**Complete, autogenerated Talend Code**

**Note**: Most of the code here is autogenerated. All routines, regex expressions, filter expressions, join configurations & logic flows that have been developed & designed by Group 1-1 have been documented separately. The tool used to perform data cleansing, transformations & normalization operations is Talend Open Studio for Data Integration.

-------------------------------------------------------Code starts here--------------------------------------------------------------

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

//

// Copyright (c) 2006-2015, Talend Inc.

//

// This source code has been automatically generated by\_Talend Open Studio for Data Integration

// / Licensed under the Apache License, Version 2.0 (the "License");

// you may not use this file except in compliance with the License.

// You may obtain a copy of the License at

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing, software

// distributed under the License is distributed on an "AS IS" BASIS,

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

// See the License for the specific language governing permissions and

// limitations under the License.

**package** adm.sdl\_cleanup\_transformation\_0\_1;

**import** routines.Numeric;

**import** routines.DataOperation;

**import** routines.TalendDataGenerator;

**import** routines.TalendStringUtil;

**import** routines.TalendString;

**import** routines.StringHandling;

**import** routines.Relational;

**import** routines.TalendDate;

**import** routines.Mathematical;

**import** routines.DecodeString;

**import** routines.system.\*;

**import** routines.system.api.\*;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** java.util.List;

**import** java.math.BigDecimal;

**import** java.io.ByteArrayOutputStream;

**import** java.io.ByteArrayInputStream;

**import** java.io.DataInputStream;

**import** java.io.DataOutputStream;

**import** java.io.ObjectOutputStream;

**import** java.io.ObjectInputStream;

**import** java.io.IOException;

**import** java.util.Comparator;

@SuppressWarnings("unused")

/\*\*

\* Job: SDL\_cleanup\_transformation Purpose: <br>

\* Description: <br>

\* **@author** user@talend.com

\* **@version** 7.0.1.20180411\_1414

\* **@status**

\*/

**public** **class** SDL\_cleanup\_transformation **implements** TalendJob {

**protected** **static** **void** logIgnoredError(String message, Throwable cause) {

System.err.println(message);

**if** (cause != **null**) {

cause.printStackTrace();

}

}

**public** **final** Object obj = **new** Object();

// for transmiting parameters purpose

**private** Object valueObject = **null**;

**public** Object getValueObject() {

**return** **this**.valueObject;

}

**public** **void** setValueObject(Object valueObject) {

**this**.valueObject = valueObject;

}

**private** **final** **static** String defaultCharset = java.nio.charset.Charset

.defaultCharset().name();

**private** **final** **static** String utf8Charset = "UTF-8";

// contains type for every context property

**public** **class** PropertiesWithType **extends** java.util.Properties {

**private** **static** **final** **long** serialVersionUID = 1L;

**private** java.util.Map<String, String> propertyTypes = **new** java.util.HashMap<>();

**public** PropertiesWithType(java.util.Properties properties) {

**super**(properties);

}

**public** PropertiesWithType() {

**super**();

}

**public** **void** setContextType(String key, String type) {

propertyTypes.put(key, type);

}

**public** String getContextType(String key) {

**return** propertyTypes.get(key);

}

}

// create and load default properties

**private** java.util.Properties defaultProps = **new** java.util.Properties();

// create application properties with default

**public** **class** ContextProperties **extends** PropertiesWithType {

**private** **static** **final** **long** serialVersionUID = 1L;

**public** ContextProperties(java.util.Properties properties) {

**super**(properties);

}

**public** ContextProperties() {

**super**();

}

**public** **void** synchronizeContext() {

}

}

**private** ContextProperties context = **new** ContextProperties();

**public** ContextProperties getContext() {

**return** **this**.context;

}

**private** **final** String jobVersion = "0.1";

**private** **final** String jobName = "SDL\_cleanup\_transformation";

**private** **final** String projectName = "ADM";

**public** Integer errorCode = **null**;

**private** String currentComponent = "";

**private** **final** java.util.Map<String, Object> globalMap = **new** java.util.HashMap<String, Object>();

**private** **final** **static** java.util.Map<String, Object> junitGlobalMap = **new** java.util.HashMap<String, Object>();

**private** **final** java.util.Map<String, Long> start\_Hash = **new** java.util.HashMap<String, Long>();

**private** **final** java.util.Map<String, Long> end\_Hash = **new** java.util.HashMap<String, Long>();

**private** **final** java.util.Map<String, Boolean> ok\_Hash = **new** java.util.HashMap<String, Boolean>();

**public** **final** java.util.List<String[]> globalBuffer = **new** java.util.ArrayList<String[]>();

**private** RunStat runStat = **new** RunStat();

// OSGi DataSource

**private** **final** **static** String KEY\_DB\_DATASOURCES = "KEY\_DB\_DATASOURCES";

**private** **final** **static** String KEY\_DB\_DATASOURCES\_RAW = "KEY\_DB\_DATASOURCES\_RAW";

**public** **void** setDataSources(

java.util.Map<String, javax.sql.DataSource> dataSources) {

java.util.Map<String, routines.system.TalendDataSource> talendDataSources = **new** java.util.HashMap<String, routines.system.TalendDataSource>();

**for** (java.util.Map.Entry<String, javax.sql.DataSource> dataSourceEntry : dataSources

.entrySet()) {

talendDataSources.put(

dataSourceEntry.getKey(),

**new** routines.system.TalendDataSource(dataSourceEntry

.getValue()));

}

globalMap.put(KEY\_DB\_DATASOURCES, talendDataSources);

globalMap

.put(KEY\_DB\_DATASOURCES\_RAW,

**new** java.util.HashMap<String, javax.sql.DataSource>(

dataSources));

}

StatCatcherUtils tStatCatcher\_1 = **new** StatCatcherUtils(

"\_7EkaUOGDEem0Dci2YvEAnQ", "0.1");

**private** **final** java.io.ByteArrayOutputStream baos = **new** java.io.ByteArrayOutputStream();

**private** **final** java.io.PrintStream errorMessagePS = **new** java.io.PrintStream(

**new** java.io.BufferedOutputStream(baos));

**public** String getExceptionStackTrace() {

**if** ("failure".equals(**this**.getStatus())) {

errorMessagePS.flush();

**return** baos.toString();

}

**return** **null**;

}

**private** Exception exception;

**public** Exception getException() {

**if** ("failure".equals(**this**.getStatus())) {

**return** **this**.exception;

}

**return** **null**;

}

**private** **class** TalendException **extends** Exception {

**private** **static** **final** **long** serialVersionUID = 1L;

**private** java.util.Map<String, Object> globalMap = **null**;

**private** Exception e = **null**;

**private** String currentComponent = **null**;

**private** String virtualComponentName = **null**;

**public** **void** setVirtualComponentName(String virtualComponentName) {

**this**.virtualComponentName = virtualComponentName;

}

**private** TalendException(Exception e, String errorComponent,

**final** java.util.Map<String, Object> globalMap) {

**this**.currentComponent = errorComponent;

**this**.globalMap = globalMap;

**this**.e = e;

}

**public** Exception getException() {

**return** **this**.e;

}

**public** String getCurrentComponent() {

**return** **this**.currentComponent;

}

**public** String getExceptionCauseMessage(Exception e) {

Throwable cause = e;

String message = **null**;

**int** i = 10;

**while** (**null** != cause && 0 < i--) {

message = cause.getMessage();

**if** (**null** == message) {

cause = cause.getCause();

} **else** {

**break**;

}

}

**if** (**null** == message) {

message = e.getClass().getName();

}

**return** message;

}

@Override

**public** **void** printStackTrace() {

**if** (!(e **instanceof** TalendException || e **instanceof** TDieException)) {

**if** (virtualComponentName != **null**

&& currentComponent.indexOf(virtualComponentName + "\_") == 0) {

globalMap.put(virtualComponentName + "\_ERROR\_MESSAGE",

getExceptionCauseMessage(e));

}

globalMap.put(currentComponent + "\_ERROR\_MESSAGE",

getExceptionCauseMessage(e));

System.err.println("Exception in component " + currentComponent

+ " (" + jobName + ")");

}

**if** (!(e **instanceof** TDieException)) {

**if** (e **instanceof** TalendException) {

e.printStackTrace();

} **else** {

e.printStackTrace();

e.printStackTrace(errorMessagePS);

SDL\_cleanup\_transformation.**this**.exception = e;

}

}

**if** (!(e **instanceof** TalendException)) {

**try** {

**for** (java.lang.reflect.Method m : **this**.getClass()

.getEnclosingClass().getMethods()) {

**if** (m.getName().compareTo(currentComponent + "\_error") == 0) {

m.invoke(SDL\_cleanup\_transformation.**this**,

**new** Object[] { e, currentComponent,

globalMap });

**break**;

}

}

**if** (!(e **instanceof** TDieException)) {

}

} **catch** (Exception e) {

**this**.e.printStackTrace();

}

}

}

}

**public** **void** tFileInputDelimited\_1\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_1\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tMap\_1\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_1\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileOutputDelimited\_1\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_1\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileOutputDelimited\_3\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_1\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileInputDelimited\_4\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_4\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tMap\_5\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_4\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileOutputDelimited\_5\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_4\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileOutputDelimited\_6\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_4\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tStatCatcher\_1\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tStatCatcher\_1\_onSubJobError(exception, errorComponent, globalMap);

}

**public** **void** tLogRow\_2\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tStatCatcher\_1\_onSubJobError(exception, errorComponent, globalMap);

}

**public** **void** tFileInputDelimited\_2\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_2\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tUniqRow\_1\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_2\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tMap\_3\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_2\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileOutputDelimited\_2\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_2\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileInputDelimited\_3\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_3\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tMap\_2\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_3\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tUniqRow\_2\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_3\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tMap\_4\_error(Exception exception, String errorComponent,

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_3\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileOutputDelimited\_4\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_3\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileInputDelimited\_5\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_5\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tAdvancedHash\_cleaned\_dict\_lookup\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_2\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tAdvancedHash\_cleaned\_sdl\_inv\_error(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

end\_Hash.put(errorComponent, System.currentTimeMillis());

status = "failure";

tFileInputDelimited\_5\_onSubJobError(exception, errorComponent,

globalMap);

}

**public** **void** tFileInputDelimited\_1\_onSubJobError(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

resumeUtil.addLog("SYSTEM\_LOG", "NODE:" + errorComponent, "", Thread

.currentThread().getId() + "", "FATAL", "",

exception.getMessage(),

ResumeUtil.getExceptionStackTrace(exception), "");

}

**public** **void** tFileInputDelimited\_4\_onSubJobError(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

resumeUtil.addLog("SYSTEM\_LOG", "NODE:" + errorComponent, "", Thread

.currentThread().getId() + "", "FATAL", "",

exception.getMessage(),

ResumeUtil.getExceptionStackTrace(exception), "");

}

**public** **void** tStatCatcher\_1\_onSubJobError(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

resumeUtil.addLog("SYSTEM\_LOG", "NODE:" + errorComponent, "", Thread

.currentThread().getId() + "", "FATAL", "",

exception.getMessage(),

ResumeUtil.getExceptionStackTrace(exception), "");

}

**public** **void** tFileInputDelimited\_2\_onSubJobError(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

resumeUtil.addLog("SYSTEM\_LOG", "NODE:" + errorComponent, "", Thread

.currentThread().getId() + "", "FATAL", "",

exception.getMessage(),

ResumeUtil.getExceptionStackTrace(exception), "");

}

**public** **void** tFileInputDelimited\_3\_onSubJobError(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

resumeUtil.addLog("SYSTEM\_LOG", "NODE:" + errorComponent, "", Thread

.currentThread().getId() + "", "FATAL", "",

exception.getMessage(),

ResumeUtil.getExceptionStackTrace(exception), "");

}

**public** **void** tFileInputDelimited\_5\_onSubJobError(Exception exception,

String errorComponent, **final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

resumeUtil.addLog("SYSTEM\_LOG", "NODE:" + errorComponent, "", Thread

.currentThread().getId() + "", "FATAL", "",

exception.getMessage(),

ResumeUtil.getExceptionStackTrace(exception), "");

}

**public** **static** **class** normalized\_checkoutsStruct **implements**

routines.system.IPersistableRow<normalized\_checkoutsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String ID;

