```
package org.apache.catalina.core;
/**
 *
 * @author  nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 25 September 2020

import org.apache.catalina.core.RAMfileArray;
import jakarta.servlet.Ramfile;
import org.apache.catalina.core.RAMStringB64Array;
import jakarta.servlet.RamStringB64;

*
*
* // ============================================================================================
*   //  (THIS IS IN "core") ApplicationContextFacade
*
public RAMfileArray ramfile = new RAMfileArray();
public Ramfile ramfileinterface = ramfile;

   @Override
   public Ramfile getRamFileInstance(){
   return ramfileinterface;
   }


public RAMStringB64Array ramstring64 = new RAMStringB64Array();
public RamStringB64 ramstring64interface = ramstring64;

   @Override
   public RamStringB64 getRamStringB64Instance(){
   return ramstring64interface;
   }
   *
   *
 * // ============================================================================================
*   //  (THIS IS IN "core") ApplicationContext
*
  @Override
  public Ramfile getRamFileInstance(){
  return facade.getRamFileInstance();
  }

  @Override
  public RamStringB64 getRamStringB64Instance(){
  return facade.getRamStringB64Instance();
  }

 *
 *
 * // ============================================================================================
*   //  (THIS IS IN "core" StandardContext.java -  private static class NoPluggabilityServletContext )
*
public Ramfile getRamFileInstance(){
return sc.getRamFileInstance();
}

public RamStringB64 getRamStringB64Instance(){
return sc.getRamStringB64Instance();
}
*
*
 * // ============================================================================================
*   //  THIS IS IN package org.apache.jasper.servlet.JspCServletContext
*
    @Override
    public Ramfile getRamFileInstance() {
    return null;
    }

   @Override
   public RamStringB64 getRamStringB64Instance(){
   return null;
   }
*
*
 * // ============================================================================================
*   //  ServletContext
*
public Ramfile getRamFileInstance();

public RamString64 getRamStringB64Instance();
 */



 // ============================================================================================

import jakarta.servlet.Ramfile;
import java.io.*;
import java.util.regex.*;
import java.lang.Thread;
import java.nio.charset.*;
import java.io.RandomAccessFile;
public class RAMfileArray implements Ramfile{

    // HEADERS https://tools.ietf.org/id/draft-reschke-rfc2183-in-http-02.html

private byte[][] RAMfiles; // the storage array for usage
private long[] ramFilesLength; // array list of the byte length of each file in order
private FileArrayLoader filArrod; // inner subclass Thread
private ArrayLoadingController arrrrLod; // inner subclass Thread
//
private String loaded_files_list = ""; // inventory string list for output
protected String[] loaded_RAM_file_inventory; // loaded_RAM_file_inventory
//
private String[] checkDigitsForConcat = {"0","1","2","3","4","5","6","7","8","9"};
private final String[] Array_Status={"UNASSIGNED_ARRAY","LOADING_ARRAY","FAILED_LOAD","ARRAY_LOADED_READY"};
private String arrayReady = Array_Status[0]; // return object status for the user ready/unready...
//
//public Ramfile stored_bytes; // interface of this class from this instance
//
private File[] flsarr;
private int flslen = 0;
//int terminateFlag=0;
private int preloadertask=0; // file array loading guard  - called in readFileToPreArray(
private boolean Ramfile_ifaceReady=false; // is the context of THIS Ramfile "LOADED ITS ARRAY" and READY;

private long flsarrtotalBytes = 0;

// https://tools.ietf.org/html/rfc6266#page-5
private final String filesep = System.getProperty("file.separator");
private final String syslin = System.getProperty("line.separator");
private String CR_LF=""; // set by constructor
/*
public final String contyp = "Content-Type: ";
public final String conentJPG = "image/jpg"+syslin;
public final String conentJPEG = "image/jpeg"+syslin;
public final String conentTIFF = "image/tiff"+syslin;
public final String conentPNG = "image/png"+syslin;
public final String conentBMP = "image/bmp"+syslin;
```

```
public final String conentGIF = "image/gif"+syslin;
public final String conentBinary = "application/octet-stream"+syslin;
public final String conentNoSniff = "X-Content-Type-Options: nosniff"+syslin;
public final String contentJAR= "application/java-archive"+syslin;
public final String conentBZp = "application/x-bzip"+syslin;
public final String conentBZp2 = "application/x-bzip2"+syslin;
public final String conentGZ = "application/gzip"+syslin;
public final String conentTAR = "application/x-tar"+syslin;
public final String conent7z = "application/x-7z-compressed"+syslin;
public final String conentZIP = "application/zip"+syslin;
//
public final String conentAWebm = "audio/webm"+syslin;
public final String conentAMPEG = "audio/mpeg"+syslin;
public final String conentVWebm = "video/web"+syslin;
public final String conentVMPEG = "video/mpeg"+syslin;
//
public final String conentPDF = "application/pdf"+syslin;
public final String conentMSWord = "application/msword"+syslin;
//
public final String condis = "Content-Disposition: ";
public final String inLine = condis+"inline"+syslin;
public final String attAchment = condis+"attachment"+syslin;
*/

//  HEADER BUILDER CONCAT SYSTEM TO DO


private final String[] headerCharsLegend = {
"0  index symbol  return ( Macintosh OS newline ) character",
"1  index symbol  newline ( UNIX / LINUX newline ) character",
"2  index symbol  carriage return and newline ( Microsoft Corp. Windows OS newline ) ( two chars )",
"3  index symbol  comma symbol",
"4  index symbol  semi colon symbol",
"5  index symbol  colon symbol",
"6  index symbol  single quote mark",
"7  index symbol  double quote mark",
"8  index symbol  dash or minus negative sign",
"9  index symbol  equal sign",
"10 index symbol  less than bracket",
"11 index symbol  greater than bracket",
"12 index symbol  underscore",
"13 index symbol  question mark",
"14 index symbol  ampersand",
"15 index symbol  plus addition sign",
"16 index symbol  dot full stop sentence end sign",
"17 index symbol  left side square bracket symbol",
"18 index symbol  right side square bracket symbol",
"19 index symbol  left side curved parenthesis symbol",
"20 index symbol  right side curved parenthesis symbol",
"21 index symbol  left side curly braces symbol",
"22 index symbol  right side curly braces symbol",
"23 index symbol  SYSTEM OS file separator symbol",
"24 index symbol  text space  ( text break space ) symbol",
"25 index symbol  forward slash stroke",
"26 index symbol  address or At symbol",
"27 index symbol  sharp or number numerated (adnoun) symbol",
"28 index symbol  exclaimation  is not  invertive opposite symbol",
"29 index symbol  numeric percent sign math symbol",
"30 index symbol  dollar symbol",
"31 index symbol  asterix / asterisc denote notation language or computing multiplier math symbol",
"32 index symbol  accent written language  caret written language and computing symbol",
"33 index symbol  pipe operator computing and written language symbol",
};

// (new Character(              ).toString())
private final char[] carr_ret_line = {'\r','\n'}; // CRLF
private final String[] http1011CacheHeaderParts = {
    ((String)new Character('\r').toString()),
        ((String)new Character('\n').toString()),
        ((String)new String(carr_ret_line)),
        ((String)new Character(',').toString()),
        ((String)new Character(';').toString()),
        ((String)new Character(':').toString()),
        ((String)new Character('\'').toString()),
        ((String)new Character('"').toString()),
        ((String)new Character('-').toString()),
        ((String)new Character('=').toString()),
        ((String)new Character('<').toString()),
        ((String)new Character('>').toString()),
        ((String)new Character('_').toString()),
        ((String)new Character('?').toString()),
        ((String)new Character('&').toString()),
        ((String)new Character('+').toString()),
        ((String)new Character('.').toString()),
        ((String)new Character('[').toString()),
        ((String)new Character(']').toString()),
        ((String)new Character('(').toString()),
        ((String)new Character(')').toString()),
        ((String)new Character('{').toString()),
        ((String)new Character('}').toString()),
        ((String)System.getProperty("file.separator")),
        ((String)new Character(' ').toString()),
        ((String)new Character('/').toString()),
        ((String)new Character('@').toString()),
        ((String)new Character('#').toString()),
        ((String)new Character('!').toString()),
        ((String)new Character('%').toString()),
        ((String)new Character('$').toString()),
        ((String)new Character('*').toString()),
        ((String)new Character('^').toString()),
        ((String)new Character('|').toString()),
"Content-Type",
"Content-Range",
"Content-MD5",
"Content-ID",
"Content-Location",
"Content-Length",
"Content-Encoding",
"Content-Language",
"Cache-Control",
"Content-Disposition",
"Content-Description",
"Transfer-Encoding",
"Content-Transfer-Encoding",
"Accept-Charset",
"public",
"private",
"no-cache",
"no-store",
"no-transform",
"must-revalidate",
"proxy-revalidate",
"max-age",
"s-maxage",
"cache-extension",
"min-fresh",
"max-stale",
"inline",
"attachment",
"filename",
```

```
"creation-date",
"modification-date",
"read-date",
"token",
"ext-token",
"If-Match",
"If-Modified-Since",
"Expires",
"ETag",
"expect-params",
"expectation-extension",
"expectation",
"Expect",
"disp-ext",
"chunked",
"compress",
"deflate",
"gzip",
