Reading a text file

Class `java.io.BufferedReader` provides methods for reading lines from a file of characters, like a .txt file. It's pretty simple. Once a `BufferedReader` object `bf` has been created for a file, calling method `bf.readLine()` reads and returns a line of text. If there are no more lines to read, `bf.readLine()` returns **`null`**. After reading the file, close the file by calling method `bf.close()`.

We give the pattern that *all methods that read a text file should use*, assuming that `Path p` describes the file to be read:

```
// Read and process all lines of the character file given by Path p.
BufferedReader bf= Files.newBufferedReader(p);   // Store a new BufferedReader for p in bf;
String lin= bf.readLine();   // Read first line. Note: If file is empty, there is no first line.

// invariant: All lines before line lin have been read and processed, and
//          line lin has been read but not yet processed
while (lin != null) {
     Process line lin;

     lin= bf.readLine();
}
bf.close();
```

Here are important points about this pattern.

1. Always get a `BufferedReader` using function `Files.newBufferedReader(p)`.
2. Read the first line *before* the loop. If the file length is 0, which can happen, `lin` will be set to null initially and the repetend will not be executed.
3. The goals of the repetend are: (1) Make progress toward termination, by reading the next line into `lin`, and (2) In order to keep the loop invariant true, process line `lin` *before* reading the next line. The meaning of "Process" depends on the context. We give an example below.
4. Once the file has been read, close it, using method `bf.close`. This statement comes after the loop.

An example

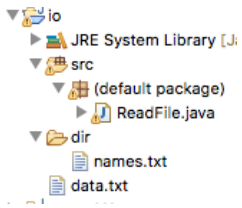
Consider the Eclipse project shown to the right below. We give the code to read and print file data.txt:

```
Path p= Paths.get("data.txt");
BufferedReader bf= Files.newBufferedReader(p);

String lin= bf.readLine();

// invariant: All lines before line lin have been read and processed, and
//          lin has been read but not yet processed
while (lin != null) {
  System.out.println(lin);

  lin= bf.readLine();
}
bf.close();
```



Using Legacy class File instead Path

Below, the first line creates an object of class `File` for file data.txt. The second line crease a `BufferedReader bf` for the file. Thereafter, the lines of the class can be read as shown above. Class `FileReader` reads streams of characters. Wrapping it in a `BufferedReader` object makes it possible to read one line at a time.

```
File f= new File("data.txt");
BufferedReader bf= new BufferedReader(new FileReader(f));
```