Java ファイルの分割結合

[Java]
java sdk 5.0 以上

簡易ファイル分割結合ツール。JavaSDK がインストールされた PC でコンパイルして使ってください。

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.util.Arrays;

public class Split {
  private final int BUF_SIZE_1KB = 1024; //1kb
  private final String SEQ_LEN = "3";
  private final String FILE_SEQ_SUFFIX = "%0"+ SEQ_LEN + "d";
  private final String FILE_NAME_REP   = "_[0-9]{"+ SEQ_LEN + "}$";

  public static void main(String[] args) {
    Split me = new Split();
    int splitSize = 1000; // 1mb default
    try {
      String mode = args[0];
      if ("-help".equals(mode)) {
        Split.printUsage();
        return;
      } else if
        ("s".equals(mode) || "-s".equals(mode)) {

        try {
          splitSize = Integer.parseInt(args[3]);
        } catch (Exception e) {}

        me.split(args[1], args[2], splitSize);

      } else if
        ("c".equals(mode) || "-c".equals(mode)) {

        me.concat(args[1]);
      } else {
        Split.printUsage();
      }

    } catch (Exception e) {
      Split.printUsage();
      e.printStackTrace();
    }
    System.out.println("finish");
  }
  /**
   * ファイルを指定サイズ(KB)で分割
   * @param inFile 対象ファイル
   * @param outDir 出力ディレクトリ
   * @param sizeKb サイズ
   * @throws Exception
   */
  public void split(String inFile, String outDir, int sizeKb) throws Exception {

    File od   = createDir(outDir);

    byte[] buf = new byte[BUF_SIZE_1KB];

    BufferedInputStream is = getBufferedInputStream(inFile);
    BufferedOutputStream os = null;

    int sizeLimit = sizeKb;
    int sizeCnt = 0;
    int idx = 1;
    int ret = -1;
    boolean isFirstCreateFlg = true;
```

```
      while((ret = is.read(buf)) > 0) {
        if ((sizeCnt >= sizeLimit) || isFirstCreateFlg) {
          closeBufferdOutputStream(os);
          os = getBufferedOutputStream(
               absolutePath(od,
                (new File(inFile)).getName() + "_" + String.format(FILE_SEQ_SUFFIX, idx++))
             );
          isFirstCreateFlg = false;
          sizeCnt = 0;
        }
        os.write(buf,0, ret);
        os.flush();
        sizeCnt++; // kb
      }
      closeBufferdOutputStream(os);
      closeBufferdInputStream(is);
    }
    /**
     * ファイルを結合
     * @param outDir 分割されたファイルがあるディレクトリ
     * @throws Exception
     */
    public void concat(String outDir) throws Exception {

      File od = new File(outDir);
      if (!od.exists() || !od.isDirectory()) {
        return;
      }

      String[] fs = od.list(new FilenameFilter(){
                    public boolean accept(File dir, String name) {
                      return name.matches("^.*" + FILE_NAME_REP);
                  }
                  });

      BufferedInputStream  is = null;
      BufferedOutputStream os = null;
      Arrays.sort(fs);
      byte[] buf = new byte[BUF_SIZE_1KB];
      int ret = -1;

      for (int i=0; i<fs.length ;i++) {
        if (os == null) {
          String originalFileName = absolutePath(od, fs[i].replaceFirst(FILE_NAME_REP, ""));
          os = getBufferedOutputStream(originalFileName);
        }
        is = getBufferedInputStream(absolutePath(od, fs[i]));

        while((ret = is.read(buf)) > 0) {
          os.write(buf, 0, ret);
          os.flush();
        }
        os.flush();
        closeBufferdInputStream(is);
      }

      closeBufferdOutputStream(os);
    }

    private String absolutePath(File dir, String filename) {
      return dir.getAbsolutePath() + File.separator + filename;
    }
        private    BufferedInputStream    getBufferedInputStream(String    inputFilename)    throws
FileNotFoundException {
      return new BufferedInputStream(new FileInputStream(new File(inputFilename)));
    }
    private void closeBufferdOutputStream(BufferedOutputStream os) throws Exception {
      if (os != null) {
        os.flush();
        os.close();
      }
    }
    private void closeBufferdInputStream(BufferedInputStream is) throws Exception {
      if (is != null) {
        is.close();
      }
    }
    private BufferedOutputStream getBufferedOutputStream(String fineName) throws Exception {
      File f = new File(fineName);
      f.createNewFile();
      return new BufferedOutputStream(new FileOutputStream(f));
    }
```

```
  private File createDir(String dirName) {
    File od = new File(dirName);
    if (!od.exists()) {
      od.mkdir();
    }
    return od;
  }
  private static void printUsage() {
    StringBuilder buf = new StringBuilder();

    buf.append("java split mode target_file output_dir [output_filesize(kb)]\n");
    buf.append("\tmode:-s split file, -c concatinate files\n");
    buf.append("\ttarget_file:target file to split.\n");
    buf.append("\toutput_dir:target directory for output splitted files.\n");
    buf.append("\toutput_filesize(kb):output file size as KB.\n");

    System.out.println(buf.toString());
  }
}
```