"identity",
"base64",
"q-uoted-printable",
"7bit",
"8bit",
"binary",
"q=0.0",
"q=0.1",
"q=0.2",
"q=0.3",
"q=0.4",
"q=0.5",
"q=0.6",
"q=0.7",
"q=0.8",
"q=0.9",
"immutable",
"stale-while-revalidate",
"stale-if-error",
"only-if-cached",
"Set-Cookie",
"sessionId",
"id",
"SameSite",
"__Host-",
"__Secure-",
"Secure",
"Domain",
"Path",
"Lax",
"None",
"Strict",
"Last-Modified",
"Pragma",
"HttpOnly",
"Cross-Origin-Resource-Policy",
"same-origin",
"cross-origin",
"same-site",
"If-None-Match",
"If-Unmodified-Since",
"If-Range",
"Accept",
"Accept-Encoding",
"Accept-Language",
"Vary",
"Location",
"link",
"rel",
"preconnect",
"Range",
"curl",
"bytes",
};

private int htttpArrpartsLEN; // length of http1011CacheHeaderParts


/**
 *
 */
public RAMfileArray(){
    setLineCRLF(syslin);
    htttpArrpartsLEN = http1011CacheHeaderParts.length;
} // CONSTRUCTOR


public synchronized String getHttpPartsArrayElement(int idxsymb){
String syntxout="Error - Array index out of bounds - must be [ 0 - "+(htttpArrpartsLEN - 1)+" ]";
if((idxsymb > -1)&&(idxsymb < htttpArrpartsLEN)){
syntxout=(String)""+(http1011CacheHeaderParts[idxsymb]);
}
return (String)syntxout;
}//enmeth


// NEW FOR USER API
public String getHttpPartsFullArrayLegend(boolean line){
String symbol_Legend_Msg = "Documentation List of array indexes Cache Control Commands an Header Syntax Symbols - http 1.1 / 1.0 header syntax grammar";
String atch=" <p>"+CR_LF;
if(line==false){
atch=CR_LF+CR_LF;
}
String outputLeg="";
if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){
outputLeg="WARNING RAMFILE ARRAY UNINSTANTIATED"+atch;
}
outputLeg=atch+symbol_Legend_Msg+atch;
    for(int su=0;(su < http1011CacheHeaderParts.length);su++){
    outputLeg+=(" Index "+su+":  ")+(http1011CacheHeaderParts[su]+atch);
    }
return(String)outputLeg;
}//enmeth


public String getHttpPartsSymbolArrayLegend(boolean line){
String symbol_Legend_Msg = "(Short - first 33 indexes) Documentation List of array indexes for header syntax grammar Symbols";
String atch=" <p>"+CR_LF;
if(line==false){
atch=CR_LF+CR_LF;
}
String outputLeg="";
if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){
outputLeg="WARNING RAMFILE ARRAY UNINSTANTIATED"+atch;
}
outputLeg=atch+symbol_Legend_Msg+atch;
    for(int su=0;(su < headerCharsLegend.length);su++){
    outputLeg+=(headerCharsLegend[su]+atch);
    }
return(String)outputLeg;
}//enmeth
```

```
// time for most is for expires header GMT   java.time.LocalDateTime for plus minus minutes seconds hours

public synchronized java.time.LocalDateTime systenTimeNow(){
return (java.time.LocalDateTime)java.time.LocalDateTime.now();
}


public synchronized java.util.Date obtainHeaderDate(java.time.LocalDateTime timeset){
// Date(int year, int month, int date, int hrs, int min, int sec)
java.util.Date adjusted = new java.util.Date(timeset.getYear(),(timeset.getMonthValue()-1),timeset.getDayOfMonth(),timeset.getHour(),timeset.getMinute(),timeset.getSecond());
return (java.util.Date)adjusted;
}

public synchronized String getCreatedGMTDate(java.util.Date headerGMTdate){
return (String)headerGMTdate.toGMTString();
}

public synchronized java.time.LocalDateTime plusTimeDays(java.time.LocalDateTime adjt,int amo){
java.time.LocalDateTime adjt2 = adjt.plus(((java.time.temporal.TemporalAmount)java.time.Period.ofDays(amo)));
return (java.time.LocalDateTime)adjt2;
}

public synchronized java.time.LocalDateTime plusTimeWeeks(java.time.LocalDateTime adjt,int amo){
java.time.LocalDateTime adjt2 = adjt.plus(((java.time.temporal.TemporalAmount)java.time.Period.ofWeeks(amo)));
return (java.time.LocalDateTime)adjt2;
}

public synchronized java.time.LocalDateTime plusTimeMonths(java.time.LocalDateTime adjt,int amo){
java.time.LocalDateTime adjt2 = adjt.plus(((java.time.temporal.TemporalAmount)java.time.Period.ofMonths(amo)));
return (java.time.LocalDateTime)adjt2;
}

public synchronized java.time.LocalDateTime minusTimeDays(java.time.LocalDateTime adjt,int amo){
java.time.LocalDateTime adjt2 = adjt.minus(((java.time.temporal.TemporalAmount)java.time.Period.ofDays(amo)));
return (java.time.LocalDateTime)adjt2;
}

public synchronized java.time.LocalDateTime minusTimeWeeks(java.time.LocalDateTime adjt,int amo){
java.time.LocalDateTime adjt2 = adjt.minus(((java.time.temporal.TemporalAmount)java.time.Period.ofWeeks(amo)));
return (java.time.LocalDateTime)adjt2;
}

public synchronized java.time.LocalDateTime minusTimeMonths(java.time.LocalDateTime adjt,int amo){
java.time.LocalDateTime adjt2 = adjt.minus(((java.time.temporal.TemporalAmount)java.time.Period.ofMonths(amo)));
return (java.time.LocalDateTime)adjt2;
}

//  supplied to USER API
public int getTotalLoadedFiles(){
return (int)flslen;
}


private long garbageCalibrateSizeTotal(){
for(int ie=0;ie<flslen;ie++){
flsarrtotalBytes+=ramFilesLength[ie];
}
flsarrtotalBytes = (flsarrtotalBytes*2);
return flsarrtotalBytes;
}



private long getFreememory(){
Runtime rmtn = Runtime.getRuntime();
return ((long)rmtn.freeMemory());
}


// must be void
private void triggerGarbageRequest(){
if(flsarrtotalBytes > ((long)getFreememory())){
System.gc();
}
}

// shielded
private boolean setIsInterfaceInstanceReady(){
    if(arrayReady.indexOf("ARRAY_LOADED_READY")!=-1){ // "FAILED_LOAD"
  Ramfile_ifaceReady=true;
    }else{
  Ramfile_ifaceReady=false;
    }
return Ramfile_ifaceReady;
}

// supplied to User API interface
public boolean getIsInterfaceInstanceReady(){
return (boolean)Ramfile_ifaceReady;
}


public String getInterfaceReadyFlag(){
return (String)arrayReady; // 0 "UNASSIGNED_ARRAY"     1 "LOADING_ARRAY"      2 "FAILED_LOAD"     3 "ARRAY_LOADED_READY"
}

private String setInterfaceReadyFlag(int rdy,String debug){
arrayReady = Array_Status[rdy]+=debug;
return arrayReady;
}

// shielded - server class use only
private boolean interfaceLoadedReady(boolean boo){
    Ramfile_ifaceReady=boo;
return Ramfile_ifaceReady;
}


private String testSpaceToken=getHttpPartsArrayElement(24);  // GLOBAL


private String setTestSpaceChar(String spaceSymbolChar){
testSpaceToken=(String)spaceSymbolChar;
return testSpaceToken;
}//enmeth


public void setTestSpaceToken(String spaceSymbolChar){
setTestSpaceChar(spaceSymbolChar);
}//enmeth

public String getTestSpaceToken(){
return (String)testSpaceToken;
}


private String[] setDigitCharCheckArray(String[] chario){
checkDigitsForConcat=(String[])chario;
    return checkDigitsForConcat;
```

```
}//enmeth


// for user to change the digits 0 - 9
public void setDigitISOTestArray(String[] iso_Digit_charArray){
setDigitCharCheckArray(iso_Digit_charArray);
}
private String[] getDigitISOTestArray09(){
 return (String[])checkDigitsForConcat;
}

public String[] getDigitISOTestArray(){
return (String[])getDigitISOTestArray09();
}

private synchronized boolean checkOnSingleChar(String partstr){
boolean chko = false;
// GLOBAL final String[] checkDigitsForConcat = {"0","1","2","3","4","5","6","7","8","9",};
    for(int nu=0;(nu < 10);nu++){
        if(partstr.indexOf(((String)checkDigitsForConcat[nu])) != -1){
    chko = true;
    nu=10;
            }
    }//enfr
return (boolean)chko;
}//enmeth

private synchronized boolean checkSubstrSet(String[] subparts){ // (one  single index) of strings[] set  of the SET of strings as an array of chars
boolean output=true;
int arle2 = subparts.length;
for(int bsec = 0; (bsec < arle2) ; bsec++){
output = checkOnSingleChar(subparts[bsec]);
if(output==false){
bsec = arle2;
}//enif
}//enfr
return(boolean)output;
}

private synchronized String concatHttpArrayCustomParts(String[]  http1011CacheHeaderelements){
/// check for digits only or a single space followed by digits only
boolean checkout = true;
int arle = http1011CacheHeaderelements.length;
int idexon = 0;
String prepspac = "";
String strout=" ERROR IN concatHttpArrayParts(String[]  http1011CacheHeaderelements) INPUT - USE CLAUSE";
String[] prepspcContent;
//
for(int sec = 0; (sec < arle) ; sec++){
prepspac = http1011CacheHeaderelements[sec].substring(0,1);
if(prepspac.indexOf(testSpaceToken) == -1){ // NO SPACE IT SHOULD HAVE DIGITS
prepspcContent = http1011CacheHeaderelements[sec].split("");
// tomcat10-stdout.2020-09-25.log
//System.out.print("prepspcContent: LENGTH:"+prepspcContent.length+"   prepspac: "+prepspac+CR_LF+CR_LF);
for(int kj=0;kj<prepspcContent.length;kj++){
//System.out.print("prepspcContent: val:  "+prepspcContent[kj]+CR_LF+CR_LF);

}

checkout = checkSubstrSet(prepspcContent);
if(checkout == false){
strout+=" SUMMARY-ERROR-INFORMATION-ONLY TOKEN-STRING-LENGTH[ "+prepspac+" ] PROBABLY MATCHES SPECIAL LITERAL TOKEN STRING-LENGTH[ IS ALWAYS ONE STRING CHARACTER ]";
sec=arle;
}else{
    http1011CacheHeaderelements[sec].trim();
idexon = (int)new Integer(http1011CacheHeaderelements[sec]).intValue();
http1011CacheHeaderelements[sec] = http1011CacheHeaderParts[idexon];
}//enif
}else{
http1011CacheHeaderelements[sec] = http1011CacheHeaderelements[sec].substring(1);
}//enif

}//enfr

if(checkout==true){
int strt = 0;
strout="";
while(strt < arle){
strout+=http1011CacheHeaderelements[strt];
strt++;
}//enwhl
}//enif


return (String)strout;
}//enmwth


public synchronized String concatHttpArrayParts(String[]  http1011CacheHeaderelements){
String sendout = concatHttpArrayCustomParts(http1011CacheHeaderelements);
return (String)sendout;
}//enmwth


private synchronized String concatHttpArrayCustomParts(int[]  http1011CacheHeaderelements){
String joinr = "";
int arlenth = http1011CacheHeaderelements.length;
int oops=0;
if(arlenth > 1){
for(int tutut = 0; tutut < arlenth; tutut++){
    oops = http1011CacheHeaderelements[tutut];
joinr += http1011CacheHeaderParts[oops];
}//enfr
}else{
joinr = "ERROR int[] method joiner clause - requires at least TWO INDEXS AS INPUT ARGUMENT for http1011CacheHeaderParts[] USING concatHttpArrayCustomParts(int[]  http1011CacheHeaderelements)";