**public** String getID() {

**return** **this**.ID;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** Integer CheckoutMonth;

**public** Integer getCheckoutMonth() {

**return** **this**.CheckoutMonth;

}

**public** Integer CheckoutDate;

**public** Integer getCheckoutDate() {

**return** **this**.CheckoutDate;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String CheckoutTime;

**public** String getCheckoutTime() {

**return** **this**.CheckoutTime;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.ID = readString(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.CheckoutMonth = readInteger(dis);

**this**.CheckoutDate = readInteger(dis);

**this**.CheckoutYear = readInteger(dis);

**this**.CheckoutTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.ID, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// Integer

writeInteger(**this**.CheckoutMonth, dos);

// Integer

writeInteger(**this**.CheckoutDate, dos);

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.CheckoutTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + ID);

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",CheckoutMonth=" + String.valueOf(CheckoutMonth));

sb.append(",CheckoutDate=" + String.valueOf(CheckoutDate));

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",CheckoutTime=" + CheckoutTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(normalized\_checkoutsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** invalid\_checkoutsStruct **implements**

routines.system.IPersistableRow<invalid\_checkoutsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String ID;

**public** String getID() {

**return** **this**.ID;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** String ItemTitle;

**public** String getItemTitle() {

**return** **this**.ItemTitle;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**public** String CheckoutDateTime;

**public** String getCheckoutDateTime() {

**return** **this**.CheckoutDateTime;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.ID = readString(dis);

**this**.CheckoutYear = readInteger(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.ItemTitle = readString(dis);

**this**.Subjects = readString(dis);

**this**.CheckoutDateTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.ID, dos);

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// String

writeString(**this**.ItemTitle, dos);

// String

writeString(**this**.Subjects, dos);

// String

writeString(**this**.CheckoutDateTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + ID);

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",ItemTitle=" + ItemTitle);

sb.append(",Subjects=" + Subjects);

sb.append(",CheckoutDateTime=" + CheckoutDateTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(invalid\_checkoutsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** raw\_checkoutsStruct **implements**

routines.system.IPersistableRow<raw\_checkoutsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** Double ID;

**public** Double getID() {

**return** **this**.ID;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** String ItemTitle;

**public** String getItemTitle() {

**return** **this**.ItemTitle;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**public** String CheckoutDateTime;

**public** String getCheckoutDateTime() {

**return** **this**.CheckoutDateTime;

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

**this**.ID = **null**;

} **else** {

**this**.ID = dis.readDouble();

}

**this**.CheckoutYear = readInteger(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.ItemTitle = readString(dis);

**this**.Subjects = readString(dis);

**this**.CheckoutDateTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// Double

**if** (**this**.ID == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeDouble(**this**.ID);

}

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// String

writeString(**this**.ItemTitle, dos);

// String

writeString(**this**.Subjects, dos);

// String

writeString(**this**.CheckoutDateTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + String.valueOf(ID));

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",ItemTitle=" + ItemTitle);

sb.append(",Subjects=" + Subjects);

sb.append(",CheckoutDateTime=" + CheckoutDateTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(raw\_checkoutsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** after\_tFileInputDelimited\_1Struct **implements**

routines.system.IPersistableRow<after\_tFileInputDelimited\_1Struct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** Double ID;

**public** Double getID() {

**return** **this**.ID;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** String ItemTitle;

**public** String getItemTitle() {

**return** **this**.ItemTitle;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**public** String CheckoutDateTime;

**public** String getCheckoutDateTime() {

**return** **this**.CheckoutDateTime;

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

**this**.ID = **null**;

} **else** {

**this**.ID = dis.readDouble();

}

**this**.CheckoutYear = readInteger(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.ItemTitle = readString(dis);

**this**.Subjects = readString(dis);

**this**.CheckoutDateTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// Double

**if** (**this**.ID == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeDouble(**this**.ID);

}

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// String

writeString(**this**.ItemTitle, dos);

// String

writeString(**this**.Subjects, dos);

// String

writeString(**this**.CheckoutDateTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + String.valueOf(ID));

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",ItemTitle=" + ItemTitle);

sb.append(",Subjects=" + Subjects);

sb.append(",CheckoutDateTime=" + CheckoutDateTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(after\_tFileInputDelimited\_1Struct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **void** tFileInputDelimited\_1Process(

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

globalMap.put("tFileInputDelimited\_1\_SUBPROCESS\_STATE", 0);

**final** **boolean** execStat = **this**.execStat;

String iterateId = "";

String currentComponent = "";

java.util.Map<String, Object> resourceMap = **new** java.util.HashMap<String, Object>();

**try** {

// TDI-39566 avoid throwing an useless Exception

**boolean** resumeIt = **true**;

**if** (globalResumeTicket == **false** && resumeEntryMethodName != **null**) {

String currentMethodName = **new** java.lang.Exception()

.getStackTrace()[0].getMethodName();

resumeIt = resumeEntryMethodName.equals(currentMethodName);

}

**if** (resumeIt || globalResumeTicket) { // start the resume

globalResumeTicket = **true**;

tFileInputDelimited\_3Process(globalMap);

raw\_checkoutsStruct raw\_checkouts = **new** raw\_checkoutsStruct();

normalized\_checkoutsStruct normalized\_checkouts = **new** normalized\_checkoutsStruct();

invalid\_checkoutsStruct invalid\_checkouts = **new** invalid\_checkoutsStruct();

/\*\*

\* [tFileOutputDelimited\_1 begin ] start

\*/

ok\_Hash.put("tFileOutputDelimited\_1", **false**);

start\_Hash.put("tFileOutputDelimited\_1",

System.currentTimeMillis());

currentComponent = "tFileOutputDelimited\_1";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("normalized\_checkouts"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tFileOutputDelimited\_1 = 0;

**class** BytesLimit65535\_tFileOutputDelimited\_1 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileOutputDelimited\_1().limitLog4jByte();

String fileName\_tFileOutputDelimited\_1 = "";

fileName\_tFileOutputDelimited\_1 = (**new** java.io.File(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/out\_accept.csv"))

.getAbsolutePath().replace("\\", "/");

String fullName\_tFileOutputDelimited\_1 = **null**;

String extension\_tFileOutputDelimited\_1 = **null**;

String directory\_tFileOutputDelimited\_1 = **null**;

**if** ((fileName\_tFileOutputDelimited\_1.indexOf("/") != -1)) {

**if** (fileName\_tFileOutputDelimited\_1.lastIndexOf(".") < fileName\_tFileOutputDelimited\_1

.lastIndexOf("/")) {

fullName\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1;

extension\_tFileOutputDelimited\_1 = "";

} **else** {

fullName\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1

.substring(0, fileName\_tFileOutputDelimited\_1

.lastIndexOf("."));

extension\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1

.substring(fileName\_tFileOutputDelimited\_1

.lastIndexOf("."));

}

directory\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1

.substring(0, fileName\_tFileOutputDelimited\_1

.lastIndexOf("/"));

} **else** {

**if** (fileName\_tFileOutputDelimited\_1.lastIndexOf(".") != -1) {

fullName\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1

.substring(0, fileName\_tFileOutputDelimited\_1

.lastIndexOf("."));

extension\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1

.substring(fileName\_tFileOutputDelimited\_1

.lastIndexOf("."));

} **else** {

fullName\_tFileOutputDelimited\_1 = fileName\_tFileOutputDelimited\_1;

extension\_tFileOutputDelimited\_1 = "";

}

directory\_tFileOutputDelimited\_1 = "";

}

**boolean** isFileGenerated\_tFileOutputDelimited\_1 = **true**;

java.io.File filetFileOutputDelimited\_1 = **new** java.io.File(

fileName\_tFileOutputDelimited\_1);

globalMap.put("tFileOutputDelimited\_1\_FILE\_NAME",

fileName\_tFileOutputDelimited\_1);

**int** nb\_line\_tFileOutputDelimited\_1 = 0;

**int** splitedFileNo\_tFileOutputDelimited\_1 = 0;

**int** currentRow\_tFileOutputDelimited\_1 = 0;

**final** String OUT\_DELIM\_tFileOutputDelimited\_1 = /\*\*

\* Start field

\* tFileOutputDelimited\_1:FIELDSEPARATOR

\*/

"^"/\*\* End field tFileOutputDelimited\_1:FIELDSEPARATOR \*/

;

**final** String OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_1 = /\*\*

\* Start

\* field tFileOutputDelimited\_1:ROWSEPARATOR

\*/

"\n"/\*\* End field tFileOutputDelimited\_1:ROWSEPARATOR \*/

;

// create directory only if not exists

**if** (directory\_tFileOutputDelimited\_1 != **null**

&& directory\_tFileOutputDelimited\_1.trim().length() != 0) {

java.io.File dir\_tFileOutputDelimited\_1 = **new** java.io.File(

directory\_tFileOutputDelimited\_1);

**if** (!dir\_tFileOutputDelimited\_1.exists()) {

dir\_tFileOutputDelimited\_1.mkdirs();

}

}

// routines.system.Row

java.io.Writer outtFileOutputDelimited\_1 = **null**;

java.io.File fileToDelete\_tFileOutputDelimited\_1 = **new** java.io.File(

fileName\_tFileOutputDelimited\_1);

**if** (fileToDelete\_tFileOutputDelimited\_1.exists()) {

fileToDelete\_tFileOutputDelimited\_1.delete();

}

outtFileOutputDelimited\_1 = **new** java.io.BufferedWriter(

**new** java.io.OutputStreamWriter(

**new** java.io.FileOutputStream(

fileName\_tFileOutputDelimited\_1, **false**),

"ISO-8859-15"));

**if** (filetFileOutputDelimited\_1.length() == 0) {

outtFileOutputDelimited\_1.write("ID");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("BibNumber");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("ItemBarcode");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("ItemType");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("Collection");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("CallNumber");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("CheckoutMonth");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("CheckoutDate");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("CheckoutYear");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.write("CheckoutTime");

outtFileOutputDelimited\_1

.write(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1.flush();

}

resourceMap.put("out\_tFileOutputDelimited\_1",

outtFileOutputDelimited\_1);

resourceMap.put("nb\_line\_tFileOutputDelimited\_1",

nb\_line\_tFileOutputDelimited\_1);

/\*\*

\* [tFileOutputDelimited\_1 begin ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_3 begin ] start

\*/

ok\_Hash.put("tFileOutputDelimited\_3", **false**);

start\_Hash.put("tFileOutputDelimited\_3",

System.currentTimeMillis());

currentComponent = "tFileOutputDelimited\_3";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("invalid\_checkouts"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tFileOutputDelimited\_3 = 0;

**class** BytesLimit65535\_tFileOutputDelimited\_3 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileOutputDelimited\_3().limitLog4jByte();

String fileName\_tFileOutputDelimited\_3 = "";

fileName\_tFileOutputDelimited\_3 = (**new** java.io.File(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/out\_reject.csv"))

.getAbsolutePath().replace("\\", "/");

String fullName\_tFileOutputDelimited\_3 = **null**;

String extension\_tFileOutputDelimited\_3 = **null**;

String directory\_tFileOutputDelimited\_3 = **null**;

**if** ((fileName\_tFileOutputDelimited\_3.indexOf("/") != -1)) {

**if** (fileName\_tFileOutputDelimited\_3.lastIndexOf(".") < fileName\_tFileOutputDelimited\_3

.lastIndexOf("/")) {

fullName\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3;

extension\_tFileOutputDelimited\_3 = "";

} **else** {

fullName\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3

.substring(0, fileName\_tFileOutputDelimited\_3

.lastIndexOf("."));

extension\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3

.substring(fileName\_tFileOutputDelimited\_3

.lastIndexOf("."));

}

directory\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3

.substring(0, fileName\_tFileOutputDelimited\_3

.lastIndexOf("/"));

} **else** {

**if** (fileName\_tFileOutputDelimited\_3.lastIndexOf(".") != -1) {

fullName\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3

.substring(0, fileName\_tFileOutputDelimited\_3

.lastIndexOf("."));

extension\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3

.substring(fileName\_tFileOutputDelimited\_3

.lastIndexOf("."));

} **else** {

fullName\_tFileOutputDelimited\_3 = fileName\_tFileOutputDelimited\_3;

extension\_tFileOutputDelimited\_3 = "";

}

directory\_tFileOutputDelimited\_3 = "";

}

**boolean** isFileGenerated\_tFileOutputDelimited\_3 = **true**;

java.io.File filetFileOutputDelimited\_3 = **new** java.io.File(

fileName\_tFileOutputDelimited\_3);

globalMap.put("tFileOutputDelimited\_3\_FILE\_NAME",

fileName\_tFileOutputDelimited\_3);

