# PENGAKSESAN FILE DI JAVA

Rachmat Selamet

Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI
Jl. Ir. H. Juanda 96 Bandung 40132

E-mail : rachmatselametskom@gmail.com

**ABSTRAK**

File adalah tempat untuk menyimpan / membaca data yang tersimpan di dalam disk secara permanen. Java sudah menyediakan kelas yang mengakses file di dalam paket io. Kelas yang tersedia sudah mampu digunakan untuk mengakses file dalam bentuk file binary, file text maupun file kompresi zip. Untuk menghubungkan file yang tersimpan dalam disk dengan java, menggunakan kelas File. Untuk mengakses file binary digunakan kelas FileOutputStream dan FileInputStream. Untuk mengakses file text digunakan kelas FileReader dan FileWriter. Untuk mengakses file zip digunakan kelas File zip berjenis zip lebih baik dari gzip karena menyimpan informasi lebih lengkap.

*Kata Kunci : Java, File, File Text, File Binary, File Zip*

## 1 PENDAHULUAN

Secara umum, struktur data yang ada selain file, tersimpan di memori, sehingga data yang tersimpan akan hilang setelah program dimatikan. Agar data tersimpan secara pemanen dalam disk, digunakan struktur file.

Dalam program java, file terdapat dalam paket io. Terdapat 4 kelas utama untuk mengakses file, yaitu File, File streaming, File reader writer, dan File zip. Kelas file merupakan kelas yang menghubungkan file dalam disk dengan file sebagai *object*. Kelas file streaming berguna untuk mengakses file binary (file yang datanya tidak dapat dibaca secara langsung karena datanya diubah dalam bentuk biner oleh program). Kelas file reader writer berguna untuk mengakses file text (file yang datanya dapat dibaca secara langsung menggunakan notepad). File zip merupakan file kompresi yang paling banyak digunakan. Terdapat 2 jenis zip yang dapat diakses, yaitu gzip dan zip.

## 2 FILE

Merupakan kelas untuk mengakses file secara umum di sistem operasi. Cara deklarasi :

```
File file = new File("in.txt");     // file yang ada di direktori skrg
File file = new File("d:\\myproject\\java\\Hello.java");  // file dengan absolute path
File dir  = new File("c:\\temp");   // direktori/folder
```

Method dari File :

a. public boolean exists()  // untuk memeriksa file/folder ada
b. public long length()  // menghitung panjang dari file
c. public boolean isDirectory()  // memeriksa apakah folder/bukan
d. public boolean isFile()  // memeriksa apakah file/bukan
e. public boolean canRead()  // memeriksa apakah file dapat dibaca
f. public boolean canWrite()  // memeriksa apakah file dapat ditulis
g. public boolean delete()  // menghapus file/folder
h. public void deleteOnExit()  // menghapus file ini setelah program keluar
i. public boolean renameTo(File dest) // mengubah nama file ini
j. public boolean mkdir()  // membuat folder
k. public String[] list()  // mengambil isi folder ini dalam array of string
l. public File[] listFiles()  // mengambil isi folder ini dalam array of File
m. public String[] list(FilenameFilter filter) //mengfilter isi folder
n. public File[] listFiles(FilenameFilter filter) //mengfilter isi folder
o. public File[] listFiles(FileFilter filter) //mengfilter isi folder
p. public boolean accept(File dir, String file) //untuk menerima file/folder yang sudah difilter

Contoh File1.java

```
import java.io.File;
import java.io.FilenameFilter;
public class ContohFile1 {
  public static void main(String[] args) {
    File dir = new File(".\\src\\ContohFile1");
    if (dir.isDirectory()) {
        String[] files = dir.list(new FilenameFilter() {
        public boolean accept(File dir, String file) {
          return file.endsWith (".java");
        }
      });
      for (String file : files) {
        System.out.println(file);
      }
    }
  }
}
```

## 3 FILE BINARY

Untuk mengakses file binary digunakn kelas FileOutputStream dan FileInputStream.