}//enifels
return (String)joinr;
}


public synchronized String concatHttpArrayParts(int[]  http1011CacheHeaderelements){
String sendout = (String)concatHttpArrayCustomParts(http1011CacheHeaderelements);
    return (String)sendout;
}//enmwth




/*
java.nio.charset.Charset defchar = java.nio.charset.Charset.defaultCharset();
String charsetInputName = defchar.name(); // get default OS system underlying base charset code page
//  "UTF_8";  "GB18030"  //java.nio.charset.StandardCharsets.UTF_8.name(); // "GB2312";  ///  "UTF_16";  // "UTF_8";  "GB18030"
FOLLOWING is middle hop transfer reencode from ("GB18030") PRC Chinese simplified carrying an English subset to a (GB2312) Web Cinese simplified with English subset
THEN from the middle hop (GB2312) web charset to a standard English charset (UTF_8)
String charsetFrom = "GB2312";
String charsetNameto = java.nio.charset.StandardCharsets.UTF_8.name(); //"GB2312"; US_ASCII   UTF_8
String charout = reEncodechars(java.util.Locale.ENGLISH,headerOrigin,"GB18030","GB2312");
charout = reEncodechars(java.util.Locale.ENGLISH,charout,charsetFrom,charsetNameto);
*/
private synchronized java.nio.charset.CharsetDecoder setCharsetDecoder(String Iname){
```

```
return (java.nio.charset.Charset.forName(Iname)).newDecoder();
}

private synchronized java.nio.charset.CharsetEncoder setCharsetEncoder(String Outname){
return (java.nio.charset.Charset.forName(Outname)).newEncoder();
}

private synchronized java.nio.CharBuffer getCharInBuffer(java.util.Locale locum,String headerOrigin,String charsetInputName)throws UnsupportedEncodingException,CharacterCodingException{
headerOrigin = (String)String.format(locum,"%s",headerOrigin);    // locale SVC served output control (is some help)
java.nio.charset.CharsetDecoder ches = setCharsetDecoder(charsetInputName);
ches.onMalformedInput(java.nio.charset.CodingErrorAction.IGNORE);  // IGNORE cannot do much inside a server
ches.onUnmappableCharacter(java.nio.charset.CodingErrorAction.IGNORE);
 return (java.nio.CharBuffer) ches.decode(((java.nio.ByteBuffer)java.nio.ByteBuffer.wrap(headerOrigin.getBytes(charsetInputName))));
}//enmeth

private synchronized String inChars(java.util.Locale locum,String headerOrigin,String charsetInputName,String charsetnameto){
 String output="";  //reEncodechars() "+arrayReady;
 try{
java.nio.charset.CharsetEncoder chunda = setCharsetEncoder(charsetnameto);
chunda.onMalformedInput(java.nio.charset.CodingErrorAction.IGNORE);
chunda.onUnmappableCharacter(java.nio.charset.CodingErrorAction.IGNORE);
output = new String(((byte[])((java.nio.ByteBuffer)chunda.encode(((java.nio.CharBuffer)getCharInBuffer(locum,headerOrigin,charsetInputName))).array()),charsetnameto);
 output = (String)String.format(locum,"%s",output);  // locale SVC served output control (is some help)
 }catch (CharacterCodingException ex) {
ex.printStackTrace();
    }catch(java.io.UnsupportedEncodingException unsuppode){
unsuppode.printStackTrace();
    }
//
return ((String)output);
}//enmeth


// supplied to User API interface -- WARNING must have the encoding installed available
// ---   OLD public synchronized String reEncodechars(String headerOrigin,String charsetnameto){
public synchronized String reEncodechars(java.util.Locale locum,String headerOrigin,String charsetInputName,String charsetnameto){
String retrieve = "";
String retrieve2 = "";
int lenorig = headerOrigin.length();
String sti = "";
for(int xz = 0; xz < lenorig ; xz++){
sti = headerOrigin.substring(xz,(xz+1));
retrieve += inChars(locum,sti,charsetInputName,charsetnameto);
}//enfr

    try {
        byte[] retr2 = retrieve.getBytes(charsetnameto);
        retrieve2 = new String(retr2,((Charset)Charset.forName(charsetnameto)));
    } catch (UnsupportedEncodingException ex) {
        ex.printStackTrace();
    }
return ((String)String.format(locum,"%s",retrieve2));
}//enmeth


// to load by "File object"  to load by text String   --- UTFdDECODER , LOCALE ISO TO WRITE INTO STRINGS SECTIONS FOR PLATFORM - WEB
/**
 * supplied to User API interface
 * @param setlist
 * @throws FileNotFoundException
 * @throws IOException
 */
public void loadPreLoadingList(File setlist){ // servlet parsed in list-of-names--single-file object
//
 if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
setInterfaceReadyFlag(0,""); // loading flag
byte[] listed;
long flen = setlist.length();


if(setlist.exists()){
try{
RandomAccessFile tasklist = new RandomAccessFile(setlist,"r");
listed = new byte[new Long(flen).intValue()];
tasklist.read(listed,0,new Long(flen).intValue());
tasklist.close();

String listset = new String(listed);  //byte array to string
loadPreLoadingList(listset);
}catch(FileNotFoundException fino){
    fino.printStackTrace();
interfaceLoadedReady(false);
    setInterfaceReadyFlag(2,"_loadinglist-txt-file"); // loading flag
}catch(IOException seeno){
    seeno.printStackTrace();
interfaceLoadedReady(false);

    setInterfaceReadyFlag(2,"_loadinglist-txt-file"); // loading flag
}
}else{
   // new Throwable required  .printStackTrace();
    interfaceLoadedReady(false);
setInterfaceReadyFlag(2,"_loadinglist-txt-file loadPreLoadingList(File) File did not exist"); // loading flag
}
 }//enif ready
}//enmeth


// to load by text String   --- UTFdDECODER , LOCALE ISO TO WRITE INTO STRINGS SECTIONS FOR PLATFORM - WEB
/**
 * supplied to User API interface
 * @param listset
 * @throws java.io.IOException
 */
public void loadPreLoadingList(String listset){ // String input version whether web FORM or TXT-FILE output parsed in
 if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
    setInterfaceReadyFlag(0,"_"); // loading flag
    String[] listsetArr; // array for filenames as strings
String cr=http1011CacheHeaderParts[3]; // if the file string is a comma separated list ,
if(listset.indexOf(http1011CacheHeaderParts[6])!=-1){
Pattern.compile(http1011CacheHeaderParts[6]).matcher(listset).replaceAll("");
}
if(listset.indexOf(http1011CacheHeaderParts[7])!=-1){
Pattern.compile(http1011CacheHeaderParts[7]).matcher(listset).replaceAll("");
}
if(listset.indexOf(cr)!=-1){
listsetArr = (Pattern.compile(cr)).split(listset);
}else if(listset.indexOf(http1011CacheHeaderParts[2])!=-1){
    cr=http1011CacheHeaderParts[2];
listsetArr = (Pattern.compile(http1011CacheHeaderParts[2])).split(listset);
}else if(listset.indexOf(http1011CacheHeaderParts[1])!=-1){
    cr=http1011CacheHeaderParts[1];
listsetArr = (Pattern.compile(http1011CacheHeaderParts[1])).split(listset);
}else if(listset.indexOf(http1011CacheHeaderParts[0])!=-1){
    cr=http1011CacheHeaderParts[0];
listsetArr = (Pattern.compile(http1011CacheHeaderParts[0])).split(listset);
}else{
    cr=syslin;
 listsetArr = (Pattern.compile(syslin)).split(listset); // if there is a chance anything is different
}
```

```
//  if the array for filenames as strings is not instantiated now then an error has occurred - cease
if(listsetArr!=null){
loadPreLoadingList(listsetArr);
}else{
interfaceLoadedReady(false);
setInterfaceReadyFlag(2,"_loadPreLoadingList(String)"); // loading flag   0"UNASSIGNED_ARRAY",   1"LOADING_ARRAY",    2"FAILED_LOAD",     3"ARRAY_LOADED_READY"
}
 }//enif ready
}//enmeth


// shielded - server class use only
private String setLineCRLFIn(String CR_LFin){
return CR_LF;
}

// userr interface supplied
public void setLineCRLF(String CR_LFin){ // oddly it were temp changed for some reason
setLineCRLFIn(CR_LFin);
}

// shielded - server class use only
private String getLineCRLFOut(){
return (String)CR_LF;
}


// userr interface supplied
public String getLineCRLF(){ // oddly it were temp changed for some reason
String crlf = (String) getLineCRLFOut();
return (String)crlf;
}


/**
 * supplied to User API interface
 * @param fls
 * @throws FileNotFoundException
 * @throws IOException
 */
public void loadPreLoadingList(String[] fls){
     if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
setInterfaceReadyFlag(1,"_loadPreLoadingList(String[])");
 if(fls.length > 0){
    loadInventoryFileList(fls);
    int fileLen = fls.length;
File[] firay = new File[fileLen];
//
for(int cfl=0;cfl<fileLen;cfl++){
firay[cfl] = new File((fls[cfl].trim()));

}//enfr

loadPreLoadingList(firay);
 }else{
     interfaceLoadedReady(false);
   setInterfaceReadyFlag(2,"_loadinglist(String[] fls)");  // 0 "UNASSIGNED_ARRAY"     1 "LOADING_ARRAY"      2 "FAILED_LOAD"     3 "ARRAY_LOADED_READY"
 }
     }//enif ready
}//enmeth


/**
 * supplied to User API interface
 * @param fistset
 * @throws IOException
 */
public void loadPreLoadingList(File[] fistset){
     if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
 setInterfaceReadyFlag(0,"_");
    try{
   if(fistset.length > 0){
    loadFilesAndHeaders(fistset);
   }else{
   setInterfaceReadyFlag(2,"_loadinglist-arrasy-file - loadPreLoadingList(File[])");  // 0 "UNASSIGNED_ARRAY"     1 "LOADING_ARRAY"      2 "FAILED_LOAD"     3 "ARRAY_LOADED_READY"
   }
    }catch(IOException ionode){

    }
     }//enif ready
}//enmeth

/**
 * shielded - server class use only
 * @param loaded_RAM_file_inventory
 * @return
 */
private String[] loadInventoryFileList(String[] loaded_inventory){ // load full file names to global for reference
loaded_RAM_file_inventory=(String[])loaded_inventory;
    return loaded_RAM_file_inventory;
}



// **** *** ***   ***** *** *****    *****
// rewiring to global variable and remove passed references TO DO
private File[] fileset(File[] flis){
flsarr=(File[])flis;
    return flsarr;
}


/**
 * shielded - server class use only
 * @param flis
 * @throws java.io.IOException
 */

private void loadFilesAndHeaders(File[] flis) throws java.io.IOException{
fileset(flis);