**int** nb\_line\_tFileOutputDelimited\_3 = 0;

**int** splitedFileNo\_tFileOutputDelimited\_3 = 0;

**int** currentRow\_tFileOutputDelimited\_3 = 0;

**final** String OUT\_DELIM\_tFileOutputDelimited\_3 = /\*\*

\* Start field

\* tFileOutputDelimited\_3:FIELDSEPARATOR

\*/

"^"/\*\* End field tFileOutputDelimited\_3:FIELDSEPARATOR \*/

;

**final** String OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_3 = /\*\*

\* Start

\* field tFileOutputDelimited\_3:ROWSEPARATOR

\*/

"\n"/\*\* End field tFileOutputDelimited\_3:ROWSEPARATOR \*/

;

// create directory only if not exists

**if** (directory\_tFileOutputDelimited\_3 != **null**

&& directory\_tFileOutputDelimited\_3.trim().length() != 0) {

java.io.File dir\_tFileOutputDelimited\_3 = **new** java.io.File(

directory\_tFileOutputDelimited\_3);

**if** (!dir\_tFileOutputDelimited\_3.exists()) {

dir\_tFileOutputDelimited\_3.mkdirs();

}

}

// routines.system.Row

java.io.Writer outtFileOutputDelimited\_3 = **null**;

java.io.File fileToDelete\_tFileOutputDelimited\_3 = **new** java.io.File(

fileName\_tFileOutputDelimited\_3);

**if** (fileToDelete\_tFileOutputDelimited\_3.exists()) {

fileToDelete\_tFileOutputDelimited\_3.delete();

}

outtFileOutputDelimited\_3 = **new** java.io.BufferedWriter(

**new** java.io.OutputStreamWriter(

**new** java.io.FileOutputStream(

fileName\_tFileOutputDelimited\_3, **false**),

"ISO-8859-15"));

**if** (filetFileOutputDelimited\_3.length() == 0) {

outtFileOutputDelimited\_3.write("ID");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("CheckoutYear");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("BibNumber");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("ItemBarcode");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("ItemType");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("Collection");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("CallNumber");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("ItemTitle");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("Subjects");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.write("CheckoutDateTime");

outtFileOutputDelimited\_3

.write(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3.flush();

}

resourceMap.put("out\_tFileOutputDelimited\_3",

outtFileOutputDelimited\_3);

resourceMap.put("nb\_line\_tFileOutputDelimited\_3",

nb\_line\_tFileOutputDelimited\_3);

/\*\*

\* [tFileOutputDelimited\_3 begin ] stop

\*/

/\*\*

\* [tMap\_1 begin ] start

\*/

ok\_Hash.put("tMap\_1", **false**);

start\_Hash.put("tMap\_1", System.currentTimeMillis());

currentComponent = "tMap\_1";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("raw\_checkouts"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tMap\_1 = 0;

**class** BytesLimit65535\_tMap\_1 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tMap\_1().limitLog4jByte();

// ###############################

// # Lookup's keys initialization

org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_dict\_lookupStruct> tHash\_Lookup\_cleaned\_dict\_lookup = (org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_dict\_lookupStruct>) ((org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_dict\_lookupStruct>) globalMap

.get("tHash\_Lookup\_cleaned\_dict\_lookup"));

cleaned\_dict\_lookupStruct cleaned\_dict\_lookupHashKey = **new** cleaned\_dict\_lookupStruct();

cleaned\_dict\_lookupStruct cleaned\_dict\_lookupDefault = **new** cleaned\_dict\_lookupStruct();

// ###############################

// ###############################

// # Vars initialization

**class** Var\_\_tMap\_1\_\_Struct {

}

Var\_\_tMap\_1\_\_Struct Var\_\_tMap\_1 = **new** Var\_\_tMap\_1\_\_Struct();

// ###############################

// ###############################

// # Outputs initialization

normalized\_checkoutsStruct normalized\_checkouts\_tmp = **new** normalized\_checkoutsStruct();

invalid\_checkoutsStruct invalid\_checkouts\_tmp = **new** invalid\_checkoutsStruct();

// ###############################

/\*\*

\* [tMap\_1 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_1 begin ] start

\*/

ok\_Hash.put("tFileInputDelimited\_1", **false**);

start\_Hash.put("tFileInputDelimited\_1",

System.currentTimeMillis());

currentComponent = "tFileInputDelimited\_1";

**int** tos\_count\_tFileInputDelimited\_1 = 0;

**class** BytesLimit65535\_tFileInputDelimited\_1 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileInputDelimited\_1().limitLog4jByte();

**final** routines.system.RowState rowstate\_tFileInputDelimited\_1 = **new** routines.system.RowState();

**int** nb\_line\_tFileInputDelimited\_1 = 0;

org.talend.fileprocess.FileInputDelimited fid\_tFileInputDelimited\_1 = **null**;

**try** {

Object filename\_tFileInputDelimited\_1 = "E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/df\_checkouts\_input.csv";

**if** (filename\_tFileInputDelimited\_1 **instanceof** java.io.InputStream) {

**int** footer\_value\_tFileInputDelimited\_1 = 0, random\_value\_tFileInputDelimited\_1 = -1;

**if** (footer\_value\_tFileInputDelimited\_1 > 0

|| random\_value\_tFileInputDelimited\_1 > 0) {

**throw** **new** java.lang.Exception(

"When the input source is a stream,footer and random shouldn't be bigger than 0.");

}

}

**try** {

fid\_tFileInputDelimited\_1 = **new** org.talend.fileprocess.FileInputDelimited(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/df\_checkouts\_input.csv",

"US-ASCII", "^", "\n", **false**, 1, 0, -1, -1,

**false**);

} **catch** (java.lang.Exception e) {

System.err.println(e.getMessage());

}

**while** (fid\_tFileInputDelimited\_1 != **null**

&& fid\_tFileInputDelimited\_1.nextRecord()) {

rowstate\_tFileInputDelimited\_1.reset();

raw\_checkouts = **null**;

**boolean** whetherReject\_tFileInputDelimited\_1 = **false**;

raw\_checkouts = **new** raw\_checkoutsStruct();

**try** {

**int** columnIndexWithD\_tFileInputDelimited\_1 = 0;

String temp = "";

columnIndexWithD\_tFileInputDelimited\_1 = 0;

temp = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

**if** (temp.length() > 0) {

**try** {

raw\_checkouts.ID = ParserUtils

.parseTo\_Double(temp);

} **catch** (java.lang.Exception ex\_tFileInputDelimited\_1) {

rowstate\_tFileInputDelimited\_1

.setException(**new** RuntimeException(

String.format(

"Couldn't parse value for column '%s' in '%s', value is '%s'. Details: %s",

"ID",

"raw\_checkouts",

temp,

ex\_tFileInputDelimited\_1),

ex\_tFileInputDelimited\_1));

}

} **else** {

raw\_checkouts.ID = **null**;

}

columnIndexWithD\_tFileInputDelimited\_1 = 1;

temp = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

**if** (temp.length() > 0) {

**try** {

raw\_checkouts.CheckoutYear = ParserUtils

.parseTo\_Integer(temp);

} **catch** (java.lang.Exception ex\_tFileInputDelimited\_1) {

rowstate\_tFileInputDelimited\_1

.setException(**new** RuntimeException(

String.format(

"Couldn't parse value for column '%s' in '%s', value is '%s'. Details: %s",

"CheckoutYear",

"raw\_checkouts",

temp,

ex\_tFileInputDelimited\_1),

ex\_tFileInputDelimited\_1));

}

} **else** {

raw\_checkouts.CheckoutYear = **null**;

}

columnIndexWithD\_tFileInputDelimited\_1 = 2;

raw\_checkouts.BibNumber = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 3;

raw\_checkouts.ItemBarcode = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 4;

raw\_checkouts.ItemType = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 5;

raw\_checkouts.Collection = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 6;

raw\_checkouts.CallNumber = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 7;

raw\_checkouts.ItemTitle = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 8;

raw\_checkouts.Subjects = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

columnIndexWithD\_tFileInputDelimited\_1 = 9;

raw\_checkouts.CheckoutDateTime = fid\_tFileInputDelimited\_1

.get(columnIndexWithD\_tFileInputDelimited\_1);

**if** (rowstate\_tFileInputDelimited\_1.getException() != **null**) {

**throw** rowstate\_tFileInputDelimited\_1

.getException();

}

} **catch** (java.lang.Exception e) {

whetherReject\_tFileInputDelimited\_1 = **true**;

System.err.println(e.getMessage());

raw\_checkouts = **null**;

}

/\*\*

\* [tFileInputDelimited\_1 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_1 main ] start

\*/

currentComponent = "tFileInputDelimited\_1";

tos\_count\_tFileInputDelimited\_1++;

/\*\*

\* [tFileInputDelimited\_1 main ] stop

\*/

/\*\*

\* [tFileInputDelimited\_1 process\_data\_begin ] start

\*/

currentComponent = "tFileInputDelimited\_1";

/\*\*

\* [tFileInputDelimited\_1 process\_data\_begin ] stop

\*/

// Start of branch "raw\_checkouts"

**if** (raw\_checkouts != **null**) {

/\*\*

\* [tMap\_1 main ] start

\*/

currentComponent = "tMap\_1";

// raw\_checkouts

// raw\_checkouts

**if** (execStat) {

runStat.updateStatOnConnection("raw\_checkouts"

+ iterateId, 1, 1);

}

**boolean** hasCasePrimitiveKeyWithNull\_tMap\_1 = **false**;

// ###############################

// # Input tables (lookups)

**boolean** rejectedInnerJoin\_tMap\_1 = **false**;

**boolean** mainRowRejected\_tMap\_1 = **false**;

// /////////////////////////////////////////////

// Starting Lookup Table "cleaned\_dict\_lookup"

// /////////////////////////////////////////////

**boolean** forceLoopcleaned\_dict\_lookup = **false**;

cleaned\_dict\_lookupStruct cleaned\_dict\_lookupObjectFromLookup = **null**;

**if** (!rejectedInnerJoin\_tMap\_1) { // G\_TM\_M\_020

hasCasePrimitiveKeyWithNull\_tMap\_1 = **false**;

cleaned\_dict\_lookupHashKey.Code = raw\_checkouts.Collection;

cleaned\_dict\_lookupHashKey.hashCodeDirty = **true**;

tHash\_Lookup\_cleaned\_dict\_lookup

.lookup(cleaned\_dict\_lookupHashKey);

**if** (!tHash\_Lookup\_cleaned\_dict\_lookup.hasNext()) { // G\_TM\_M\_090

rejectedInnerJoin\_tMap\_1 = **true**;

forceLoopcleaned\_dict\_lookup = **true**;

} // G\_TM\_M\_090

} // G\_TM\_M\_020

**else** { // G 20 - G 21

forceLoopcleaned\_dict\_lookup = **true**;

} // G 21

cleaned\_dict\_lookupStruct cleaned\_dict\_lookup = **null**;

**while** ((tHash\_Lookup\_cleaned\_dict\_lookup != **null** && tHash\_Lookup\_cleaned\_dict\_lookup

.hasNext()) || forceLoopcleaned\_dict\_lookup) { // G\_TM\_M\_043

// CALL close loop of lookup

// 'cleaned\_dict\_lookup'

cleaned\_dict\_lookupStruct fromLookup\_cleaned\_dict\_lookup = **null**;

cleaned\_dict\_lookup = cleaned\_dict\_lookupDefault;

**if** (!forceLoopcleaned\_dict\_lookup) { // G 46

fromLookup\_cleaned\_dict\_lookup = tHash\_Lookup\_cleaned\_dict\_lookup

.next();

**if** (fromLookup\_cleaned\_dict\_lookup != **null**) {

cleaned\_dict\_lookup = fromLookup\_cleaned\_dict\_lookup;

}

} // G 46

forceLoopcleaned\_dict\_lookup = **false**;

// ###############################

{ // start of Var scope

// ###############################

// # Vars tables

Var\_\_tMap\_1\_\_Struct Var = Var\_\_tMap\_1;// ###############################

// ###############################

// # Output tables

normalized\_checkouts = **null**;

invalid\_checkouts = **null**;

**if** (!rejectedInnerJoin\_tMap\_1) {

// # Output table :

