File I/O summary

read keyboard using Scanner

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

read keyboard using BufferedReader

```
BufferedReader input = new BufferedReader (new InputStreamReader(System.in));
```

read file using Scanner

```
Scanner input = new Scanner (new File(filename));
```

read file using buffered reader

```
BufferedReader brFile = new BufferedReader (new FileReader (f));
```

read URL

```
URL myURL = new URL("http://www.yahoo.com/");
URLConnection urlConn = myURL.openConnection();

BufferedReader brWeb = new BufferedReader( new InputStreamReader( urlConn.getInputStream() ));
```

read URL (simpler)

```
BufferedReader brWeb = new BufferedReader(new InputStreamReader(myURL.openStream()));
```

write to file (PrintWriter)

```
PrintWriter pwFile = new PrintWriter( new BufferedWriter( new FileWriter (f)));

with flushing instead of buffering
PrintWriter pwFile = new PrintWriter( new FileWriter (f), true);
```

write to file (BufferedWriter)

```
BufferedWriter bwFile =  new BufferedWriter( new FileWriter (f)));
```

Location of files

```
System.out.println(  System.getProperty("user.dir") );
```

Or use absolute path names (but then it won't work on other computers)
Or make a resource folder and put things in there.

Methods

How to decide on Scanner vs BufferedReader? The final IO object that you create, the one that wraps all of the others, determines which methods you use to read/write the data.
My quick rule-of-thumb: use Scanner for keyboard input and use the buffered objects for everything else.

```
Scanner
          nextInt(), next(), nextLine(), ...

BufferedReader extends Reader:
          read()  -- reads a character (as int).
          readLine()

PrintWriter extends Writer:
          print(), println(), printf()
          write()

BufferedWriter extends Writer:          FileWriter extends Writer:
          write()
```

Quick code for reading files (URLs are the same)

(add your own try-catch)
(and also close the readers)

I'm not sure if you need to declare things outside the try-catch.
It depends on how your code is set up.

BufferedReader

```
try {
        BufferedReader brf = new BufferedReader (new FileReader(f));
        String str = null;
        //read until the end of file (or URL)
        while( (str = brf.readLine() ) != null) {
                System.out.println(str);  //process the line
        }
} catch (...) { ...
```

Scanner

```
try {
        Scanner scf = new Scanner (new FileReader(f));
         //check for input. If no input, then we've reached the end of the file/URL
        while(scf.hasNext()) {
                String str = scf.nextLine();
                System.out.println(str);          //process the line
        }
} catch (...) { ...
```

Strange things (not important)

If you close System.in, you can no longer get keyboard input.

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

OR

```
BufferedReader input = new BufferedReader (
                       new InputStreamReader(System.in));
input.close() ; //KEYBOARD IS CLOSED HERE. R.I.P.
```

If you close System.out, you can no longer display things to the screen.

```
PrintWriter pw = new PrintWriter(System.out);

pw.close();   //MONITOR IS CLOSED.  R.I.P.

System.out.println("Nothing will print!");
```

It doesn't seem that you can reopen these once they are closed. You have to start the program again.