setInterfaceReadyFlag(0,"_"); // loading flag
resetRamFTaskCount(); // convenient location though already set
fileLen(flis.length);
createFileByteLengthArray(flslen); // create the stored File byte lengths array for each byte array to be instantiated (long)
// call the thread to load files here
runthreadFileLoad(flis);

}//enmeth


/**
 * shielded - server class use only
 * @param lenf
 * @return
 */
private int fileLen(int lenf){
flslen=lenf;
    return flslen;
}
```

```
/**
 * shielded - server class use only
 * @param farr
 * @return
 * @throws FileNotFoundException
 * @throws java.io.IOException
 */
private void instanceFileArrayLoader(File[] farr)throws FileNotFoundException,java.io.IOException{
(filArrod = new FileArrayLoader(farr)).start(); // NOTE PROBABLY SAFEST TO BE 10 PRIORITY
 //return filArrod;
}

// shielded - server class use only
private ArrayLoadingController instanceArrayLoader(File[] farr)throws FileNotFoundException,java.io.IOException{
arrrrLod = new ArrayLoadingController(farr); // NOTE PROBABLY SAFEST TO BE 10 PRIORITY
 return arrrrLod;
}

// shielded - server class use only
private void initRamfileArray(){
    RAMfiles = new byte[flslen][];
}

/**
 * shielded - server class use only
 * @param farr
 * @throws FileNotFoundException
 * @throws java.io.IOException
 */
// SPECIAL PURPOSE VOID - WITH BASIC INSTANTIATION OF MAIN ARRAY INDEXES
private void runthreadFileLoad(File[] farr)throws FileNotFoundException,java.io.IOException{
setInterfaceReadyFlag(1,"_"); // loading flag
    if(preloadertask == 0){ // if as it should be
instanceArrayLoader(farr);
arrrrLod.start();
}
}//enmeth


/**
 * shielded - server class use only
 * @param felen
 * @return
 */
private long[] createFileByteLengthArray(int felen){ // create array - stores lengths - bytes of file byte array global array indexed
ramFilesLength = new long[felen];
for(int ni=0;ni<felen;ni++){
ramFilesLength[ni]=0;
}
return ramFilesLength; // global
}//enmeth


/**
 * shielded - server class use only
 * @param len
 * @param idx
 * @return
 */
private long setRamlengths(long len,int idx){ // set each corresponding indexes of ramFilesLength array to ofRAMfile array - byte lengths
ramFilesLength[idx] = len;
return ramFilesLength[idx];  //  global
}// enmeth


// supplied to User API interface
public synchronized long getOutWriteRAMbyteLength(int arrLenIdX){
long retrivLen=-1;  // returns -1 if out of bounds
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(arrLenIdX > -1)&&(arrLenIdX < ramFilesLength.length)){
    if((arrLenIdX > ((int)-1)) && (arrLenIdX < ramFilesLength.length)){
retrivLen=ramFilesLength[arrLenIdX];
}else{
retrivLen=-1;
}
}//enchk
    return (long)retrivLen;
}



/**
 * shielded - server class use only
 * @return
 */
private int resetRamFTaskCount(){
preloadertask=0;
return preloadertask;// global
}


/**
 * shielded - server class use only
 * @return
 */
private int preTaskerCount(){
preloadertask++;
return preloadertask;// global
}//enmeth


/**
 * supplied to User API interface
 * @param idxAr
 * @param indByt
 * @return
 */
public synchronized byte[] getRAMByteArray(int idxAr){
    triggerGarbageRequest();
return (byte[])RAMfiles[idxAr];
}//enmeth



 // shielded - server class use only
private String getLoadedRAMFilesList(boolean HtmlTXT){
  String linesepSYS =  "<p>"+syslin; // true
    if(HtmlTXT == false){
linesepSYS =  syslin+syslin;
    }
 if(arrayReady.indexOf("ARRAY_LOADED_READY")!=-1){
    int numf = loaded_RAM_file_inventory.length;
    loaded_files_list = "List of files loaded to ram array:  "+linesepSYS;
for(int cnum=0;cnum<numf;cnum++){
loaded_files_list+=("array-index:  "+cnum+"   filename:  "+loaded_RAM_file_inventory[cnum]+"   bytes: "+getOutWriteRAMbyteLength(cnum)+linesepSYS);
}
 }//enchk
return (String)loaded_files_list;
}
```

```
/**
 * supplied to User API interface
 * @return
 */
public String getLoadedRAMFileNamesInventory(boolean HtmlTXT){ // TRUE = HTML LINES            FALSE = TEXT-CRLF
String qjump = getLoadedRAMFilesList(HtmlTXT);
    return (String)qjump;
}

    /**
     * supplied to User API interface
     * @param idxName
     * @return
     */

public synchronized String getLoadedFileNameAtIndex(int idxName){
String fipth="FILENAME_ARRAY_UNAVAILABLE   or ERROR array index out of bounds  "+arrayReady;
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(idxName > -1)&&(idxName < flsarr.length)){ //="FILENAME_ARRAY_UNAVAILABLE";
fipth = ((String)flsarr[idxName].getName());
}
    return ((String)fipth);
}

//  supplied to User API interface
public synchronized String getLoadedCanonicalFileNameAtIndex(int idxName){
String fipth="FILENAME_ARRAY_UNAVAILABLE   or ERROR array index out of bounds  "+arrayReady;
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(idxName > -1)&&(idxName < flsarr.length)){ //="FILENAME_ARRAY_UNAVAILABLE";
    try{
fipth = ((String)flsarr[idxName].getCanonicalPath());
    }catch(IOException fnout){
            fnout.printStackTrace();
            setInterfaceReadyFlag(2,"_getLoadedCanonicalFileNameAtIndex() IOE  "); //global String arrayReady
        }
}
    return ((String)fipth);
}

//  supplied to User API interface
public String pingMessageTest(){
String message = "UNAVAILABLE    "+arrayReady;
if(arrayReady.indexOf("ARRAY_LOADED_READY")!=-1){ //
message = "Version 1 org.apache.catalina.core.RAMfileArray class and interface jakarta.servlet.Ramfile 23rd September 2020 ( class to load and store B64 Strings readyhand on reference for IMG html taags or MIME attachments ) -  pingMessag
}
return (String)message;
}

private byte[] fileSetter(int elemindx,int lenf,byte[] data){
RAMfiles[elemindx]=new byte[lenf];
for(int sr=0;(sr < lenf);sr++){
RAMfiles[elemindx][sr] = data[sr];
}
data=null;
return RAMfiles[elemindx];
}//enmeth


private void fileLoader(File[] fis,File fii){
// load the files to RAM on arrays one by one - global not two by two weave pole
Long lenny = new Long(ramFilesLength[preloadertask]);
int promint = (new Long(lenny).intValue());
    byte[] bty = new byte[promint];
//  RAMfiles[preloadertask] = new byte[(lenny.intValue())]; // instantiate a byte array ready for file bytes
try{
RandomAccessFile f = new RandomAccessFile(fii,"r");
    f.readFully(bty,0,promint);
    fileSetter(preloadertask,promint,bty);
f.close();
}catch(FileNotFoundException fnofl){
    fnofl.printStackTrace();
        setInterfaceReadyFlag(2,"_fileLoader()  FNF  fileLoader(File[] fis,File fii)"); //
        }catch(IOException fnout){
            fnout.printStackTrace();
        setInterfaceReadyFlag(2,"_fileLoader() IOE  fileLoader(File[] fis,File fii)"); //
        }

// WARNING - SET THE ARRAY DATA ON THE ELEMENT BEFORE INCREMENT OR GARBAGE CLEANUP

if((preloadertask != (flslen-1)) && (preloadertask < flslen)){
arrrrLod.reInstanceLoad(fis);
}else{
garbageCalibrateSizeTotal();
resetRamFTaskCount();
       System.gc();
}
//
//return RAMfiles[preloadertask]; // finalised load of the prepared index onto global access storage array
}//enmeth

// ALL NESTED CLASSES AND CONTENTS ARE * shielded - server class use only ================================

private class ArrayLoadingController extends Thread{
    File[] fis;

private ArrayLoadingController(File[] fis)throws FileNotFoundException,java.io.IOException{
this.fis=fis;
this.setPriority(10); // thread priority 1 - 10
long leng =0;

int lodflgInv = 0; // controls whether the loading on the STRING LOGGING INVENTORY array takes place after the loop
if(loaded_RAM_file_inventory==null){ // BECAUSE OF THIS THERE MUST BE A RESET METHOD TO RELOAD TO CLEAR THE ARRAY  AND FLAG TO KNOW RELOADING HAS BEEN REQUESTED !!!!
    lodflgInv = 1;
}
// get all file-byte array lengths and set to storage on the long array - global
String[] loaded_inventory = new String[fis.length]; // local - REQUIRES INIT ASSURANCE
for(int ixa=0;ixa < flslen;ixa++){
setRamlengths((fis[ixa].length()),ixa); // load the file-length-size of each file onto an array
if(lodflgInv == 1){
loaded_inventory[ixa] = fis[ixa].getCanonicalPath(); // put the filename onto a string array element (name only)  loaded_RAM_file_inventory
}
}//enfor
if(lodflgInv == 1){
loadInventoryFileList(loaded_inventory); // set global array of filenames        ----------   MAY NEED ARRAYCOPY
}

}//enconstr

public void run(){

   if(preloadertask==0){
initRamfileArray();

        try{
instanceFileArrayLoader(fis);
        }catch(FileNotFoundException fnofl){
            fnofl.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-run"); // LOADED
        }catch(IOException fnout){
            fnout.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-run"); // LOADED
        }
```

```
    }

}//enrun


private void reInstanceLoad(File[] fis){
preTaskerCount(); // ++ ready the NOTATION on its array for next file index in superclass for file loading
 try{
if(preloadertask < flslen){
instanceFileArrayLoader(fis); // ready the next++ NEW Thread to load a file and byte array (calling this)
   if(preloadertask==(flslen-1)){ // last file loaded
       setInterfaceReadyFlag(3,""); // LOADED
       if(setIsInterfaceInstanceReady()){
interfaceLoadedReady(true); // allow user test flag to be set to true
loaded_files_list = getLoadedRAMFileNamesInventory(true);
       }
      }//enif-last-file
// call ready
}
}catch(FileNotFoundException fnofl){
    fnofl.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-reInstanceLoad"); //
        }catch(IOException fnout){
            fnout.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-reInstanceLoad"); //
        }

}//enmeth

}//enclss


/*
ALL NESTED CLASSES AND CONTENTS ARE * shielded - server class use only ================================
*/

private class FileArrayLoader extends Thread{
//ramary.