// 'normalized\_checkouts'

normalized\_checkouts\_tmp.ID = Numeric

.sequence("s1", 1, 1) + "";

normalized\_checkouts\_tmp.BibNumber = raw\_checkouts.BibNumber;

normalized\_checkouts\_tmp.ItemBarcode = raw\_checkouts.ItemBarcode;

normalized\_checkouts\_tmp.ItemType = raw\_checkouts.ItemType;

normalized\_checkouts\_tmp.Collection = cleaned\_dict\_lookup.Code\_ID;

normalized\_checkouts\_tmp.CallNumber = raw\_checkouts.CallNumber;

normalized\_checkouts\_tmp.CheckoutMonth = Integer

.parseInt(raw\_checkouts.CheckoutDateTime

.substring(0, 2).trim());

normalized\_checkouts\_tmp.CheckoutDate = Integer

.parseInt(raw\_checkouts.CheckoutDateTime

.substring(3, 5).trim());

normalized\_checkouts\_tmp.CheckoutYear = raw\_checkouts.CheckoutYear;

normalized\_checkouts\_tmp.CheckoutTime = raw\_checkouts.CheckoutDateTime

.substring(10, 22).trim();

normalized\_checkouts = normalized\_checkouts\_tmp;

} // closing inner join bracket (1)

// ###### START REJECTS #####

// # Output reject table :

// 'invalid\_checkouts'

// # Filter conditions

**if** (rejectedInnerJoin\_tMap\_1) {

invalid\_checkouts\_tmp.ID = Numeric

.sequence("s3", 1, 1) + "";

invalid\_checkouts\_tmp.CheckoutYear = raw\_checkouts.CheckoutYear;

invalid\_checkouts\_tmp.BibNumber = raw\_checkouts.BibNumber;

invalid\_checkouts\_tmp.ItemBarcode = raw\_checkouts.ItemBarcode;

invalid\_checkouts\_tmp.ItemType = raw\_checkouts.ItemType;

invalid\_checkouts\_tmp.Collection = raw\_checkouts.Collection;

invalid\_checkouts\_tmp.CallNumber = raw\_checkouts.CallNumber;

invalid\_checkouts\_tmp.ItemTitle = raw\_checkouts.ItemTitle;

invalid\_checkouts\_tmp.Subjects = raw\_checkouts.Subjects;

invalid\_checkouts\_tmp.CheckoutDateTime = raw\_checkouts.CheckoutDateTime;

invalid\_checkouts = invalid\_checkouts\_tmp;

} // closing filter/reject

// ###############################

} // end of Var scope

rejectedInnerJoin\_tMap\_1 = **false**;

tos\_count\_tMap\_1++;

/\*\*

\* [tMap\_1 main ] stop

\*/

/\*\*

\* [tMap\_1 process\_data\_begin ] start

\*/

currentComponent = "tMap\_1";

/\*\*

\* [tMap\_1 process\_data\_begin ] stop

\*/

// Start of branch "normalized\_checkouts"

**if** (normalized\_checkouts != **null**) {

/\*\*

\* [tFileOutputDelimited\_1 main ] start

\*/

currentComponent = "tFileOutputDelimited\_1";

// normalized\_checkouts

// normalized\_checkouts

**if** (execStat) {

runStat.updateStatOnConnection(

"normalized\_checkouts"

+ iterateId, 1, 1);

}

StringBuilder sb\_tFileOutputDelimited\_1 = **new** StringBuilder();

**if** (normalized\_checkouts.ID != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.ID);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.BibNumber != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.BibNumber);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.ItemBarcode != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.ItemBarcode);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.ItemType != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.ItemType);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.Collection != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.Collection);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.CallNumber != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.CallNumber);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.CheckoutMonth != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.CheckoutMonth);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.CheckoutDate != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.CheckoutDate);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.CheckoutYear != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.CheckoutYear);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_tFileOutputDelimited\_1);

**if** (normalized\_checkouts.CheckoutTime != **null**) {

sb\_tFileOutputDelimited\_1

.append(normalized\_checkouts.CheckoutTime);

}

sb\_tFileOutputDelimited\_1

.append(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_1);

nb\_line\_tFileOutputDelimited\_1++;

resourceMap.put(

"nb\_line\_tFileOutputDelimited\_1",

nb\_line\_tFileOutputDelimited\_1);

outtFileOutputDelimited\_1

.write(sb\_tFileOutputDelimited\_1

.toString());

tos\_count\_tFileOutputDelimited\_1++;

/\*\*

\* [tFileOutputDelimited\_1 main ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_1

\* process\_data\_begin ] start

\*/

currentComponent = "tFileOutputDelimited\_1";

/\*\*

\* [tFileOutputDelimited\_1

\* process\_data\_begin ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_1 process\_data\_end

\* ] start

\*/

currentComponent = "tFileOutputDelimited\_1";

/\*\*

\* [tFileOutputDelimited\_1 process\_data\_end

\* ] stop

\*/

} // End of branch "normalized\_checkouts"

// Start of branch "invalid\_checkouts"

**if** (invalid\_checkouts != **null**) {

/\*\*

\* [tFileOutputDelimited\_3 main ] start

\*/

currentComponent = "tFileOutputDelimited\_3";

// invalid\_checkouts

// invalid\_checkouts

**if** (execStat) {

runStat.updateStatOnConnection(

"invalid\_checkouts" + iterateId,

1, 1);

}

StringBuilder sb\_tFileOutputDelimited\_3 = **new** StringBuilder();

**if** (invalid\_checkouts.ID != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.ID);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.CheckoutYear != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.CheckoutYear);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.BibNumber != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.BibNumber);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.ItemBarcode != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.ItemBarcode);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.ItemType != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.ItemType);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.Collection != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.Collection);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.CallNumber != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.CallNumber);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.ItemTitle != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.ItemTitle);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.Subjects != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.Subjects);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_tFileOutputDelimited\_3);

**if** (invalid\_checkouts.CheckoutDateTime != **null**) {

sb\_tFileOutputDelimited\_3

.append(invalid\_checkouts.CheckoutDateTime);

}

sb\_tFileOutputDelimited\_3

.append(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_3);

nb\_line\_tFileOutputDelimited\_3++;

resourceMap.put(

"nb\_line\_tFileOutputDelimited\_3",

nb\_line\_tFileOutputDelimited\_3);

outtFileOutputDelimited\_3

.write(sb\_tFileOutputDelimited\_3

.toString());

tos\_count\_tFileOutputDelimited\_3++;

/\*\*

\* [tFileOutputDelimited\_3 main ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_3

\* process\_data\_begin ] start

\*/

currentComponent = "tFileOutputDelimited\_3";

/\*\*

\* [tFileOutputDelimited\_3

\* process\_data\_begin ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_3 process\_data\_end

\* ] start

\*/

currentComponent = "tFileOutputDelimited\_3";

/\*\*

\* [tFileOutputDelimited\_3 process\_data\_end

\* ] stop

\*/

} // End of branch "invalid\_checkouts"

} // close loop of lookup 'cleaned\_dict\_lookup' //

// G\_TM\_M\_043

/\*\*

\* [tMap\_1 process\_data\_end ] start

\*/

currentComponent = "tMap\_1";

/\*\*

\* [tMap\_1 process\_data\_end ] stop

\*/

} // End of branch "raw\_checkouts"

/\*\*

\* [tFileInputDelimited\_1 process\_data\_end ] start

\*/

currentComponent = "tFileInputDelimited\_1";

/\*\*

\* [tFileInputDelimited\_1 process\_data\_end ] stop

\*/

/\*\*

\* [tFileInputDelimited\_1 end ] start

\*/

currentComponent = "tFileInputDelimited\_1";

}

} **finally** {

**if** (!((Object) ("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/df\_checkouts\_input.csv") **instanceof** java.io.InputStream)) {

**if** (fid\_tFileInputDelimited\_1 != **null**) {

fid\_tFileInputDelimited\_1.close();

}

}

**if** (fid\_tFileInputDelimited\_1 != **null**) {

globalMap.put("tFileInputDelimited\_1\_NB\_LINE",

fid\_tFileInputDelimited\_1.getRowNumber());

}

}

ok\_Hash.put("tFileInputDelimited\_1", **true**);

end\_Hash.put("tFileInputDelimited\_1",

System.currentTimeMillis());

/\*\*

\* [tFileInputDelimited\_1 end ] stop

\*/

/\*\*

\* [tMap\_1 end ] start

\*/

currentComponent = "tMap\_1";

// ###############################

// # Lookup hashes releasing

**if** (tHash\_Lookup\_cleaned\_dict\_lookup != **null**) {

tHash\_Lookup\_cleaned\_dict\_lookup.endGet();

}

globalMap.remove("tHash\_Lookup\_cleaned\_dict\_lookup");

// ###############################

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("raw\_checkouts"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tMap\_1", **true**);

end\_Hash.put("tMap\_1", System.currentTimeMillis());

/\*\*

\* [tMap\_1 end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_1 end ] start

\*/

currentComponent = "tFileOutputDelimited\_1";

**if** (outtFileOutputDelimited\_1 != **null**) {

outtFileOutputDelimited\_1.flush();

outtFileOutputDelimited\_1.close();

}

globalMap.put("tFileOutputDelimited\_1\_NB\_LINE",

nb\_line\_tFileOutputDelimited\_1);

globalMap.put("tFileOutputDelimited\_1\_FILE\_NAME",

fileName\_tFileOutputDelimited\_1);

resourceMap.put("finish\_tFileOutputDelimited\_1", **true**);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("normalized\_checkouts"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tFileOutputDelimited\_1", **true**);

end\_Hash.put("tFileOutputDelimited\_1",

System.currentTimeMillis());

/\*\*

\* [tFileOutputDelimited\_1 end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_3 end ] start

\*/

currentComponent = "tFileOutputDelimited\_3";

**if** (outtFileOutputDelimited\_3 != **null**) {

outtFileOutputDelimited\_3.flush();

outtFileOutputDelimited\_3.close();

}

globalMap.put("tFileOutputDelimited\_3\_NB\_LINE",

nb\_line\_tFileOutputDelimited\_3);

globalMap.put("tFileOutputDelimited\_3\_FILE\_NAME",

fileName\_tFileOutputDelimited\_3);

resourceMap.put("finish\_tFileOutputDelimited\_3", **true**);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("invalid\_checkouts"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tFileOutputDelimited\_3", **true**);

end\_Hash.put("tFileOutputDelimited\_3",

System.currentTimeMillis());

/\*\*

\* [tFileOutputDelimited\_3 end ] stop

\*/

}// end the resume

**if** (resumeEntryMethodName == **null** || globalResumeTicket) {

resumeUtil

.addLog("CHECKPOINT",

"CONNECTION:SUBJOB\_OK:tFileInputDelimited\_1:OnSubjobOk",

"", Thread.currentThread().getId() + "", "",

"", "", "", "");

}

**if** (execStat) {

runStat.updateStatOnConnection("OnSubjobOk2", 0, "ok");

}

tFileInputDelimited\_4Process(globalMap);

} **catch** (java.lang.Exception e) {

TalendException te = **new** TalendException(e, currentComponent,

globalMap);

**throw** te;

} **catch** (java.lang.Error error) {

runStat.stopThreadStat();

**throw** error;

} **finally** {

// free memory for "tMap\_1"

globalMap.remove("tHash\_Lookup\_cleaned\_dict\_lookup");

**try** {

/\*\*

\* [tFileInputDelimited\_1 finally ] start

\*/

currentComponent = "tFileInputDelimited\_1";

/\*\*

\* [tFileInputDelimited\_1 finally ] stop

\*/

/\*\*

\* [tMap\_1 finally ] start

\*/

currentComponent = "tMap\_1";

/\*\*

\* [tMap\_1 finally ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_1 finally ] start

\*/

currentComponent = "tFileOutputDelimited\_1";

**if** (resourceMap.get("finish\_tFileOutputDelimited\_1") == **null**) {

java.io.Writer outtFileOutputDelimited\_1 = (java.io.Writer) resourceMap

.get("out\_tFileOutputDelimited\_1");

**if** (outtFileOutputDelimited\_1 != **null**) {

outtFileOutputDelimited\_1.flush();

outtFileOutputDelimited\_1.close();

}

}

/\*\*

\* [tFileOutputDelimited\_1 finally ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_3 finally ] start

\*/

currentComponent = "tFileOutputDelimited\_3";

**if** (resourceMap.get("finish\_tFileOutputDelimited\_3") == **null**) {

java.io.Writer outtFileOutputDelimited\_3 = (java.io.Writer) resourceMap

.get("out\_tFileOutputDelimited\_3");