Method dari FileOutputStream :

a. public void close() throws IOException{}

   // menutup fileoutputstream

b. protected void finalize() throws IOException {}

   // membersihkan koneksi dari file dan memastikan file sudah ditutup dan tidak terhubung dengan stream

c. public void write (int w) throws IOException{}

   // menuliskan byte ke fileoutputstream

d. public void write(byte[] w)

   // menuliskan sejumlah byte dari array ke fileoutputstream

e. public void flush()

   // membersihkan stream dan menyimpan seluruh stream ke file

Method dari FileInputStream :

a. public void close() throws IOException{}

   // menutup fileoutputstream

b. protected void finalize()throws IOException {}

   // membersihkan koneksi dari file dan memastikan file sudah ditutup dan tidak terhubung dengan stream

c. public int read(int r)throws IOException{}

// membaca data perbyte dari fileinputstream dan menghasilkan -1 untuk EOF (End Of File)

d. public int read(byte[] r) throws IOException{}

// membaca data dari fileinputstream dan menghasilkan -1 untuk EOF (End Of File)

e. public int available() throws IOException{}

// menghitung jumlah byte data yang dapat diambil oleh inputstream

Contoh FileStream.java

```
public static void main(String[] args)throws IOException{
      String nama;       int jumlah;       double harga;
      char tanya;
      File file=new File("jual.dat");
      FileOutputStream out=new FileOutputStream(file);
      DataOutputStream outs=new DataOutputStream(new    BufferedOutputStream(out));
      byte[] isi;
      System.out.printf("Nama : ");nama=scn.next();
      while (!nama.equalsIgnoreCase("x")){
         System.out.printf("Jumlah : "); jumlah=scn.nextInt();
         System.out.printf("Harga  : "); harga=scn.nextDouble();
         System.out.printf("Disimpan(Y/T)? "); t anya=scn.next().toUpperCase().charAt(0);
         If (tanya=='Y'){
            isi=(nama+",").getBytes();              outs.write(isi);
            outs.writeInt(jumlah); outs.writeDouble(harga);              outs.flush();          }
         System.out.println();
         System.out.printf("Nama : ");nama=scn.next();
      }
      if(out!=null) out.close();
      FileInputStream in=new FileInputStream("jual.dat");
      DataInputStream ins=new DataInputStream(new BufferedInputStream(in));
      System.out.println("Nama    Jumlah    Harga    Bayar");
      while(ins.available()>0){
            tanya=(char)ins.readByte();
            nama="";
            while(tanya!=','){
               nama+=tanya;
               tanya=(char)ins.readByte();
            }
            jumlah=ins.readInt();
            harga=ins.readDouble();
            System.out.printf("%s %d %f %f\n",nama,jumlah,harga,jumlah*harga);
      }
      System.out.println();
      if(in!=null) in.close();
}
```

## 4 FILE TEXT

Untuk mengakses file text digunakan kelas FileReader, kelas FileWriter, dan kelas Scanner.

Method dari FileWriter dan Buffered :

a. public void close() throws IOException{ }
   // menutup filewriter
b. protected void finalize()throws IOException { }
   // membersihkan koneksi dari file dan memastikan file sudah ditutup dan tidak terhubung dengan buffer
c. public void write(String w)
   // menuliskan sejumlah byte dari array ke filewriter
d. public void flush()
   // membersihkan buffer dan menyimpan seluruh buffer ke file

Method dari FileReader dan Scanner :

a. public void close() throws IOException{ }
   // menutup filereader
b. protected void finalize()throws IOException { }
   // membersihkan koneksi dari file dan memastikan file sudah ditutup dan tidak terhubung dengan buffer
c. public boolean hasNext()
   // memeriksa apakah ada data selanjutnya dalam scanner
d. public String next()
   // mengambil string dari scanner
e. public int nextInt()
   // mengambil integer dari scanner
f. public int nextFloat()
   // mengambil float dari scanner
g. public int nextDouble()
   // mengambil double dari scanner