   File[] fis;

protected FileArrayLoader(File[] fis){
this.fis=fis;
this.setPriority(10); // thread priority 1 - 10
}//enconstr

// THREADS REQUIRE WHETHER CLASS OR RUNNABLE TO OVERIDE RUN (BECAUSE START AND STOP MUST BE GOVERNED UNLIKE CALLING FROM CONSTRUCTOR OR USING A TIMER)
public void run(){
if(preloadertask < flslen){
fileLoader(fis,fis[preloadertask]); // essential start of file and array index loading sequence
}
}//enrun

}//ensubclss

}//enclss







//=============================================================================
```

```
//package javax.servlet;
//package org.apache.catalina.core;
package jakarta.servlet;
import java.io.*;
import jakarta.servlet.*;
//import javax.servlet.Ramfile;
import org.apache.catalina.core.RAMfileArray;
/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 25 September 2020
 *
 * uses the API coded javax.servlet.ServletContext THROUGH extension and modification of ApplicationContextFacade.java CLASS
 */
public interface Ramfile{

    // !!!   loadFilesAndHeaders    DO NOT ACCESS THIS METHOD FROM INTERFACE

// !!!   DO NOT ACCESS ANY METHODS THAT OPERATE DIRECTLY IN OR THROUGH THE SUBCLASSES
public String getHttpPartsArrayElement(int idxsymb);

public String getHttpPartsFullArrayLegend(boolean line);

public String getHttpPartsSymbolArrayLegend(boolean line);

public java.time.LocalDateTime systenTimeNow();

public java.util.Date obtainHeaderDate(java.time.LocalDateTime timeset);

public String getCreatedGMTDate(java.util.Date headerGMTdate);

public java.time.LocalDateTime plusTimeDays(java.time.LocalDateTime adjt,int amo);

public java.time.LocalDateTime plusTimeWeeks(java.time.LocalDateTime adjt,int amo);

public java.time.LocalDateTime plusTimeMonths(java.time.LocalDateTime adjt,int amo);

public java.time.LocalDateTime minusTimeDays(java.time.LocalDateTime adjt,int amo);

public java.time.LocalDateTime minusTimeWeeks(java.time.LocalDateTime adjt,int amo);

public java.time.LocalDateTime minusTimeMonths(java.time.LocalDateTime adjt,int amo);


public String concatHttpArrayParts(String[] http1011CacheHeaderelements);

public String concatHttpArrayParts(int[] http1011CacheHeaderelements);

public void setDigitISOTestArray(String[] iso_Digit_charArray);

public void setTestSpaceToken(String spaceSymbolChar);

public String getTestSpaceToken();

public boolean getIsInterfaceInstanceReady();

public String getInterfaceReadyFlag(); // required to know if some subtle failure has occurred or the process is not yet completed

public String getLoadedFileNameAtIndex(int idxName);
```

```
public String getLoadedCanonicalFileNameAtIndex(int idxName);

public void setLineCRLF(String CR_LF);

public String getLineCRLF();

public int getTotalLoadedFiles();

public String[] getDigitISOTestArray();

public String reEncodechars(java.util.Locale locum,String headerOrigin,String charsetInputName,String charsetnameto);

public void loadPreLoadingList(File setlist); // throws FileNotFoundException, IOException;

public void loadPreLoadingList(String listset); // throws java.io.IOException

public void loadPreLoadingList(String[] fls); // throws FileNotFoundException,IOException

public void loadPreLoadingList(File[] fistset); // throws IOException

public long getOutWriteRAMbyteLength(int arrLenIdX); // supplies the length of the stored file bytes array from index

public byte[] getRAMByteArray(int idxAr);

public String getLoadedRAMFileNamesInventory(boolean HtmlTXT);  // user test debug

public String pingMessageTest();  // user test debug


}//enface




//============================================================================
```

```
package org.apache.catalina.core;
/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 25 September 2020
 */

import jakarta.servlet.RamStringB64;
import java.io.*;
import java.util.regex.*;
import java.lang.Thread;
import java.nio.charset.*;
import java.io.RandomAccessFile;
import org.apache.tomcat.util.codec.binary.Base64;
// org\apache tomcat util codec binary Base64.java
public class RAMStringB64Array implements RamStringB64{  // org.apache.catalina.core.RAMStringB64Array

    // HEADERS https://tools.ietf.org/id/draft-reschke-rfc2183-in-http-02.html
private boolean urlsafe=false;
private String[] RAMfiles; // the storage array for usage
private long[] ramFilesLength; // array list of the byte length of each file in order
private FileArrayLoader filArrod; // inner subclass Thread
private ArrayLoadingController arrrrLod; // inner subclass Thread
//
private String loaded_files_list = ""; // inventory string list for output
protected String[] loaded_RAM_file_inventory; // loaded_RAM_file_inventory
//
//private String[] checkDigitsForConcat = {"0","1","2","3","4","5","6","7","8","9"};
private final String[] Array_Status={"UNASSIGNED_ARRAY","LOADING_ARRAY","FAILED_LOAD","ARRAY_LOADED_READY"};
private String arrayReady = Array_Status[0]; // return object status for the user ready/unready...
//
//public Ramfile stored_bytes; // interface of this class from this instance
//
private File[] flsarr;
private int flslen = 0;
//int terminateFlag=0;
private int preloadertask=0; // file array loading guard  - called in readFileToPreArray(
private boolean Ramfile_ifaceReady=false; // is the context of THIS Ramfile "LOADED ITS ARRAY" and READY;

private long flsarrtotalBytes = 0;

// https://tools.ietf.org/html/rfc6266#page-5
private final String filesep = System.getProperty("file.separator");
private final String syslin = System.getProperty("line.separator");
private String CR_LF=""; // set by constructor
private final char[] carr_ret_line = {'\r','\n'}; // CRLF
private final String[] http1011CacheHeaderParts = {
((String)new Character('\r').toString()),
((String)new Character('\n').toString()),
((String)new String(carr_ret_line)),
((String)new Character(',').toString()),
((String)new Character('\'').toString()),
       ((String)new Character('"').toString())
};

/**
 *
 */
public RAMStringB64Array(){
    setLineCRLFB64(http1011CacheHeaderParts[2]);
} // CONSTRUCTOR

//  supplied to USER API
public int getTotalLoadedB64Files(){
return (int)flslen;
}

private long garbageCalibrateSizeTotal(){
for(int ie=0;ie<flslen;ie++){
flsarrtotalBytes+=ramFilesLength[ie];
}
flsarrtotalBytes = (flsarrtotalBytes*2);
return flsarrtotalBytes;
}

private long getFreememory(){
Runtime rmtn = Runtime.getRuntime();
return ((long)rmtn.freeMemory());
}

// must be void
private void triggerGarbageRequest(){
if(flsarrtotalBytes > ((long)getFreememory())){
System.gc();
```

```
}
}

// shielded
private boolean setIsInterfaceInstanceReady(){
    if(arrayReady.indexOf("ARRAY_LOADED_READY")!=-1){ // "FAILED_LOAD"
  Ramfile_ifaceReady=true;
    }else{
  Ramfile_ifaceReady=false;
    }
return Ramfile_ifaceReady;
}

// supplied to User API interface
public boolean getIsInterfaceInstanceReadyB64(){
return (boolean)Ramfile_ifaceReady;
}

public String getInterfaceReadyFlagB64(){
return (String)arrayReady; // 0 "UNASSIGNED_ARRAY"     1 "LOADING_ARRAY"      2 "FAILED_LOAD"     3 "ARRAY_LOADED_READY"
}

private String setInterfaceReadyFlag(int rdy,String debug){
arrayReady = Array_Status[rdy]+=debug;
return arrayReady;
}

// shielded - server class use only
private boolean interfaceLoadedReady(boolean boo){
    Ramfile_ifaceReady=boo;
return Ramfile_ifaceReady;
}

private boolean setEncodeUrlSafeB64setting(boolean urlSafe){
urlsafe=urlSafe;
    return urlsafe;
}

public void setEncodeUrlSafeB64(boolean urlSafe){
setEncodeUrlSafeB64setting(urlSafe);
}

public String getUrlSafeLoadingFlagB64(){
String urlSafeFlag="FALSE";
if(((boolean)urlsafe) == true){
urlSafeFlag="TRUE";
}
return (String)urlSafeFlag;
}
// to load by "File object"  to load by text String   --- UTFdDECODER , LOCALE ISO TO WRITE INTO STRINGS SECTIONS FOR PLATFORM - WEB
/**
 * supplied to User API interface
 * @param setlist
 * @throws FileNotFoundException
 * @throws IOException
 */
public void loadPreLoadingListB64(File setlist,boolean urlSafe){ // servlet parsed in list-of-names--single-file object
setEncodeUrlSafeB64(urlSafe);
//
 if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
setInterfaceReadyFlag(0,""); // loading flag
byte[] listed;
long flen = setlist.length();
if(setlist.exists()){
try{
RandomAccessFile tasklist = new RandomAccessFile(setlist,"r");
listed = new byte[new Long(flen).intValue()];
tasklist.read(listed,0,new Long(flen).intValue());
tasklist.close();

String listset = new String(listed);  //byte array to string
loadPreLoadingListB64(listset,urlSafe);
}catch(FileNotFoundException fino){
    fino.printStackTrace();
interfaceLoadedReady(false);
    setInterfaceReadyFlag(2,"_loadinglist-txt-file"); // loading flag
}catch(IOException seeno){
    seeno.printStackTrace();
interfaceLoadedReady(false);

    setInterfaceReadyFlag(2,"_loadinglist-txt-file"); // loading flag
}
}else{
   // new Throwable required  .printStackTrace();
    interfaceLoadedReady(false);
setInterfaceReadyFlag(2,"_loadinglist-txt-file loadPreLoadingList(File) File did not exist"); // loading flag
}
 }//enif ready
}//enmeth


// to load by text String   --- UTFdDECODER , LOCALE ISO TO WRITE INTO STRINGS SECTIONS FOR PLATFORM - WEB
/**
 * supplied to User API interface
 * @param listset
 * @throws java.io.IOException
 */
public void loadPreLoadingListB64(String listset,boolean urlSafe){ // String input version whether web FORM or TXT-FILE output parsed in
    if(urlsafe!=urlSafe){
    setEncodeUrlSafeB64(urlSafe);
    }
 if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
    setInterfaceReadyFlag(0,"_"); // loading flag
    String[] listsetArr; // array for filenames as strings
String cr=http1011CacheHeaderParts[3]; // if the file string is a comma separated list ,
if(listset.indexOf(http1011CacheHeaderParts[4])!=-1){ // SINGLE QUOTE
Pattern.compile(http1011CacheHeaderParts[4]).matcher(listset).replaceAll("");
}
if(listset.indexOf(http1011CacheHeaderParts[5])!=-1){
Pattern.compile(http1011CacheHeaderParts[5]).matcher(listset).replaceAll("");
}
if(listset.indexOf(cr)!=-1){
listsetArr = (Pattern.compile(cr)).split(listset);
}else if(listset.indexOf(http1011CacheHeaderParts[2])!=-1){
    cr=http1011CacheHeaderParts[2];