**if** (outtFileOutputDelimited\_3 != **null**) {

outtFileOutputDelimited\_3.flush();

outtFileOutputDelimited\_3.close();

}

}

/\*\*

\* [tFileOutputDelimited\_3 finally ] stop

\*/

} **catch** (java.lang.Exception e) {

// ignore

} **catch** (java.lang.Error error) {

// ignore

}

resourceMap = **null**;

}

globalMap.put("tFileInputDelimited\_1\_SUBPROCESS\_STATE", 1);

}

**public** **static** **class** validcheckoutsStruct **implements**

routines.system.IPersistableRow<validcheckoutsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String ID;

**public** String getID() {

**return** **this**.ID;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** Integer CheckoutMonth;

**public** Integer getCheckoutMonth() {

**return** **this**.CheckoutMonth;

}

**public** Integer CheckoutDate;

**public** Integer getCheckoutDate() {

**return** **this**.CheckoutDate;

}

**public** String CheckoutYear;

**public** String getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String CheckoutTime;

**public** String getCheckoutTime() {

**return** **this**.CheckoutTime;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.ID = readString(dis);

**this**.BibNumber = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.CheckoutMonth = readInteger(dis);

**this**.CheckoutDate = readInteger(dis);

**this**.CheckoutYear = readString(dis);

**this**.CheckoutTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.ID, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// Integer

writeInteger(**this**.CheckoutMonth, dos);

// Integer

writeInteger(**this**.CheckoutDate, dos);

// String

writeString(**this**.CheckoutYear, dos);

// String

writeString(**this**.CheckoutTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + ID);

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",CheckoutMonth=" + String.valueOf(CheckoutMonth));

sb.append(",CheckoutDate=" + String.valueOf(CheckoutDate));

sb.append(",CheckoutYear=" + CheckoutYear);

sb.append(",CheckoutTime=" + CheckoutTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(validcheckoutsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** invalidcheckoutsStruct **implements**

routines.system.IPersistableRow<invalidcheckoutsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String ID;

**public** String getID() {

**return** **this**.ID;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** String Title;

**public** String getTitle() {

**return** **this**.Title;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**public** String Author;

**public** String getAuthor() {

**return** **this**.Author;

}

**public** Integer CheckoutMonth;

**public** Integer getCheckoutMonth() {

**return** **this**.CheckoutMonth;

}

**public** Integer CheckoutDate;

**public** Integer getCheckoutDate() {

**return** **this**.CheckoutDate;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String CheckoutTime;

**public** String getCheckoutTime() {

**return** **this**.CheckoutTime;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.ID = readString(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.Title = readString(dis);

**this**.Subjects = readString(dis);

**this**.Author = readString(dis);

**this**.CheckoutMonth = readInteger(dis);

**this**.CheckoutDate = readInteger(dis);

**this**.CheckoutYear = readInteger(dis);

**this**.CheckoutTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.ID, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// String

writeString(**this**.Title, dos);

// String

writeString(**this**.Subjects, dos);

// String

writeString(**this**.Author, dos);

// Integer

writeInteger(**this**.CheckoutMonth, dos);

// Integer

writeInteger(**this**.CheckoutDate, dos);

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.CheckoutTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + ID);

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",Title=" + Title);

sb.append(",Subjects=" + Subjects);

sb.append(",Author=" + Author);

sb.append(",CheckoutMonth=" + String.valueOf(CheckoutMonth));

sb.append(",CheckoutDate=" + String.valueOf(CheckoutDate));

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",CheckoutTime=" + CheckoutTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(invalidcheckoutsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** norm\_checkoutsStruct **implements**

routines.system.IPersistableRow<norm\_checkoutsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String ID;

**public** String getID() {

**return** **this**.ID;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** Integer CheckoutMonth;

**public** Integer getCheckoutMonth() {

**return** **this**.CheckoutMonth;

}

**public** Integer CheckoutDate;

**public** Integer getCheckoutDate() {

**return** **this**.CheckoutDate;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String CheckoutTime;

**public** String getCheckoutTime() {

**return** **this**.CheckoutTime;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.ID = readString(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.CheckoutMonth = readInteger(dis);

**this**.CheckoutDate = readInteger(dis);

**this**.CheckoutYear = readInteger(dis);

**this**.CheckoutTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.ID, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// Integer

writeInteger(**this**.CheckoutMonth, dos);

// Integer

writeInteger(**this**.CheckoutDate, dos);

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.CheckoutTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + ID);

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",CheckoutMonth=" + String.valueOf(CheckoutMonth));

sb.append(",CheckoutDate=" + String.valueOf(CheckoutDate));

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",CheckoutTime=" + CheckoutTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(norm\_checkoutsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** after\_tFileInputDelimited\_4Struct **implements**

routines.system.IPersistableRow<after\_tFileInputDelimited\_4Struct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String ID;

**public** String getID() {

**return** **this**.ID;

}

**public** String BibNumber;

**public** String getBibNumber() {

**return** **this**.BibNumber;

}

**public** String ItemBarcode;

**public** String getItemBarcode() {

**return** **this**.ItemBarcode;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String Collection;

**public** String getCollection() {

**return** **this**.Collection;

}

**public** String CallNumber;

**public** String getCallNumber() {

**return** **this**.CallNumber;

}

**public** Integer CheckoutMonth;

**public** Integer getCheckoutMonth() {

**return** **this**.CheckoutMonth;

}

**public** Integer CheckoutDate;

**public** Integer getCheckoutDate() {

**return** **this**.CheckoutDate;

}

**public** Integer CheckoutYear;

**public** Integer getCheckoutYear() {

**return** **this**.CheckoutYear;

}

**public** String CheckoutTime;

**public** String getCheckoutTime() {

**return** **this**.CheckoutTime;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.ID = readString(dis);

**this**.BibNumber = readString(dis);

**this**.ItemBarcode = readString(dis);

**this**.ItemType = readString(dis);

**this**.Collection = readString(dis);

**this**.CallNumber = readString(dis);

**this**.CheckoutMonth = readInteger(dis);

**this**.CheckoutDate = readInteger(dis);

**this**.CheckoutYear = readInteger(dis);

**this**.CheckoutTime = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.ID, dos);

// String

writeString(**this**.BibNumber, dos);

// String

writeString(**this**.ItemBarcode, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.Collection, dos);

// String

writeString(**this**.CallNumber, dos);

// Integer

writeInteger(**this**.CheckoutMonth, dos);

// Integer

writeInteger(**this**.CheckoutDate, dos);

// Integer

writeInteger(**this**.CheckoutYear, dos);

// String

writeString(**this**.CheckoutTime, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("ID=" + ID);

sb.append(",BibNumber=" + BibNumber);

sb.append(",ItemBarcode=" + ItemBarcode);

sb.append(",ItemType=" + ItemType);

sb.append(",Collection=" + Collection);

sb.append(",CallNumber=" + CallNumber);

sb.append(",CheckoutMonth=" + String.valueOf(CheckoutMonth));

sb.append(",CheckoutDate=" + String.valueOf(CheckoutDate));

sb.append(",CheckoutYear=" + String.valueOf(CheckoutYear));

sb.append(",CheckoutTime=" + CheckoutTime);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(after\_tFileInputDelimited\_4Struct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **void** tFileInputDelimited\_4Process(

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

globalMap.put("tFileInputDelimited\_4\_SUBPROCESS\_STATE", 0);

**final** **boolean** execStat = **this**.execStat;

String iterateId = "";

String currentComponent = "";

java.util.Map<String, Object> resourceMap = **new** java.util.HashMap<String, Object>();

**try** {

// TDI-39566 avoid throwing an useless Exception

**boolean** resumeIt = **true**;

**if** (globalResumeTicket == **false** && resumeEntryMethodName != **null**) {

String currentMethodName = **new** java.lang.Exception()

.getStackTrace()[0].getMethodName();

resumeIt = resumeEntryMethodName.equals(currentMethodName);

}

**if** (resumeIt || globalResumeTicket) { // start the resume

globalResumeTicket = **true**;

tFileInputDelimited\_5Process(globalMap);

norm\_checkoutsStruct norm\_checkouts = **new** norm\_checkoutsStruct();

validcheckoutsStruct validcheckouts = **new** validcheckoutsStruct();

invalidcheckoutsStruct invalidcheckouts = **new** invalidcheckoutsStruct();

/\*\*

\* [tFileOutputDelimited\_5 begin ] start

\*/

ok\_Hash.put("tFileOutputDelimited\_5", **false**);

start\_Hash.put("tFileOutputDelimited\_5",

System.currentTimeMillis());

currentComponent = "tFileOutputDelimited\_5";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("validcheckouts"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tFileOutputDelimited\_5 = 0;

**class** BytesLimit65535\_tFileOutputDelimited\_5 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileOutputDelimited\_5().limitLog4jByte();

String fileName\_tFileOutputDelimited\_5 = "";

fileName\_tFileOutputDelimited\_5 = (**new** java.io.File(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/clean\_final/out\_accept\_cleaner.csv"))

.getAbsolutePath().replace("\\", "/");

String fullName\_tFileOutputDelimited\_5 = **null**;

String extension\_tFileOutputDelimited\_5 = **null**;

String directory\_tFileOutputDelimited\_5 = **null**;

**if** ((fileName\_tFileOutputDelimited\_5.indexOf("/") != -1)) {

**if** (fileName\_tFileOutputDelimited\_5.lastIndexOf(".") < fileName\_tFileOutputDelimited\_5

.lastIndexOf("/")) {

fullName\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5;

extension\_tFileOutputDelimited\_5 = "";

} **else** {

fullName\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5

.substring(0, fileName\_tFileOutputDelimited\_5

.lastIndexOf("."));

extension\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5

.substring(fileName\_tFileOutputDelimited\_5

.lastIndexOf("."));

}

directory\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5

.substring(0, fileName\_tFileOutputDelimited\_5

.lastIndexOf("/"));

} **else** {

**if** (fileName\_tFileOutputDelimited\_5.lastIndexOf(".") != -1) {

fullName\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5

.substring(0, fileName\_tFileOutputDelimited\_5

.lastIndexOf("."));

extension\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5

.substring(fileName\_tFileOutputDelimited\_5

.lastIndexOf("."));

} **else** {

fullName\_tFileOutputDelimited\_5 = fileName\_tFileOutputDelimited\_5;

extension\_tFileOutputDelimited\_5 = "";

}

directory\_tFileOutputDelimited\_5 = "";

}

**boolean** isFileGenerated\_tFileOutputDelimited\_5 = **true**;

java.io.File filetFileOutputDelimited\_5 = **new** java.io.File(

fileName\_tFileOutputDelimited\_5);

globalMap.put("tFileOutputDelimited\_5\_FILE\_NAME",

fileName\_tFileOutputDelimited\_5);

**int** nb\_line\_tFileOutputDelimited\_5 = 0;

**int** splitedFileNo\_tFileOutputDelimited\_5 = 0;

**int** currentRow\_tFileOutputDelimited\_5 = 0;

**final** String OUT\_DELIM\_tFileOutputDelimited\_5 = /\*\*

\* Start field

\* tFileOutputDelimited\_5:FIELDSEPARATOR

\*/

"^"/\*\* End field tFileOutputDelimited\_5:FIELDSEPARATOR \*/

;

**final** String OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_5 = /\*\*

\* Start

\* field tFileOutputDelimited\_5:ROWSEPARATOR

\*/

"\n"/\*\* End field tFileOutputDelimited\_5:ROWSEPARATOR \*/

;

// create directory only if not exists

**if** (directory\_tFileOutputDelimited\_5 != **null**

&& directory\_tFileOutputDelimited\_5.trim().length() != 0) {

java.io.File dir\_tFileOutputDelimited\_5 = **new** java.io.File(

directory\_tFileOutputDelimited\_5);

**if** (!dir\_tFileOutputDelimited\_5.exists()) {

dir\_tFileOutputDelimited\_5.mkdirs();

}

}

// routines.system.Row

java.io.Writer outtFileOutputDelimited\_5 = **null**;

java.io.File fileToDelete\_tFileOutputDelimited\_5 = **new** java.io.File(

fileName\_tFileOutputDelimited\_5);

**if** (fileToDelete\_tFileOutputDelimited\_5.exists()) {

fileToDelete\_tFileOutputDelimited\_5.delete();

}

outtFileOutputDelimited\_5 = **new** java.io.BufferedWriter(

**new** java.io.OutputStreamWriter(

**new** java.io.FileOutputStream(

fileName\_tFileOutputDelimited\_5, **false**),

"ISO-8859-15"));

