Programming Task 1

You are implementing a client-server communication interface.
The client sends commands to the server using a line-oriented text protocol.
Lines containing only text “begin” start a new command.
Lines containing only text “end” end a command.
Any lines between “begin” and “end” constitute the command (“begin” and “end” are not part of a command).
The client must send at least one command.
There must be at least one line between “begin” and “end”.
There must be no extra input outside of the “begin”…”end” blocks.
The client can only close connection right after sending an “end” line.

Input shall be read using InputReader which has following methods:
getNextLine(): returns a String object that contains the next line of input, without the newline character. Usually this method will block until a new line is received from the client, but in the following cases the method returns null:

- Network failure (TCP connection broken)
- Client breaks the connection.

Commands are processed by CommandProcessor which has following methods:
processCommand(String cmdString): you should call this method after each full command that is received.
processError(): you should call this method on all error situations and stop processing the input.

Your task is to implement class which reads input from InputReader and passes parsed commands to CommandProcessor. An abstract implementation AbstractInputParser is already implemented. Extend this class and implement method **processInput()**. You can assume that parameters given in constructor are never null. Parser is also responsible for validating incoming data. If client doesn’t follow protocol specification, it should be handled as an error. If client closes the connection or an error occurs, input processing should end.

Example input with line numbers. Each line is returned by calling getNextLine().

```
1 begin
2 abcd
3 efg
4 end
5 begin
6 sdfs
7 end
(null)
```

When line 4 is retrieved, you should call processCommand(“abcd\nefg\n”).
When line 7 is retrieved, you should call processCommand(“sdfs\n”).

CommandProcessor.java

```
public interface CommandProcessor {

     public void processCommand(String command);
     public void processError();
}
```

InputReader.java

```
public interface InputReader {

     public String getNextLine();
}
```

AbstractInputParser.java

```
public abstract class AbstractInputParser {

     protected InputReader reader;
     protected CommandProcessor processor;

     public AbstractInputParser(
          InputReader reader, CommandProcessor processor){

          this.reader = reader;
          this.processor = processor;
     }

     public abstract void processInput();
}
```