listsetArr = (Pattern.compile(http1011CacheHeaderParts[2])).split(listset);
}else if(listset.indexOf(http1011CacheHeaderParts[1])!=-1){
    cr=http1011CacheHeaderParts[1];
listsetArr = (Pattern.compile(http1011CacheHeaderParts[1])).split(listset);
}else if(listset.indexOf(http1011CacheHeaderParts[0])!=-1){
    cr=http1011CacheHeaderParts[0];
listsetArr = (Pattern.compile(http1011CacheHeaderParts[0])).split(listset);
}else{
    cr=syslin;
 listsetArr = (Pattern.compile(syslin)).split(listset); // if there is a chance anything is different
}
//  if the array for filenames as strings is not instantiated now then an error has occurred - cease
if(listsetArr!=null){
loadPreLoadingListB64(listsetArr,urlSafe);
}else{
```

```
interfaceLoadedReady(false);
setInterfaceReadyFlag(2,"_loadPreLoadingList(String,boolean)"); // loading flag   0"UNASSIGNED_ARRAY",   1"LOADING_ARRAY",    2"FAILED_LOAD",     3"ARRAY_LOADED_READY"
}
 }//enif ready
}//enmeth


// shielded - server class use only
private String setLineCRLFIn(String CR_LFin){
return CR_LF;
}

// userr interface supplied
public void setLineCRLFB64(String CR_LFin){ // oddly it were temp changed for some reason
setLineCRLFIn(CR_LFin);
}

// shielded - server class use only
private String getLineCRLFOut(){
return (String)CR_LF;
}


// userr interface supplied
public String getLineCRLFB64(){ // oddly it were temp changed for some reason
String crlf = (String) getLineCRLFOut();
return (String)crlf;
}


/**
 * supplied to User API interface
 * @param fls
 * @throws FileNotFoundException
 * @throws IOException
 */
public void loadPreLoadingListB64(String[] fls,boolean urlSafe){
    if(urlsafe!=urlSafe){
    setEncodeUrlSafeB64(urlSafe);
    }
     if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
setInterfaceReadyFlag(1,"_loadPreLoadingList(String[])");
 if(fls.length > 0){
    loadInventoryFileList(fls);
    int fileLen = fls.length;
File[] firay = new File[fileLen];
//
for(int cfl=0;cfl<fileLen;cfl++){
firay[cfl] = new File((fls[cfl].trim()));

}//enfr

loadPreLoadingListB64(firay,urlSafe);
 }else{
     interfaceLoadedReady(false);
   setInterfaceReadyFlag(2,"_loadinglist(String[] fls)");  // 0 "UNASSIGNED_ARRAY"     1 "LOADING_ARRAY"      2 "FAILED_LOAD"     3 "ARRAY_LOADED_READY"
 }
     }//enif ready
}//enmeth


/**
 * supplied to User API interface
 * @param fistset
 * @throws IOException
 */
public void loadPreLoadingListB64(File[] fistset,boolean urlSafe){
    if(urlsafe!=urlSafe){
    setEncodeUrlSafeB64(urlSafe);
    }
     if(arrayReady.indexOf("ARRAY_LOADED_READY")==-1){ // "FAILED_LOAD"
 setInterfaceReadyFlag(0,"_");
    try{
   if(fistset.length > 0){
    loadFilesAndHeaders(fistset);
   }else{
   setInterfaceReadyFlag(2,"_loadinglist-arrasy-file - loadPreLoadingList(File[])");  // 0 "UNASSIGNED_ARRAY"     1 "LOADING_ARRAY"      2 "FAILED_LOAD"     3 "ARRAY_LOADED_READY"
   }
    }catch(IOException ionode){

    }
     }//enif ready
}//enmeth

/**
 * shielded - server class use only
 * @param loaded_RAM_file_inventory
 * @return
 */
private String[] loadInventoryFileList(String[] loaded_inventory){ // load full file names to global for reference
loaded_RAM_file_inventory=(String[])loaded_inventory;
    return loaded_RAM_file_inventory;
}



// **** *** ***   ***** *** *****    *****
// rewiring to global variable and remove passed references TO DO
private File[] fileset(File[] flis){
flsarr=(File[])flis;
    return flsarr;
}


/**
 * shielded - server class use only
 * @param flis
 * @throws java.io.IOException
 */

private void loadFilesAndHeaders(File[] flis) throws java.io.IOException{
fileset(flis);
setInterfaceReadyFlag(0,"_"); // loading flag
resetRamFTaskCount(); // convenient location though already set
fileLen(flis.length);
createFileByteLengthArray(flslen); // create the stored File byte lengths array for each byte array to be instantiated (long)
// call the thread to load files here
runthreadFileLoad(flis);

}//enmeth


/**
 * shielded - server class use only
 * @param lenf
 * @return
 */
private int fileLen(int lenf){
flslen=lenf;
    return flslen;
```

```
}

/**
 * shielded - server class use only
 * @param farr
 * @return
 * @throws FileNotFoundException
 * @throws java.io.IOException
 */
private void instanceFileArrayLoader(File[] farr)throws FileNotFoundException,java.io.IOException{
(filArrod = new FileArrayLoader(farr)).start(); // NOTE PROBABLY SAFEST TO BE 10 PRIORITY
 //return filArrod;
}

// shielded - server class use only
private ArrayLoadingController instanceArrayLoader(File[] farr)throws FileNotFoundException,java.io.IOException{
arrrrLod = new ArrayLoadingController(farr); // NOTE PROBABLY SAFEST TO BE 10 PRIORITY
 return arrrrLod;
}

// shielded - server class use only
private void initRamfileArray(){
    RAMfiles = new String[flslen];
}

/**
 * shielded - server class use only
 * @param farr
 * @throws FileNotFoundException
 * @throws java.io.IOException
 */
// SPECIAL PURPOSE VOID - WITH BASIC INSTANTIATION OF MAIN ARRAY INDEXES
private void runthreadFileLoad(File[] farr)throws FileNotFoundException,java.io.IOException{
setInterfaceReadyFlag(1,"_"); // loading flag
    if(preloadertask == 0){ // if as it should be
instanceArrayLoader(farr);
arrrrLod.start();
}
}//enmeth


/**
 * shielded - server class use only
 * @param felen
 * @return
 */
private long[] createFileByteLengthArray(int felen){ // create array - stores lengths - bytes of file byte array global array indexed
ramFilesLength = new long[felen];
for(int ni=0;ni<felen;ni++){
ramFilesLength[ni]=0;
}
return ramFilesLength; // global
}//enmeth


/**
 * shielded - server class use only
 * @param len
 * @param idx
 * @return
 */
private long setRamlengths(long len,int idx){ // set each corresponding indexes of ramFilesLength array to ofRAMfile array - byte lengths
ramFilesLength[idx] = len;
return ramFilesLength[idx];  //  global
}// enmeth


// supplied to User API interface
public synchronized long getOutWriteRAMbyteB64Length(int arrLenIdX){
long retrivLen=-1;  // returns -1 if out of bounds
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(arrLenIdX > -1)&&(arrLenIdX < ramFilesLength.length)){
    if((arrLenIdX > ((int)-1)) && (arrLenIdX < ramFilesLength.length)){
retrivLen=ramFilesLength[arrLenIdX];
}else{
retrivLen=-1;
}
}//enchk
    return (long)retrivLen;
}

public synchronized long getOutWriteRAMStringB64Length(int arrLenIdX){
long retrivLen=-1;  // returns -1 if out of bounds
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(arrLenIdX > -1)&&(arrLenIdX < ramFilesLength.length)){
    if((arrLenIdX > ((int)-1)) && (arrLenIdX < ramFilesLength.length)){
retrivLen = new Integer(RAMfiles[arrLenIdX].length()).longValue();
}else{
retrivLen=-1;
}
}//enchk
return (long)retrivLen;
}



/**
 * shielded - server class use only
 * @return
 */
private int resetRamFTaskCount(){
preloadertask=0;
return preloadertask;// global
}


/**
 * shielded - server class use only
 * @return
 */
private int preTaskerCount(){
preloadertask++;
return preloadertask;// global
}//enmeth


/**
 * supplied to User API interface
 * @param idxAr
 * @param indByt
 * @return
 */
public synchronized String getRAMstringB64(int idxAr){
    triggerGarbageRequest();
return (String)RAMfiles[idxAr];
}//enmeth



 // shielded - server class use only
private String getLoadedRAMstringB64List(boolean HtmlTXT){
  String linesepSYS =  "<p>"+syslin; // true
```

```
    if(HtmlTXT == false){
linesepSYS =  syslin+syslin;
    }
 if(arrayReady.indexOf("ARRAY_LOADED_READY")!=-1){
    int numf = loaded_RAM_file_inventory.length;
    loaded_files_list = "List of files loaded to B64 ram array:  "+linesepSYS;
for(int cnum=0;cnum<numf;cnum++){
loaded_files_list+=("array-index:  "+cnum+"   filename:  "+loaded_RAM_file_inventory[cnum]+"   bytes: "+getOutWriteRAMStringB64Length(cnum)+linesepSYS);
}
 }//enchk
return (String)loaded_files_list;
}


/**
 * supplied to User API interface
 * @return
 */
public String getLoadedRAMstringB64namesInventory(boolean HtmlTXT){ // TRUE = HTML LINES            FALSE = TEXT-CRLF
String qjump = getLoadedRAMstringB64List(HtmlTXT);
    return (String)qjump;
}

    /**
     * supplied to User API interface
     * @param idxName
     * @return
     */

public synchronized String getLoadedFileNameAtIndexB64(int idxName){
String fipth="FILENAME_ARRAY_UNAVAILABLE   or ERROR array index out of bounds  "+arrayReady;
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(idxName > -1)&&(idxName < flsarr.length)){ //="FILENAME_ARRAY_UNAVAILABLE";
fipth = ((String)flsarr[idxName].getName());
}
    return ((String)fipth);
}

//  supplied to User API interface
public synchronized String getLoadedCanonicalFileNameAtIndexB64(int idxName){
String fipth="FILENAME_ARRAY_UNAVAILABLE   or ERROR array index out of bounds  "+arrayReady;
if((arrayReady.indexOf("ARRAY_LOADED_READY")!=-1)&&(idxName > -1)&&(idxName < flsarr.length)){ //="FILENAME_ARRAY_UNAVAILABLE";
    try{
fipth = ((String)flsarr[idxName].getCanonicalPath());
    }catch(IOException fnout){
            fnout.printStackTrace();
            setInterfaceReadyFlag(2,"_getLoadedCanonicalFileNameAtIndex() IOE  "); //global String arrayReady
        }
}
    return ((String)fipth);
}

//  supplied to User API interface
public String pingMessageTestB64(){
String message = "UNAVAILABLE    "+arrayReady;
if(arrayReady.indexOf("ARRAY_LOADED_READY")!=-1){ //
message = "Version 1 org.apache.catalina.core.RAMStringB64Array class and interface jakarta.servlet.RamString64 23rd September 2020 ( 23/09/2020 added convert to B64 - 16/09/2020 header array concatenators - 12/09/2020 Date adjusters ) (
}
return (String)message;
}


private String fileLoader(File[] fis,File fii){
// load the files to RAM on arrays one by one - global not two by two weave pole
Long lenny = new Long(ramFilesLength[preloadertask]);
try{
RandomAccessFile f = new RandomAccessFile(fii,"r");
int promint = (new Long(lenny).intValue());
byte[] inter =  new byte[(lenny.intValue())]; // instantiate a byte array ready for file bytes
f.readFully(inter,0,promint);
f.close();
RAMfiles[preloadertask]=loadingConverterB64(76,inter,urlsafe); // urlsafe boolean from global setting