**if** (filetFileOutputDelimited\_5.length() == 0) {

outtFileOutputDelimited\_5.write("ID");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("BibNumber");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("ItemType");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("Collection");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("CallNumber");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("CheckoutMonth");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("CheckoutDate");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("CheckoutYear");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.write("CheckoutTime");

outtFileOutputDelimited\_5

.write(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5.flush();

}

resourceMap.put("out\_tFileOutputDelimited\_5",

outtFileOutputDelimited\_5);

resourceMap.put("nb\_line\_tFileOutputDelimited\_5",

nb\_line\_tFileOutputDelimited\_5);

/\*\*

\* [tFileOutputDelimited\_5 begin ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_6 begin ] start

\*/

ok\_Hash.put("tFileOutputDelimited\_6", **false**);

start\_Hash.put("tFileOutputDelimited\_6",

System.currentTimeMillis());

currentComponent = "tFileOutputDelimited\_6";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("invalidcheckouts"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tFileOutputDelimited\_6 = 0;

**class** BytesLimit65535\_tFileOutputDelimited\_6 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileOutputDelimited\_6().limitLog4jByte();

String fileName\_tFileOutputDelimited\_6 = "";

fileName\_tFileOutputDelimited\_6 = (**new** java.io.File(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/clean\_final/out\_inexplicable\_checkouts.csv"))

.getAbsolutePath().replace("\\", "/");

String fullName\_tFileOutputDelimited\_6 = **null**;

String extension\_tFileOutputDelimited\_6 = **null**;

String directory\_tFileOutputDelimited\_6 = **null**;

**if** ((fileName\_tFileOutputDelimited\_6.indexOf("/") != -1)) {

**if** (fileName\_tFileOutputDelimited\_6.lastIndexOf(".") < fileName\_tFileOutputDelimited\_6

.lastIndexOf("/")) {

fullName\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6;

extension\_tFileOutputDelimited\_6 = "";

} **else** {

fullName\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6

.substring(0, fileName\_tFileOutputDelimited\_6

.lastIndexOf("."));

extension\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6

.substring(fileName\_tFileOutputDelimited\_6

.lastIndexOf("."));

}

directory\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6

.substring(0, fileName\_tFileOutputDelimited\_6

.lastIndexOf("/"));

} **else** {

**if** (fileName\_tFileOutputDelimited\_6.lastIndexOf(".") != -1) {

fullName\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6

.substring(0, fileName\_tFileOutputDelimited\_6

.lastIndexOf("."));

extension\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6

.substring(fileName\_tFileOutputDelimited\_6

.lastIndexOf("."));

} **else** {

fullName\_tFileOutputDelimited\_6 = fileName\_tFileOutputDelimited\_6;

extension\_tFileOutputDelimited\_6 = "";

}

directory\_tFileOutputDelimited\_6 = "";

}

**boolean** isFileGenerated\_tFileOutputDelimited\_6 = **true**;

java.io.File filetFileOutputDelimited\_6 = **new** java.io.File(

fileName\_tFileOutputDelimited\_6);

globalMap.put("tFileOutputDelimited\_6\_FILE\_NAME",

fileName\_tFileOutputDelimited\_6);

**int** nb\_line\_tFileOutputDelimited\_6 = 0;

**int** splitedFileNo\_tFileOutputDelimited\_6 = 0;

**int** currentRow\_tFileOutputDelimited\_6 = 0;

**final** String OUT\_DELIM\_tFileOutputDelimited\_6 = /\*\*

\* Start field

\* tFileOutputDelimited\_6:FIELDSEPARATOR

\*/

"^"/\*\* End field tFileOutputDelimited\_6:FIELDSEPARATOR \*/

;

**final** String OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_6 = /\*\*

\* Start

\* field tFileOutputDelimited\_6:ROWSEPARATOR

\*/

"\n"/\*\* End field tFileOutputDelimited\_6:ROWSEPARATOR \*/

;

// create directory only if not exists

**if** (directory\_tFileOutputDelimited\_6 != **null**

&& directory\_tFileOutputDelimited\_6.trim().length() != 0) {

java.io.File dir\_tFileOutputDelimited\_6 = **new** java.io.File(

directory\_tFileOutputDelimited\_6);

**if** (!dir\_tFileOutputDelimited\_6.exists()) {

dir\_tFileOutputDelimited\_6.mkdirs();

}

}

// routines.system.Row

java.io.Writer outtFileOutputDelimited\_6 = **null**;

java.io.File fileToDelete\_tFileOutputDelimited\_6 = **new** java.io.File(

fileName\_tFileOutputDelimited\_6);

**if** (fileToDelete\_tFileOutputDelimited\_6.exists()) {

fileToDelete\_tFileOutputDelimited\_6.delete();

}

outtFileOutputDelimited\_6 = **new** java.io.BufferedWriter(

**new** java.io.OutputStreamWriter(

**new** java.io.FileOutputStream(

fileName\_tFileOutputDelimited\_6, **false**),

"ISO-8859-15"));

**if** (filetFileOutputDelimited\_6.length() == 0) {

outtFileOutputDelimited\_6.write("ID");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("BibNumber");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("ItemBarcode");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("ItemType");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("Collection");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("CallNumber");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("Title");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("Subjects");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("Author");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("CheckoutMonth");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("CheckoutDate");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("CheckoutYear");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.write("CheckoutTime");

outtFileOutputDelimited\_6

.write(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6.flush();

}

resourceMap.put("out\_tFileOutputDelimited\_6",

outtFileOutputDelimited\_6);

resourceMap.put("nb\_line\_tFileOutputDelimited\_6",

nb\_line\_tFileOutputDelimited\_6);

/\*\*

\* [tFileOutputDelimited\_6 begin ] stop

\*/

/\*\*

\* [tMap\_5 begin ] start

\*/

ok\_Hash.put("tMap\_5", **false**);

start\_Hash.put("tMap\_5", System.currentTimeMillis());

currentComponent = "tMap\_5";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("norm\_checkouts"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tMap\_5 = 0;

**class** BytesLimit65535\_tMap\_5 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tMap\_5().limitLog4jByte();

// ###############################

// # Lookup's keys initialization

org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_sdl\_invStruct> tHash\_Lookup\_cleaned\_sdl\_inv = (org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_sdl\_invStruct>) ((org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_sdl\_invStruct>) globalMap

.get("tHash\_Lookup\_cleaned\_sdl\_inv"));

cleaned\_sdl\_invStruct cleaned\_sdl\_invHashKey = **new** cleaned\_sdl\_invStruct();

cleaned\_sdl\_invStruct cleaned\_sdl\_invDefault = **new** cleaned\_sdl\_invStruct();

// ###############################

// ###############################

// # Vars initialization

**class** Var\_\_tMap\_5\_\_Struct {

}

Var\_\_tMap\_5\_\_Struct Var\_\_tMap\_5 = **new** Var\_\_tMap\_5\_\_Struct();

// ###############################

// ###############################

// # Outputs initialization

validcheckoutsStruct validcheckouts\_tmp = **new** validcheckoutsStruct();

invalidcheckoutsStruct invalidcheckouts\_tmp = **new** invalidcheckoutsStruct();

// ###############################

/\*\*

\* [tMap\_5 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_4 begin ] start

\*/

ok\_Hash.put("tFileInputDelimited\_4", **false**);

start\_Hash.put("tFileInputDelimited\_4",

System.currentTimeMillis());

currentComponent = "tFileInputDelimited\_4";

**int** tos\_count\_tFileInputDelimited\_4 = 0;

**class** BytesLimit65535\_tFileInputDelimited\_4 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileInputDelimited\_4().limitLog4jByte();

**final** routines.system.RowState rowstate\_tFileInputDelimited\_4 = **new** routines.system.RowState();

**int** nb\_line\_tFileInputDelimited\_4 = 0;

org.talend.fileprocess.FileInputDelimited fid\_tFileInputDelimited\_4 = **null**;

**try** {

Object filename\_tFileInputDelimited\_4 = "E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/out\_accept.csv";

**if** (filename\_tFileInputDelimited\_4 **instanceof** java.io.InputStream) {

**int** footer\_value\_tFileInputDelimited\_4 = 0, random\_value\_tFileInputDelimited\_4 = -1;

**if** (footer\_value\_tFileInputDelimited\_4 > 0

|| random\_value\_tFileInputDelimited\_4 > 0) {

**throw** **new** java.lang.Exception(

"When the input source is a stream,footer and random shouldn't be bigger than 0.");

}

}

**try** {

fid\_tFileInputDelimited\_4 = **new** org.talend.fileprocess.FileInputDelimited(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/out\_accept.csv",

"US-ASCII", "^", "\n", **false**, 1, 0, -1, -1,

**false**);

} **catch** (java.lang.Exception e) {

System.err.println(e.getMessage());

}

**while** (fid\_tFileInputDelimited\_4 != **null**

&& fid\_tFileInputDelimited\_4.nextRecord()) {

rowstate\_tFileInputDelimited\_4.reset();

norm\_checkouts = **null**;

**boolean** whetherReject\_tFileInputDelimited\_4 = **false**;

norm\_checkouts = **new** norm\_checkoutsStruct();

**try** {

**int** columnIndexWithD\_tFileInputDelimited\_4 = 0;

String temp = "";

columnIndexWithD\_tFileInputDelimited\_4 = 0;

norm\_checkouts.ID = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

columnIndexWithD\_tFileInputDelimited\_4 = 1;

norm\_checkouts.BibNumber = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

columnIndexWithD\_tFileInputDelimited\_4 = 2;

norm\_checkouts.ItemBarcode = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

columnIndexWithD\_tFileInputDelimited\_4 = 3;

norm\_checkouts.ItemType = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

columnIndexWithD\_tFileInputDelimited\_4 = 4;

norm\_checkouts.Collection = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

columnIndexWithD\_tFileInputDelimited\_4 = 5;

norm\_checkouts.CallNumber = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

columnIndexWithD\_tFileInputDelimited\_4 = 6;

temp = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

**if** (temp.length() > 0) {

**try** {

norm\_checkouts.CheckoutMonth = ParserUtils

.parseTo\_Integer(temp);

} **catch** (java.lang.Exception ex\_tFileInputDelimited\_4) {

rowstate\_tFileInputDelimited\_4

.setException(**new** RuntimeException(

String.format(

"Couldn't parse value for column '%s' in '%s', value is '%s'. Details: %s",

"CheckoutMonth",

"norm\_checkouts",

temp,

ex\_tFileInputDelimited\_4),

ex\_tFileInputDelimited\_4));

}

} **else** {

norm\_checkouts.CheckoutMonth = **null**;

}

columnIndexWithD\_tFileInputDelimited\_4 = 7;

temp = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

**if** (temp.length() > 0) {

**try** {

norm\_checkouts.CheckoutDate = ParserUtils

.parseTo\_Integer(temp);

} **catch** (java.lang.Exception ex\_tFileInputDelimited\_4) {

rowstate\_tFileInputDelimited\_4

.setException(**new** RuntimeException(

String.format(

"Couldn't parse value for column '%s' in '%s', value is '%s'. Details: %s",

"CheckoutDate",

"norm\_checkouts",

temp,

ex\_tFileInputDelimited\_4),

ex\_tFileInputDelimited\_4));

}

} **else** {

norm\_checkouts.CheckoutDate = **null**;

}

columnIndexWithD\_tFileInputDelimited\_4 = 8;

temp = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

**if** (temp.length() > 0) {

**try** {

norm\_checkouts.CheckoutYear = ParserUtils

.parseTo\_Integer(temp);

} **catch** (java.lang.Exception ex\_tFileInputDelimited\_4) {

rowstate\_tFileInputDelimited\_4

.setException(**new** RuntimeException(

String.format(

"Couldn't parse value for column '%s' in '%s', value is '%s'. Details: %s",

"CheckoutYear",

"norm\_checkouts",

temp,

ex\_tFileInputDelimited\_4),

ex\_tFileInputDelimited\_4));

}

} **else** {

norm\_checkouts.CheckoutYear = **null**;

}

columnIndexWithD\_tFileInputDelimited\_4 = 9;

norm\_checkouts.CheckoutTime = fid\_tFileInputDelimited\_4

.get(columnIndexWithD\_tFileInputDelimited\_4);

**if** (rowstate\_tFileInputDelimited\_4.getException() != **null**) {

**throw** rowstate\_tFileInputDelimited\_4

.getException();