Contoh FileReaderWriter.java

```
public static void main(String[] args)throws IOException{
      String nama;       int jumlah;       double harga;       char tanya;
      File file=new File("jual.txt");
      FileWriter fw=new FileWriter(file);
      BufferedWriter bw=new BufferedWriter(fw);
      System.out.printf("Nama : ");nama=scn.next();
      while(!nama.equalsIgnoreCase("x")){
         System.out.printf("Jumlah : ");jumlah=scn.nextInt();
         System.out.printf("Harga  : ");harga=scn.nextDouble();
         System.out.printf("Disimpan(Y/T)? ");tanya=scn.next().toUpperCase().charAt(0);
         if(tanya=='Y'){
            bw.write(nama+" ");              bw.write(String.valueOf(jumlah)+" ");
           bw.write(String.valueOf(harga)+" ");              bw.flush();
         }
         System.out.println();
         System.out.printf("Nama : ");nama=scn.next();
      }
      if(fw!=null) fw.close();

      FileReader fr=new FileReader("jual.txt");
      Scanner fscn=new Scanner(fr);
      while(fscn.hasNext()){
         nama=fscn.next();
         jumlah=fscn.nextInt();
         harga=fscn.nextDouble();
         System.out.printf("%s %d %f %f\n",nama,jumlah,harga,jumlah*harga);
      }
      if(fr!=null)    fr.close();
}
```

## 5 FILE ZIP

Terdapat 2 jenis file zip yang dapat diakses, yaitu gzip dan zip. Untuk membuat file zip menggunakan ZipOutputStream yang terdapat dalam paket java.util.zip.

Berikut contoh penggunaan ZipOutputStream dengan menggunakan file tambahanjurnal.txt yang dikompres menjadi tes.zip di dalam folder periksa drive D:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.zip.ZipOutputStream;
import java.util.zip.ZipEntry;
public class ContohZip {

 public static void main( String[] args )
 {
  ContohZip zipObj = new ContohZip();
  zipObj.zipMyFile();
 }
```

```
  public void zipMyFile(){
    byte[] buffer = new byte[1024];
    try{
      ZipOutputStream gos = new ZipOutputStream(new FileOutputStream
("D://Periksa/tes.zip"));

      FileInputStream fis =   new FileInputStream("D:/Periksa/tambahanjurnal.txt");

      ZipEntry ze= new ZipEntry("/Periksa/tambahanjurnal.txt");
      gos.putNextEntry(ze);

      int length;
      while ((length = fis.read(buffer)) > 0) {
        gos.write(buffer, 0, length);
      }

      fis.close();
      gos.finish();
      gos.close();

      System.out.println("File Compressed!!");

    }catch(IOException ioe){
       ioe.printStackTrace();
     }
  }
}
```

Berikut contoh penggunaan GZIPOutputStream dengan menggunakan file tambahanjurnal.txt yang dikompres menjadi tes.gz di dalam folder periksa di drive D :

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.zip.GZIPOutputStream;
public class ContohZip {

  public static void main( String[] args )
  {
    ContohZip zipObj = new ContohZip();
    zipObj.gzipMyFile();
  }
  public void gzipMyFile(){
    byte[] buffer = new byte[1024];
    try{
      GZIPOutputStream gos =
       new GZIPOutputStream(new FileOutputStream("D:/Periksa/tes.gz"));

      FileInputStream fis =  new FileInputStream("D:/Periksa/tambahanjurnal.txt");

      int length;
      while ((length = fis.read(buffer)) > 0) {
        gos.write(buffer, 0, length);
```

```
        }

        fis.close();
        gos.finish();
        gos.close();

        System.out.println("File Compressed!!");

      }catch(IOException ioe){
         ioe.printStackTrace();
       }
    }
}
```