// WARNING - SET THE ARRAY DATA ON THE ELEMENT BEFORE INCREMENT OR GARBAGE CLEANUP

}catch(FileNotFoundException fnofl){
    fnofl.printStackTrace();
        setInterfaceReadyFlag(2,"_fileLoader()  FNF  fileLoader(File[] fis,File fii)"); //
        }catch(IOException fnout){
            fnout.printStackTrace();
        setInterfaceReadyFlag(2,"_fileLoader() IOE  fileLoader(File[] fis,File fii)"); //
        }
/*
 get BASE 64 HERE - NOTE - MAXIMUM SIZE FOR URL IMAGES IS NOT LIMITED FOR MAIL MIME EXTENSIONS
org.apache.tomcat.util.codec.binary.Base64
 https://tomcat.apache.org/tomcat-10.0-doc/api/org/apache/tomcat/util/codec/binary/Base64.html
 IE8 maximum data URI size 32768 Bytes

MODIFY THE LOADING METHODS FOR A SAFE / NORMAL B64 ENCODING FLAG

TO DO loadingConverterB64safe(byte[] b64str)
*/
return RAMfiles[preloadertask];
}//enmeth



private synchronized String loadingConverterB64(int chunkLenf, byte[] b64str,boolean usaf){
if(chunkLenf < 2){
chunkLenf=76;
}
Base64 enco = new Base64(chunkLenf,((CR_LF).getBytes()),usaf);
return (String)enco.encodeBase64String(b64str);
}//enmeth

/* org.apache.tomcat.util.codec.binary.Base64
private String loadingConverterB64urlSafe(byte[] b64str){
Base64 enco = new Base64(76,((CR_LF).getBytes()),true);
return (String)enco.encodeBase64URLSafeString(b64str);
}//enmeth
*/

public synchronized String convertToStandardB64(int chunklength,String enctostr){
return (String)loadingConverterB64(chunklength,(enctostr.getBytes()),false);
}//enmeth


public synchronized String convertToUrlSafeB64(int chunklength,String enctostr){
return (String)loadingConverterB64(chunklength,(enctostr.getBytes()),true);
}//enmeth


public synchronized String convertToStandardB64(int chunklength,byte[] encto){
return (String)loadingConverterB64(chunklength,encto,false);
}//enmeth

public synchronized String convertToUrlSafeB64(int chunklength,byte[] encto){
return (String)loadingConverterB64(chunklength,encto,true);
}//enmeth

// obtain object for decoding if neededd
```

```
private synchronized org.apache.tomcat.util.codec.binary.Base64 b64EcoderObject(int lineLength, byte[] lineCRLF,boolean uarelSafe){
org.apache.tomcat.util.codec.binary.Base64 b64obj = new org.apache.tomcat.util.codec.binary.Base64(lineLength,lineCRLF,uarelSafe);
return (org.apache.tomcat.util.codec.binary.Base64)b64obj;
}//enmeth

public synchronized org.apache.tomcat.util.codec.binary.Base64 newb64EcoderObject(int lineLength, byte[] lineCRLF,boolean uarelSafe){
return (org.apache.tomcat.util.codec.binary.Base64)b64EcoderObject(lineLength,lineCRLF,uarelSafe);
}//enmeth

// ALL NESTED CLASSES AND CONTENTS ARE * shielded - server class use only ================================




private class ArrayLoadingController extends Thread{
    File[] fis;

private ArrayLoadingController(File[] fis)throws FileNotFoundException,java.io.IOException{
this.fis=fis;
this.setPriority(10); // thread priority 1 - 10
long leng =0;

int lodflgInv = 0; // controls whether the loading on the STRING LOGGING INVENTORY array takes place after the loop
if(loaded_RAM_file_inventory==null){ // BECAUSE OF THIS THERE MUST BE A RESET METHOD TO RELOAD TO CLEAR THE ARRAY  AND FLAG TO KNOW RELOADING HAS BEEN REQUESTED !!!!
    lodflgInv = 1;
}
// get all file-byte array lengths and set to storage on the long array - global
String[] loaded_inventory = new String[fis.length]; // local - REQUIRES INIT ASSURANCE
for(int ixa=0;ixa < flslen;ixa++){
setRamlengths((fis[ixa].length()),ixa); // load the file-length-size of each file onto an array
if(lodflgInv == 1){
loaded_inventory[ixa] = fis[ixa].getCanonicalPath(); // put the filename onto a string array element (name only)  loaded_RAM_file_inventory
}
}//enfor
if(lodflgInv == 1){
loadInventoryFileList(loaded_inventory); // set global array of filenames        ----------   MAY NEED ARRAYCOPY
}

}//enconstr

public void run(){

   if(preloadertask==0){
initRamfileArray();

        try{
instanceFileArrayLoader(fis);
        }catch(FileNotFoundException fnofl){
            fnofl.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-run"); // LOADED
        }catch(IOException fnout){
            fnout.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-run"); // LOADED
        }
    }

}//enrun


private void reInstanceLoad(File[] fis){
preTaskerCount(); // ++ ready the NOTATION on its array for next file index in superclass for file loading
 try{
if(preloadertask < flslen){
instanceFileArrayLoader(fis); // ready the next++ NEW Thread to load a file and byte array (calling this)
   if(preloadertask==(flslen-1)){ // last file loaded
       setInterfaceReadyFlag(3,""); // LOADED
       if(setIsInterfaceInstanceReady()){
interfaceLoadedReady(true); // allow user test flag to be set to true
loaded_files_list = getLoadedRAMstringB64namesInventory(true);
        }
      }//enif-last-file
// call ready
}
}catch(FileNotFoundException fnofl){
    fnofl.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-reInstanceLoad"); //
        }catch(IOException fnout){
            fnout.printStackTrace();
        setInterfaceReadyFlag(2,"_loadcontroller-reInstanceLoad"); //
        }

}//enmeth

}//enclss


/*
ALL NESTED CLASSES AND CONTENTS ARE * shielded - server class use only ================================
*/

private class FileArrayLoader extends Thread{
//ramary.
   File[] fis;

protected FileArrayLoader(File[] fis){
this.fis=fis;
this.setPriority(10); // thread priority 1 - 10
}//enconstr

// THREADS REQUIRE WHETHER CLASS OR RUNNABLE TO OVERIDE RUN (BECAUSE START AND STOP MUST BE GOVERNED UNLIKE CALLING FROM CONSTRUCTOR OR USING A TIMER)
public void run(){
if(preloadertask < flslen){
fileLoader(fis,fis[preloadertask]); // essential start of file and array index loading sequence
if((preloadertask != (flslen-1)) && (preloadertask < flslen)){
arrrrLod.reInstanceLoad(fis);
}else{
garbageCalibrateSizeTotal();
resetRamFTaskCount();
       System.gc();
}//enifels
}
}//enrun

}//ensubclss

}//enclss




//=============================================================================




//package javax.servlet;
```

```
//package org.apache.catalina.core;
package jakarta.servlet;
import java.io.*;
import jakarta.servlet.*;
//import javax.servlet.Ramfile;
import org.apache.catalina.core.RAMStringB64Array;
/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 25 September 2020
 *
 * uses the API coded javax.servlet.ServletContext THROUGH extension and modification of ApplicationContextFacade.java CLASS
 */
public interface RamStringB64{

    // !!!   loadFilesAndHeaders    DO NOT ACCESS THIS METHOD FROM INTERFACE

// !!!   DO NOT ACCESS ANY METHODS THAT OPERATE DIRECTLY IN OR THROUGH THE SUBCLASSES


public boolean getIsInterfaceInstanceReadyB64();

public String getInterfaceReadyFlagB64(); // required to know if some subtle failure has occurred or the process is not yet completed

// TO CONVERT
public String getLoadedFileNameAtIndexB64(int idxName);

// TO CONVERT
public String getLoadedCanonicalFileNameAtIndexB64(int idxName);

public void setLineCRLFB64(String CR_LF);

public String getLineCRLFB64();

public int getTotalLoadedB64Files();

public void loadPreLoadingListB64(File setlist,boolean urlSafe); // throws FileNotFoundException, IOException;

public void loadPreLoadingListB64(String listset,boolean urlSafe); // throws java.io.IOException

public void loadPreLoadingListB64(String[] fls,boolean urlSafe); // throws FileNotFoundException,IOException

public void loadPreLoadingListB64(File[] fistset,boolean urlSafe); // throws IOException

public String getUrlSafeLoadingFlagB64();

public long getOutWriteRAMbyteB64Length(int arrLenIdX); // supplies the length of the stored file bytes array from index

public long getOutWriteRAMstringB64Length(int arrLenIdX);

public String getRAMstringB64(int idxAr);

public String getLoadedRAMstringB64namesInventory(boolean HtmlTXT);  // user test debug


public String convertToStandardB64(int chunklength,String enctostr);

public String convertToUrlSafeB64(int chunklength,String enctostr);

public String convertToStandardB64(int chunklength,byte[] encto);

public String convertToUrlSafeB64(int chunklength,byte[] encto);

// obtain object for decoding if neededd
public org.apache.tomcat.util.codec.binary.Base64 newb64EcoderObject(int lineLength, byte[] lineCRLF,boolean uarelSafe);


public String pingMessageTestB64();  // user test debug



}//enface









//================================================================================









import jakarta.servlet.Ramfile;
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletContext;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 21st September 2020
 */
public class RAMFileLoaderServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)throws ServletException, IOException {


ServletContext   sxt =  ((ServletConfig)this.getServletConfig()).getServletContext();
Ramfile ramm = getServletContext().getRamFileInstance();
    try {
```

```
String pth = "C:/Program Files (x86)/Apache Software Foundation/Tomcat 10.0/webapps/RAMFileLoader/";
String[] fileslod = new String[9];
for(int cxf=0;cxf<9;cxf++){
fileslod[cxf]=pth+cxf+".jpg";
}
//
int flens = fileslod.length;
java.io.File[] effar = new java.io.File[flens];
for(int ut=0;(ut < flens);ut++){
effar[ut] = new java.io.File(fileslod[ut]);
}//enfr
//
ramm.loadPreLoadingList(effar);
//
}catch(Exception ecces){
    ecces.printStackTrace();
}

        try (PrintWriter out = response.getWriter()) {
             response.setContentType("text/html;charset=UTF-8");
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet RAMFileLoaderServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet RAMFileLoaderServlet at " + request.getContextPath() + "</h1>");
            out.print(""+ramm.getLoadedRAMFileNamesInventory(true));
            out.println("<head>");out.println("</head>");
            out.println(""+ramm.pingMessageTest());
            out.println("<head>");

            out.println("</head>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
     String ret = "(Calls the org.apache.catalina.core.RAMfileArray class THROUGH jakarta.servlet.Ramfile to load files and convert them to Base64)";