}

} **catch** (java.lang.Exception e) {

whetherReject\_tFileInputDelimited\_4 = **true**;

System.err.println(e.getMessage());

norm\_checkouts = **null**;

}

/\*\*

\* [tFileInputDelimited\_4 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_4 main ] start

\*/

currentComponent = "tFileInputDelimited\_4";

tos\_count\_tFileInputDelimited\_4++;

/\*\*

\* [tFileInputDelimited\_4 main ] stop

\*/

/\*\*

\* [tFileInputDelimited\_4 process\_data\_begin ] start

\*/

currentComponent = "tFileInputDelimited\_4";

/\*\*

\* [tFileInputDelimited\_4 process\_data\_begin ] stop

\*/

// Start of branch "norm\_checkouts"

**if** (norm\_checkouts != **null**) {

/\*\*

\* [tMap\_5 main ] start

\*/

currentComponent = "tMap\_5";

// norm\_checkouts

// norm\_checkouts

**if** (execStat) {

runStat.updateStatOnConnection("norm\_checkouts"

+ iterateId, 1, 1);

}

**boolean** hasCasePrimitiveKeyWithNull\_tMap\_5 = **false**;

// ###############################

// # Input tables (lookups)

**boolean** rejectedInnerJoin\_tMap\_5 = **false**;

**boolean** mainRowRejected\_tMap\_5 = **false**;

// /////////////////////////////////////////////

// Starting Lookup Table "cleaned\_sdl\_inv"

// /////////////////////////////////////////////

**boolean** forceLoopcleaned\_sdl\_inv = **false**;

cleaned\_sdl\_invStruct cleaned\_sdl\_invObjectFromLookup = **null**;

**if** (!rejectedInnerJoin\_tMap\_5) { // G\_TM\_M\_020

hasCasePrimitiveKeyWithNull\_tMap\_5 = **false**;

cleaned\_sdl\_invHashKey.BibNum = norm\_checkouts.BibNumber;

cleaned\_sdl\_invHashKey.hashCodeDirty = **true**;

tHash\_Lookup\_cleaned\_sdl\_inv

.lookup(cleaned\_sdl\_invHashKey);

**if** (!tHash\_Lookup\_cleaned\_sdl\_inv.hasNext()) { // G\_TM\_M\_090

rejectedInnerJoin\_tMap\_5 = **true**;

} // G\_TM\_M\_090

} // G\_TM\_M\_020

**if** (tHash\_Lookup\_cleaned\_sdl\_inv != **null**

&& tHash\_Lookup\_cleaned\_sdl\_inv

.getCount(cleaned\_sdl\_invHashKey) > 1) { // G

// 071

// System.out.println("WARNING: UNIQUE MATCH is configured for the lookup 'cleaned\_sdl\_inv' and it contains more one result from keys : cleaned\_sdl\_inv.BibNum = '"

// + cleaned\_sdl\_invHashKey.BibNum + "'");

} // G 071

cleaned\_sdl\_invStruct cleaned\_sdl\_inv = **null**;

cleaned\_sdl\_invStruct fromLookup\_cleaned\_sdl\_inv = **null**;

cleaned\_sdl\_inv = cleaned\_sdl\_invDefault;

**if** (tHash\_Lookup\_cleaned\_sdl\_inv != **null**

&& tHash\_Lookup\_cleaned\_sdl\_inv.hasNext()) { // G

// 099

fromLookup\_cleaned\_sdl\_inv = tHash\_Lookup\_cleaned\_sdl\_inv

.next();

} // G 099

**if** (fromLookup\_cleaned\_sdl\_inv != **null**) {

cleaned\_sdl\_inv = fromLookup\_cleaned\_sdl\_inv;

}

// ###############################

{ // start of Var scope

// ###############################

// # Vars tables

Var\_\_tMap\_5\_\_Struct Var = Var\_\_tMap\_5;// ###############################

// ###############################

// # Output tables

validcheckouts = **null**;

invalidcheckouts = **null**;

**if** (!rejectedInnerJoin\_tMap\_5) {

// # Output table : 'validcheckouts'

validcheckouts\_tmp.ID = norm\_checkouts.ID;

validcheckouts\_tmp.BibNumber = norm\_checkouts.BibNumber;

validcheckouts\_tmp.ItemType = norm\_checkouts.ItemType;

validcheckouts\_tmp.Collection = norm\_checkouts.Collection;

validcheckouts\_tmp.CallNumber = norm\_checkouts.CallNumber;

validcheckouts\_tmp.CheckoutMonth = norm\_checkouts.CheckoutMonth;

validcheckouts\_tmp.CheckoutDate = norm\_checkouts.CheckoutDate;

validcheckouts\_tmp.CheckoutYear = (norm\_checkouts.CheckoutYear + "")

.contains(",") ? StringHandling

.EREPLACE(

(norm\_checkouts.CheckoutYear + ""),

",", "")

: (norm\_checkouts.CheckoutYear + "");

validcheckouts\_tmp.CheckoutTime = norm\_checkouts.CheckoutTime;

validcheckouts = validcheckouts\_tmp;

} // closing inner join bracket (1)

// ###### START REJECTS #####

// # Output reject table : 'invalidcheckouts'

// # Filter conditions

**if** (rejectedInnerJoin\_tMap\_5) {

invalidcheckouts\_tmp.ID = norm\_checkouts.ID;

invalidcheckouts\_tmp.BibNumber = norm\_checkouts.BibNumber;

invalidcheckouts\_tmp.ItemBarcode = norm\_checkouts.ItemBarcode;

invalidcheckouts\_tmp.ItemType = norm\_checkouts.ItemType;

invalidcheckouts\_tmp.Collection = norm\_checkouts.Collection;

invalidcheckouts\_tmp.CallNumber = norm\_checkouts.CallNumber;

invalidcheckouts\_tmp.Title = cleaned\_sdl\_inv.Title;

invalidcheckouts\_tmp.Subjects = cleaned\_sdl\_inv.Subjects;

invalidcheckouts\_tmp.Author = cleaned\_sdl\_inv.Author;

invalidcheckouts\_tmp.CheckoutMonth = norm\_checkouts.CheckoutMonth;

invalidcheckouts\_tmp.CheckoutDate = norm\_checkouts.CheckoutDate;

invalidcheckouts\_tmp.CheckoutYear = norm\_checkouts.CheckoutYear;

invalidcheckouts\_tmp.CheckoutTime = norm\_checkouts.CheckoutTime;

invalidcheckouts = invalidcheckouts\_tmp;

} // closing filter/reject

// ###############################

} // end of Var scope

rejectedInnerJoin\_tMap\_5 = **false**;

tos\_count\_tMap\_5++;

/\*\*

\* [tMap\_5 main ] stop

\*/

/\*\*

\* [tMap\_5 process\_data\_begin ] start

\*/

currentComponent = "tMap\_5";

/\*\*

\* [tMap\_5 process\_data\_begin ] stop

\*/

// Start of branch "validcheckouts"

**if** (validcheckouts != **null**) {

/\*\*

\* [tFileOutputDelimited\_5 main ] start

\*/

currentComponent = "tFileOutputDelimited\_5";

// validcheckouts

// validcheckouts

**if** (execStat) {

runStat.updateStatOnConnection(

"validcheckouts" + iterateId, 1, 1);

}

StringBuilder sb\_tFileOutputDelimited\_5 = **new** StringBuilder();

**if** (validcheckouts.ID != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.ID);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.BibNumber != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.BibNumber);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.ItemType != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.ItemType);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.Collection != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.Collection);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.CallNumber != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.CallNumber);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.CheckoutMonth != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.CheckoutMonth);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.CheckoutDate != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.CheckoutDate);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.CheckoutYear != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.CheckoutYear);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_tFileOutputDelimited\_5);

**if** (validcheckouts.CheckoutTime != **null**) {

sb\_tFileOutputDelimited\_5

.append(validcheckouts.CheckoutTime);

}

sb\_tFileOutputDelimited\_5

.append(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_5);

nb\_line\_tFileOutputDelimited\_5++;

resourceMap.put(

"nb\_line\_tFileOutputDelimited\_5",

nb\_line\_tFileOutputDelimited\_5);

outtFileOutputDelimited\_5

.write(sb\_tFileOutputDelimited\_5

.toString());

tos\_count\_tFileOutputDelimited\_5++;

/\*\*

\* [tFileOutputDelimited\_5 main ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_5 process\_data\_begin ]

\* start

\*/

currentComponent = "tFileOutputDelimited\_5";

/\*\*

\* [tFileOutputDelimited\_5 process\_data\_begin ]

\* stop

\*/

/\*\*

\* [tFileOutputDelimited\_5 process\_data\_end ]

\* start

\*/

currentComponent = "tFileOutputDelimited\_5";

/\*\*

\* [tFileOutputDelimited\_5 process\_data\_end ]

\* stop

\*/

} // End of branch "validcheckouts"

// Start of branch "invalidcheckouts"

**if** (invalidcheckouts != **null**) {

/\*\*

\* [tFileOutputDelimited\_6 main ] start

\*/

currentComponent = "tFileOutputDelimited\_6";

// invalidcheckouts

// invalidcheckouts

**if** (execStat) {

runStat.updateStatOnConnection(

"invalidcheckouts" + iterateId, 1,

1);

}

StringBuilder sb\_tFileOutputDelimited\_6 = **new** StringBuilder();

**if** (invalidcheckouts.ID != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.ID);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.BibNumber != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.BibNumber);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.ItemBarcode != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.ItemBarcode);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.ItemType != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.ItemType);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.Collection != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.Collection);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.CallNumber != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.CallNumber);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.Title != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.Title);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.Subjects != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.Subjects);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.Author != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.Author);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.CheckoutMonth != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.CheckoutMonth);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.CheckoutDate != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.CheckoutDate);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.CheckoutYear != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.CheckoutYear);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_tFileOutputDelimited\_6);

**if** (invalidcheckouts.CheckoutTime != **null**) {

sb\_tFileOutputDelimited\_6

.append(invalidcheckouts.CheckoutTime);

}

sb\_tFileOutputDelimited\_6

.append(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_6);

nb\_line\_tFileOutputDelimited\_6++;

resourceMap.put(

"nb\_line\_tFileOutputDelimited\_6",

nb\_line\_tFileOutputDelimited\_6);

outtFileOutputDelimited\_6

.write(sb\_tFileOutputDelimited\_6

.toString());

tos\_count\_tFileOutputDelimited\_6++;

/\*\*

\* [tFileOutputDelimited\_6 main ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_6 process\_data\_begin ]

\* start

\*/

currentComponent = "tFileOutputDelimited\_6";

/\*\*

\* [tFileOutputDelimited\_6 process\_data\_begin ]

\* stop

\*/

/\*\*

\* [tFileOutputDelimited\_6 process\_data\_end ]

\* start

\*/

currentComponent = "tFileOutputDelimited\_6";

/\*\*

\* [tFileOutputDelimited\_6 process\_data\_end ]

\* stop

\*/

} // End of branch "invalidcheckouts"

/\*\*

\* [tMap\_5 process\_data\_end ] start

\*/

currentComponent = "tMap\_5";

/\*\*

\* [tMap\_5 process\_data\_end ] stop

\*/

} // End of branch "norm\_checkouts"

/\*\*

\* [tFileInputDelimited\_4 process\_data\_end ] start

\*/

currentComponent = "tFileInputDelimited\_4";

/\*\*

\* [tFileInputDelimited\_4 process\_data\_end ] stop

\*/

/\*\*

\* [tFileInputDelimited\_4 end ] start

\*/

currentComponent = "tFileInputDelimited\_4";

}

} **finally** {

**if** (!((Object) ("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/checkouts/out\_accept.csv") **instanceof** java.io.InputStream)) {

**if** (fid\_tFileInputDelimited\_4 != **null**) {

fid\_tFileInputDelimited\_4.close();

}

}

**if** (fid\_tFileInputDelimited\_4 != **null**) {

globalMap.put("tFileInputDelimited\_4\_NB\_LINE",

fid\_tFileInputDelimited\_4.getRowNumber());

}

}

ok\_Hash.put("tFileInputDelimited\_4", **true**);

end\_Hash.put("tFileInputDelimited\_4",

System.currentTimeMillis());

/\*\*

\* [tFileInputDelimited\_4 end ] stop

\*/

/\*\*

\* [tMap\_5 end ] start

\*/

currentComponent = "tMap\_5";

