>
^                  Exception!  ^

A\nAB\nABC\n<EOF>  next()   A\nAB\nABC\n<EOF>
^                  "A"        ^

A\nAB\nABC\n<EOF>  next()   A\nAB\nABC\n<EOF>
  ^                "AB"           ^

A\nAB\nABC\n<EOF>  nextLine()   A\nAB\nABC\n<EOF>
      ^            ""                  ^
```

Programming

Write a program that repeatedly prompts the user for input and for each line grabs each token from the response and classifies the token as an `int`, `double`, or `String`.

Pseudocode (big picture):

```
prompt for line
while there is a line to read
    process line                <= could be method call
    prompt for line
```

Programming

More detail:

```
prompt for line
while there is a line to read
    read a line
    while there are tokens left in the line
        if next token is a int
            read token and output "int"
        else if next token is an double
            read token and output "double"
        else
            read token and output "String"
    prompt for line
```

```java
import java.util.Scanner;

public class Identifier {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter a line of text: ");
        while (in.hasNextLine()) {
            Scanner line = new Scanner(in.nextLine());
            while (line.hasNext()) {
                if (line.hasNextInt()) {
                    System.out.println(line.next() + " is an int");
                } else if (line.hasNextDouble()) {
                    System.out.println(line.next() + " is a double");
                } else {
                    System.out.println(line.next() + " is a String");
                }
            }
            System.out.print("Enter a line of text: ");
        }
    }
}
```

Things to note in sample program

- I used next() when printing results so it echoes exactly the token. If I used nextDouble() it would convert the token to a double and might change the formatting.
- Every int can be read as a double, so the test for int has to come before the test for double.