Berikut contoh penggunaan ZipInputStream dengan menggunakan file tes.zip yang dibuka ke folder periksa/2 di drive D:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.zip.ZipInputStream;
import java.util.zip.ZipEntry;
public class ContohZip {

 public static void main( String[] args )
 {
  ContohZip zipObj = new ContohZip();
  zipObj.unZipIt("d:/periksa/tes.zip", "d:/periksa/2");
 }
  public void unZipIt(String zipFile, String outputFolder){
  byte[] buffer = new byte[1024];

   try{
         File folder = new File(outputFolder);
         if(!folder.exists()){
                  folder.mkdir();
         }
         ZipInputStream zis =
                  new ZipInputStream(new FileInputStream(zipFile));
         ZipEntry ze = zis.getNextEntry();
         while(ze!=null){
            String fileName = ze.getName();
        File newFile = new File(outputFolder + File.separator + fileName);

        System.out.println("file unzip : "+ newFile.getAbsoluteFile());

         new File(newFile.getParent()).mkdirs();

         FileOutputStream fos = new FileOutputStream(newFile);

         int len;
         while ((len = zis.read(buffer)) > 0) {
                  fos.write(buffer, 0, len);
```

```
            }
            fos.close();
            ze = zis.getNextEntry();
             }
        zis.closeEntry();
            zis.close();
            System.out.println("Done");

    }catch(IOException ex){
      ex.printStackTrace();
    }
   }
}
```

Berikut contoh penggunaan GZIPInputStream dengan menggunakan file tes.gz untuk membuka file menjadi hasil.txt di dalam folder periksa di drive D :

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.zip.GZIPInputStream;
public class ContohZip {

  public static void main( String[] args )
  {
   ContohZip zipObj = new ContohZip();
    zipObj.gunzipIt("d:/Periksa/tes.gz", "d:/Periksa/hasil.txt");
     zipObj.unZipIt("d:/periksa/tes.zip", "d:/periksa/2");
  }

    public void gunzipIt(String INPUT_GZIP_FILE,String OUTPUT_FILE){

    byte[] buffer = new byte[1024];
    try{
           GZIPInputStream gzis =
           New GZIPInputStream(new FileInputStream(INPUT_GZIP_FILE));

           FileOutputStream out =       new FileOutputStream(OUTPUT_FILE);
      int len;
      while ((len = gzis.read(buffer)) > 0) {
          out.write(buffer, 0, len);
      }
      gzis.close();
          out.close();
          System.out.println("Done");

    }catch(IOException ex){
      ex.printStackTrace();
    }
   }
}
```

## 6 KESIMPULAN

Kesimpulan yang dapat ditarik dari jurnal ini adalah sebagai berikut :

a. Java sudah menyediakan kelas untuk mengakses file dalam 4 bentuk, yaitu untuk file yang langsung berada di dalam OS (File.java), file binary (FileOutputStream.java dan FileInputStream.java), file text (FileReader.java dan FileWriter.java) dan file zip (GZIPOutputStream.java, GZIPInputStream.java, ZipOutputStream.java dan ZipInputStream.java).
b. Secara umum mendeklarasikan file yang akan digunakan menggunakan kelas File.java dengan konstruktor File(nama file beserta *full path*).
c. Setelah file selesai digunakan, pastikan dengan menutup file menggunakan method close.
d. File text dapat dibaca datanya secara langsung menggunakan kelas Scanner dengan menambahkan spasi sebagai pemisah antar data.
e. Dalam file stream, untuk menuliskan tipe data string harus dikonversi dalam bentuk array dengan data byte.
f. File kompresi zip dapat dibuat dengan menggunakan kelas GZIPOutputStream atau ZipOutputStream
g. File kompresi zip dapat dibuka dengan menggunakan kelas GZIPInputStream atau ZipInputStream
h. File zip dari zip lebih baik dari gzip, karena menyimpan semua informasi baik folder maupun nama file yang dikompres.

## DAFTAR PUSTAKA

[1] https://docs.oracle.com.
[2] http://www.javatpoint.com.
[3] http://beginnersbook.com.
[4] http://www.tutorialspoint.com.
[5] https://www.mkyong.com
[6] https://examples.javacodegeeks.com
[7] http://www.java2novice.com