**Epipog**

**Specification**

**Reader Class Family**

**Nov. 4, 2016**

# 1. Base Class

The reader object is the base class for reading data input from a file. Derived objects are extended from the base class, depending on their requirement in the database pipeline. This base class is defined in the file Reader.java. This base class defines the methods and interfaces for how data is read from a file into the Epipog application and the operations that may be performed with the reader object.

## 1.1 Fields

The base class defines the following fields:

protected String inputFile;

This field holds the path name to the input file to be read and then parsed . Access to the data value is limited to the derived object (protected).

protected long fileSize = 0;

This field holds the byte size on disk of the input file. Access to the data value is limited to the derived object (protected).

## Types

The base class defines the following data types:

public enum ReaderType {  
 READERMEM,  
 READERLINE,  
 READERMAPPED  
}

This enumerated type are the setting flags for specifying which derived reader object to use for reading the input data file.

READERMEM : reads the entire file into memory.  
 READERLINE : reads one line of input data from the file on disk at a time.  
 READERMAPPED: parts of the file are mapped into memory when a page fault occurs.

## 1.3 Methods

The base class contains the following implemented methods

***Accessors (Getter/Setter)***

public String InputFile();

This getter method returns the path name of the input data file.

public Long FileSize();

This getter method returns the file size of the input data file.

## 1.4 Abstract Methods (Interface)

The base class contains abstract methods, which must be implemented by the derived classes, for the following:

***I/O***

public abstract void Open() throws ParseException;

The method opens the input data file for reading by the reader object. If an error occurs, a ParseException is thrown.  
  
public abstract void Close();

The method closes the input data file, which had been previously opened.   
  
public abstract String ReadLine() throws ParseException;

This method reads the next line of data (i.e., ends in newline) from the input data file and returns as a UTF-8 string. If an error occurs, a ParseException is thrown.

# 

# 2 ReaderMem Derived Class

This derived class extends base class “Reader”, and implement the method for reading the input data file. They are implemented in the file ReaderMem.java. This extended class implements the methods and interfaces for reading the input data, when the entire input file is read into memory. This method is typically used on small files (< 1MB), which gives the fastest read/access performance and still limiting the size of the memory footprint.

## 2.1 Fields

The extended class defines the following additional fields, which are only accessible (private) by this class.

private List<String> lines = null; // list of lines read in from input file  
private int nLines = 0; // total number of lines in input file  
private int currLine = 0; // current line number being read

## Methods

The extended class contains the implementation of the following methods:

***I/O***

public void Open() throws ParseException;

This method opens the input data file and reads it entirely into memory. The input data is separated into lines as UTF-8 strings and stored in the private field List<String> lines. The input file is automatically closed when read in. If an error occurs opening or reading the file, a ParseException is thrown.

public void Close();

This method does nothing, since the file is already closed.

@Override public String ReadLine();

This method reads the next line in the input file, which has been read entirely into memory. Blank lines in the file are skipped. Upon EOF, it will return null. The method overrides the base class definition, in that it does not throw any exception.

# ReaderLine Derived Class

This derived class extends the base class “Reader”, and implement the method for reading the input data file. They are implemented in the file ReaderLine.java. This extended class implements the methods and interfaces for reading the input data, when the one line at a time is read from the file (storage) into memory. This method is typically used on medium size files , which trades off disk access for small footprint in memory.

## 3.1 Fields

The extended class defines the following additional fields.

private BufferedReader br = null;

This field is the byte buffered reader for the input data file and is only accessible by this class object.

## Methods

The extended classes contain the implementation of the following methods:

***I/O***

public void Open() throws ParseException;

This method opens the input data for buffered I/O reading in UTF-8 encoding. If an error occurs opening the file, a ParseException is thrown.

public void Close();

This method closes the file if it is open; otherwise it does nothing.

public String ReadLine() throws ParseException;

This method reads the next line in the input file. Each line is read directly from the file into a buffered input stream. Blank lines in the file are skipped. Upon EOF, it will return null. If an error occurs during reading, a ParseException is thrown.

# ReadMapped Derived Class

This derived class extends the base class “Reader”, and implement the method for reading the input data file. They are implemented in the file ReaderMapped.java. This extended class implements the methods and interfaces for reading the input data, using a memory mapped file. With a memory mapped file, the OS will load an initial portion of the file into memory. The reader access the file as if it was entirely in memory. When a page fault occurs, the OS will map another section of the file into memory. The amount of memory that is mapped is determined by the OS. This method is typically used on large files , which attempts to balance the footprint in memory and number of repetitive read accesses from storage.

## 4.1 Fields

The extended class defines the following additional fields.   
  
private int currPosition = 0; // current (read) position in input data  
private MappedByteBuffer buffer = null; // memory mapped file buffer  
private FileChannel fileChannel = null; // the file’s I/O channel

The field buffer contains the input data that has been mapped into memory. When the buffer is accessed beyond the boundary of what has been read in, a page fault occurs and the OS maps in additional portion of the file into memory. The field currPosition is the current read position in the memory mapped buffer.

## Methods

The extended classes contain the implementation of the following methods:

***I/O***

public void Open() throws ParseException;

This method opens the input data for reading in UTF-8 encoding, and maps the file (initial portion) into the memory mapped buffer. If an error occurs opening the file or mapping the file into memory, a ParseException is thrown.

public void Close();

This method closes the file if it is open; otherwise it does nothing.

public String ReadLine() throws ParseException;

This method reads the next line in the input file. Each line is read directly from the memory mapped buffer. If a page fault occurs, the OS will map in the next portion of the file. bBlank lines in the file are skipped. Upon EOF, it will return null. If an error occurs during reading, a ParseException is thrown.