    return     ret += "RAMFileLoaderServlet 25th September 2020 - author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia";
    }// </editor-fold>

}//enclss
```

```
//============================================================================
```

```
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletContext;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.RamStringB64;
```

```java
/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 21st September 2020
 */
public class RAMstringB64LoaderServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

   ServletContext   sxt =  ((ServletConfig)this.getServletConfig()).getServletContext();
RamStringB64 r64 = getServletContext().getRamStringB64Instance();
    try {

String pth = "C:/Program Files (x86)/Apache Software Foundation/Tomcat 10.0/webapps/RAMFileLoader/";
String[] fileslod = new String[11];
for(int cxf=0;cxf<9;cxf++){
fileslod[cxf]=pth+cxf+".jpg";
}
fileslod[9]="C:/Program Files (x86)/Apache Software Foundation/Tomcat 10.0/webapps/WebApplication3/671097_61de9d7_900x2999.jpg";
fileslod[10]="C:/Program Files (x86)/Apache Software Foundation/Tomcat 10.0/webapps/WebApplication3/739897.jpg";
int flens = fileslod.length;
java.io.File[] effar = new java.io.File[flens];
for(int ut=0;(ut < flens);ut++){
effar[ut] = new java.io.File(fileslod[ut]);
}//enfr
//0
r64.loadPreLoadingListB64(effar,false); //       trying url safe
//
}catch(Exception ecces){
    ecces.printStackTrace();
}
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet RAMstringB64LoaderServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet RAMstringB64LoaderServlet</h1><h2>" + r64.pingMessageTestB64() + "</h2>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
         String ret = "RAMstringB64LoaderServlet servlet class - Tomcat internal class RAMStringB64Array loader  - 25th September 2020";
     return  ret += "nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia";
    }// </editor-fold>

}//enclss









//============================================================================
```

```
import jakarta.servlet.Ramfile;
import jakarta.servlet.RamStringB64;
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletContext;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 25 September 2020
 */
public class InventoryRAMFile extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
       int imgindex = 4;
        String pram = request.getParameter("idxn");
        if(pram != null){
        imgindex = new Integer((pram.trim())).intValue();
        }
        ;
 //ServletContext   sxt =  ((ServletConfig)this.getServletConfig()).getServletContext();
String adate = "";
Ramfile ramm = getServletContext().getRamFileInstance();
   // ramm.setTestSpaceToken("-");
RamStringB64 ramm64 = getServletContext().getRamStringB64Instance();


java.util. Date dat = ramm.obtainHeaderDate(ramm.systenTimeNow());
dat.setYear(120);
adate = ramm.getCreatedGMTDate(dat);
        try (PrintWriter out = response.getWriter()) {
             response.setContentType("text/html;charset=UTF-8");
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet RAMFileLoaderServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h2>"+adate+"</h2");
            out.println("<h1>Servlet RAMFileLoaderServlet at " + request.getContextPath() + "</h1>");
            out.print(""+ramm.getLoadedRAMFileNamesInventory(true));
            out.println("<h2>");out.println("</h2");
            out.println(""+ramm.pingMessageTest());
            out.println("<h2><p>");
            out.println("<img src=\"http://localhost:8080/RAMFileLoader/RamfileByteOutput?idxn="+imgindex+"&\">");
            out.println("<p></h2>");
            out.println(("<h2> IMAGE INDEX : "+imgindex));out.println("</h2>");

            out.print(("<h2> : "+ramm.getHttpPartsFullArrayLegend(true)));
            out.println("</h2>");
            out.print(("<h2> : "+ramm.getHttpPartsSymbolArrayLegend(true)));
            out.println("</h2>");
            out.print(("<h2> index 24 ( break space ) : "+ramm.getHttpPartsArrayElement(24)+"</h2>"));
            out.println(""+ramm.pingMessageTest());
           ramm.setTestSpaceToken("#");
            String[] joiners = {"1","2","3","15","#and another","15","15","15","# HERE IS AN INSRTED LITERAL ","15","15","#SUDDENLY-ANOTHER-LITERAL","15","15"};
            out.print("<h2> concatHttpArrayParts String[] : "+ramm.concatHttpArrayParts(joiners)+"</h2>");

            out.println("<p>");
            int[] qr = {76,64,25,25};
            out.print("???"+ramm.concatHttpArrayParts(qr));

            out.println("<p>");
            out.print(ramm64.convertToStandardB64(76,ramm.getRAMByteArray(5)));
            out.println("</body>");
            out.println("</html>");
        }catch(Exception ex){
        ex.printStackTrace();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        String ret = "InventoryRAMFile servlet class to audit the org.apache.catalina.core.RAMfileArray class after it is loaded -  25th September 2020";
     return  ret   +="- author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia";
    }// </editor-fold>
```

```
}//encls
```

```
//============================================================================
```

```
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletContext;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.RamStringB64;

/**
 *
 *  @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 21st September 2020
 */
public class RAMstringB64Inventory extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// ServletContext   sxt =  ((ServletConfig)this.getServletConfig()).getServletContext();
RamStringB64 r64 = getServletContext().getRamStringB64Instance();


        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet RAMstringB64Inventory</title>");
            out.println("</head>");
            out.println("<body>");

            out.println("<h2>" + r64.getLoadedRAMstringB64namesInventory(true) + "</h2>");

            out.println("<h2>" + r64.pingMessageTestB64() + "</h2>");



            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
    String   ret = "RAMstringB64Inventory servlet class - Test return Inventory list for RAMStringB64Array Tomcat internal class - 25th September 2020";
```

```
      return  ret  +="  nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia";
    }// </editor-fold>

}//enclss

//============================================================================

import jakarta.servlet.Ramfile;
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletContext;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 *
 * @author nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia September 2020
 */
public class RamfileByteOutput extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int imgindex = 0;
        String pram = request.getParameter("idxn");
        if(pram != null){
        imgindex = new Integer((pram.trim())).intValue();
        }
        // convert to using query-string to get index    --- also application context requires PASSWORD system to protect running ramfile

ServletContext   sxt =  ((ServletConfig)this.getServletConfig()).getServletContext();
Ramfile ramm = sxt.getRamFileInstance();
//response.setContentType("text/html;charset=UTF-8");
 try{
 byte[] bit = ramm.getRAMByteArray(imgindex); // get as soon as possible
long ram1 = ramm.getOutWriteRAMbyteLength(imgindex);
String syslin = System.getProperty("line.separator");
jakarta.servlet.ServletOutputStream speem = response.getOutputStream();
java.io.OutputStream speem2 = (java.io.OutputStream)speem;
long lval = 0;
response.setContentType("image/jpeg");  // application/octet-stream
response.setContentLength((int)ram1);
response.addHeader("Content-Transfer-Encoding"," binary");
response.addHeader("Content-Disposition"," inline; filename=\""+ramm.getLoadedFileNameAtIndex(imgindex)+"\"");
response.addHeader(ramm.getHttpPartsArrayElement(34),"max-age=3600");
int ir = 0;
byte wr;
while(lval< ram1){
   ir =  (int)(new Long(lval).intValue());
    wr = bit[((int)ir)];
    // in = new Long(lval).intValue();
speem2.write(wr);  // may need to convert to read int in RAMArray
lval++;
}
speem2.close();

    }catch(IOException iexo){
iexo.printStackTrace();
}catch(Exception   sexe){
sexe.printStackTrace();



        }//entry
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
```

```
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
       String ret = "RamfileByteOutput servlet class - output byte array file to server output - 25th September 2020";
     return  ret +="  nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia";
    }// </editor-fold>

}//encls
```

```
//============================================================================
```

```
<%--
    Document   : index
    Created on : Sep 19, 2020, 3:51:39 PM
    Author     : nicephotog@gmail.com , nicephotog@yahoo.com.au -  Samuel A Marchant Sydney NSW Australia 25 September 2020
--%>
<%@ page import="jakarta.servlet.ServletContext,jakarta.servlet.Ramfile,jakarta.servlet.ServletConfig,jakarta.servlet.RamStringB64" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
      <%
          int imgno = 7;
          RamStringB64 r64 =((((ServletConfig)this.getServletConfig()).getServletContext()).getRamStringB64Instance());
          out.write("<h2>");

       %>

     <%= ((((ServletConfig)this.getServletConfig()).getServletContext()).getRamFileInstance()).pingMessageTest() %>

     <%--  NOTE  IF ANYTHING YOU USE DOES NOT LEAD A LITERAL IN THE CONCATENATOR WITH THIS $ THEN THERE WILL BE AN ERROR
     <% ((((ServletConfig)this.getServletConfig()).getServletContext()).getRamFileInstance()).setTestSpaceToken("$"); %>
     --%>
<p>
    <%
        out.write("<img src=\""+r64.getRAMstringB64(imgno)+"\" alt=\"image number "+imgno+"\" />");
    %>

</p>



     <% out.write("</h2>"); %>


       <% out.write("<h2>"); %>

       <%= ((((ServletConfig)this.getServletConfig()).getServletContext()).getRamFileInstance()).getTestSpaceToken() %>

          <% out.write("</h2>"); %>

          <p>
    <%
        out.write("<img src=\""+r64.getRAMstringB64(10)+"\" alt=\"image number "+"10"+"\" />");
    %>

</p>

    </body>
</html>
```