// ###############################

// # Lookup hashes releasing

**if** (tHash\_Lookup\_cleaned\_sdl\_inv != **null**) {

tHash\_Lookup\_cleaned\_sdl\_inv.endGet();

}

globalMap.remove("tHash\_Lookup\_cleaned\_sdl\_inv");

// ###############################

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("norm\_checkouts"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tMap\_5", **true**);

end\_Hash.put("tMap\_5", System.currentTimeMillis());

/\*\*

\* [tMap\_5 end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_5 end ] start

\*/

currentComponent = "tFileOutputDelimited\_5";

**if** (outtFileOutputDelimited\_5 != **null**) {

outtFileOutputDelimited\_5.flush();

outtFileOutputDelimited\_5.close();

}

globalMap.put("tFileOutputDelimited\_5\_NB\_LINE",

nb\_line\_tFileOutputDelimited\_5);

globalMap.put("tFileOutputDelimited\_5\_FILE\_NAME",

fileName\_tFileOutputDelimited\_5);

resourceMap.put("finish\_tFileOutputDelimited\_5", **true**);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("validcheckouts"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tFileOutputDelimited\_5", **true**);

end\_Hash.put("tFileOutputDelimited\_5",

System.currentTimeMillis());

/\*\*

\* [tFileOutputDelimited\_5 end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_6 end ] start

\*/

currentComponent = "tFileOutputDelimited\_6";

**if** (outtFileOutputDelimited\_6 != **null**) {

outtFileOutputDelimited\_6.flush();

outtFileOutputDelimited\_6.close();

}

globalMap.put("tFileOutputDelimited\_6\_NB\_LINE",

nb\_line\_tFileOutputDelimited\_6);

globalMap.put("tFileOutputDelimited\_6\_FILE\_NAME",

fileName\_tFileOutputDelimited\_6);

resourceMap.put("finish\_tFileOutputDelimited\_6", **true**);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("invalidcheckouts"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tFileOutputDelimited\_6", **true**);

end\_Hash.put("tFileOutputDelimited\_6",

System.currentTimeMillis());

/\*\*

\* [tFileOutputDelimited\_6 end ] stop

\*/

}// end the resume

**if** (resumeEntryMethodName == **null** || globalResumeTicket) {

resumeUtil

.addLog("CHECKPOINT",

"CONNECTION:SUBJOB\_OK:tFileInputDelimited\_4:OnSubjobOk",

"", Thread.currentThread().getId() + "", "",

"", "", "", "");

}

**if** (execStat) {

runStat.updateStatOnConnection("OnSubjobOk4", 0, "ok");

}

tStatCatcher\_1Process(globalMap);

} **catch** (java.lang.Exception e) {

TalendException te = **new** TalendException(e, currentComponent,

globalMap);

**throw** te;

} **catch** (java.lang.Error error) {

runStat.stopThreadStat();

**throw** error;

} **finally** {

// free memory for "tMap\_5"

globalMap.remove("tHash\_Lookup\_cleaned\_sdl\_inv");

**try** {

/\*\*

\* [tFileInputDelimited\_4 finally ] start

\*/

currentComponent = "tFileInputDelimited\_4";

/\*\*

\* [tFileInputDelimited\_4 finally ] stop

\*/

/\*\*

\* [tMap\_5 finally ] start

\*/

currentComponent = "tMap\_5";

/\*\*

\* [tMap\_5 finally ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_5 finally ] start

\*/

currentComponent = "tFileOutputDelimited\_5";

**if** (resourceMap.get("finish\_tFileOutputDelimited\_5") == **null**) {

java.io.Writer outtFileOutputDelimited\_5 = (java.io.Writer) resourceMap

.get("out\_tFileOutputDelimited\_5");

**if** (outtFileOutputDelimited\_5 != **null**) {

outtFileOutputDelimited\_5.flush();

outtFileOutputDelimited\_5.close();

}

}

/\*\*

\* [tFileOutputDelimited\_5 finally ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_6 finally ] start

\*/

currentComponent = "tFileOutputDelimited\_6";

**if** (resourceMap.get("finish\_tFileOutputDelimited\_6") == **null**) {

java.io.Writer outtFileOutputDelimited\_6 = (java.io.Writer) resourceMap

.get("out\_tFileOutputDelimited\_6");

**if** (outtFileOutputDelimited\_6 != **null**) {

outtFileOutputDelimited\_6.flush();

outtFileOutputDelimited\_6.close();

}

}

/\*\*

\* [tFileOutputDelimited\_6 finally ] stop

\*/

} **catch** (java.lang.Exception e) {

// ignore

} **catch** (java.lang.Error error) {

// ignore

}

resourceMap = **null**;

}

globalMap.put("tFileInputDelimited\_4\_SUBPROCESS\_STATE", 1);

}

**public** **static** **class** statsStruct **implements**

routines.system.IPersistableRow<statsStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** java.util.Date moment;

**public** java.util.Date getMoment() {

**return** **this**.moment;

}

**public** String pid;

**public** String getPid() {

**return** **this**.pid;

}

**public** String father\_pid;

**public** String getFather\_pid() {

**return** **this**.father\_pid;

}

**public** String root\_pid;

**public** String getRoot\_pid() {

**return** **this**.root\_pid;

}

**public** Long system\_pid;

**public** Long getSystem\_pid() {

**return** **this**.system\_pid;

}

**public** String project;

**public** String getProject() {

**return** **this**.project;

}

**public** String job;

**public** String getJob() {

**return** **this**.job;

}

**public** String job\_repository\_id;

**public** String getJob\_repository\_id() {

**return** **this**.job\_repository\_id;

}

**public** String job\_version;

**public** String getJob\_version() {

**return** **this**.job\_version;

}

**public** String context;

**public** String getContext() {

**return** **this**.context;

}

**public** String origin;

**public** String getOrigin() {

**return** **this**.origin;

}

**public** String message\_type;

**public** String getMessage\_type() {

**return** **this**.message\_type;

}

**public** String message;

**public** String getMessage() {

**return** **this**.message;

}

**public** Long duration;

**public** Long getDuration() {

**return** **this**.duration;

}

**private** java.util.Date readDate(ObjectInputStream dis)

**throws** IOException {

java.util.Date dateReturn = **null**;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

dateReturn = **null**;

} **else** {

dateReturn = **new** Date(dis.readLong());

}

**return** dateReturn;

}

**private** **void** writeDate(java.util.Date date1, ObjectOutputStream dos)

**throws** IOException {

**if** (date1 == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeLong(date1.getTime());

}

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.moment = readDate(dis);

**this**.pid = readString(dis);

**this**.father\_pid = readString(dis);

**this**.root\_pid = readString(dis);

length = dis.readByte();

**if** (length == -1) {

**this**.system\_pid = **null**;

} **else** {

**this**.system\_pid = dis.readLong();

}

**this**.project = readString(dis);

**this**.job = readString(dis);

**this**.job\_repository\_id = readString(dis);

**this**.job\_version = readString(dis);

**this**.context = readString(dis);

**this**.origin = readString(dis);

**this**.message\_type = readString(dis);

**this**.message = readString(dis);

length = dis.readByte();

**if** (length == -1) {

**this**.duration = **null**;

} **else** {

**this**.duration = dis.readLong();

}

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// java.util.Date

writeDate(**this**.moment, dos);

// String

writeString(**this**.pid, dos);

// String

writeString(**this**.father\_pid, dos);

// String

writeString(**this**.root\_pid, dos);

// Long

**if** (**this**.system\_pid == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeLong(**this**.system\_pid);

}

// String

writeString(**this**.project, dos);

// String

writeString(**this**.job, dos);

// String

writeString(**this**.job\_repository\_id, dos);

// String

writeString(**this**.job\_version, dos);

// String

writeString(**this**.context, dos);

// String

writeString(**this**.origin, dos);

// String

writeString(**this**.message\_type, dos);

// String

writeString(**this**.message, dos);

// Long

**if** (**this**.duration == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeLong(**this**.duration);

}

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("moment=" + String.valueOf(moment));

sb.append(",pid=" + pid);

sb.append(",father\_pid=" + father\_pid);

sb.append(",root\_pid=" + root\_pid);

sb.append(",system\_pid=" + String.valueOf(system\_pid));

sb.append(",project=" + project);

sb.append(",job=" + job);

sb.append(",job\_repository\_id=" + job\_repository\_id);

sb.append(",job\_version=" + job\_version);

sb.append(",context=" + context);

sb.append(",origin=" + origin);

sb.append(",message\_type=" + message\_type);

sb.append(",message=" + message);

sb.append(",duration=" + String.valueOf(duration));

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(statsStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **void** tStatCatcher\_1Process(

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

globalMap.put("tStatCatcher\_1\_SUBPROCESS\_STATE", 0);

**final** **boolean** execStat = **this**.execStat;

String iterateId = "";

String currentComponent = "";

java.util.Map<String, Object> resourceMap = **new** java.util.HashMap<String, Object>();

**try** {

// TDI-39566 avoid throwing an useless Exception

**boolean** resumeIt = **true**;

**if** (globalResumeTicket == **false** && resumeEntryMethodName != **null**) {

String currentMethodName = **new** java.lang.Exception()

.getStackTrace()[0].getMethodName();

resumeIt = resumeEntryMethodName.equals(currentMethodName);

}

**if** (resumeIt || globalResumeTicket) { // start the resume

globalResumeTicket = **true**;

statsStruct stats = **new** statsStruct();

/\*\*

\* [tLogRow\_2 begin ] start

\*/

ok\_Hash.put("tLogRow\_2", **false**);

start\_Hash.put("tLogRow\_2", System.currentTimeMillis());

currentComponent = "tLogRow\_2";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("stats" + iterateId, 0,

0);

}

}

**int** tos\_count\_tLogRow\_2 = 0;

**class** BytesLimit65535\_tLogRow\_2 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tLogRow\_2().limitLog4jByte();

// /////////////////////

**final** String OUTPUT\_FIELD\_SEPARATOR\_tLogRow\_2 = "|";

java.io.PrintStream consoleOut\_tLogRow\_2 = **null**;

StringBuilder strBuffer\_tLogRow\_2 = **null**;

**int** nb\_line\_tLogRow\_2 = 0;

// /////////////////////

/\*\*

\* [tLogRow\_2 begin ] stop

\*/

/\*\*

\* [tStatCatcher\_1 begin ] start

\*/

ok\_Hash.put("tStatCatcher\_1", **false**);

start\_Hash.put("tStatCatcher\_1", System.currentTimeMillis());

currentComponent = "tStatCatcher\_1";

**int** tos\_count\_tStatCatcher\_1 = 0;

**class** BytesLimit65535\_tStatCatcher\_1 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tStatCatcher\_1().limitLog4jByte();

**for** (StatCatcherUtils.StatCatcherMessage scm : tStatCatcher\_1

.getMessages()) {

stats.pid = pid;

stats.root\_pid = rootPid;

stats.father\_pid = fatherPid;

stats.project = projectName;

stats.job = jobName;

stats.context = contextStr;

stats.origin = (scm.getOrigin() == **null**

|| scm.getOrigin().length() < 1 ? **null** : scm

.getOrigin());

stats.message = scm.getMessage();

stats.duration = scm.getDuration();

stats.moment = scm.getMoment();

stats.message\_type = scm.getMessageType();

stats.job\_version = scm.getJobVersion();

stats.job\_repository\_id = scm.getJobId();

stats.system\_pid = scm.getSystemPid();

/\*\*

\* [tStatCatcher\_1 begin ] stop

\*/

/\*\*

\* [tStatCatcher\_1 main ] start

\*/

currentComponent = "tStatCatcher\_1";

tos\_count\_tStatCatcher\_1++;

/\*\*

\* [tStatCatcher\_1 main ] stop

\*/

/\*\*

\* [tStatCatcher\_1 process\_data\_begin ] start

\*/

currentComponent = "tStatCatcher\_1";

/\*\*

\* [tStatCatcher\_1 process\_data\_begin ] stop

\*/

/\*\*

\* [tLogRow\_2 main ] start

\*/

currentComponent = "tLogRow\_2";

// stats

// stats

**if** (execStat) {

runStat.updateStatOnConnection("stats" + iterateId, 1,

1);

}

// /////////////////////

strBuffer\_tLogRow\_2 = **new** StringBuilder();

**if** (stats.moment != **null**) { //

strBuffer\_tLogRow\_2.append(FormatterUtils.format\_Date(

stats.moment, "yyyy-MM-dd HH:mm:ss"));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.pid != **null**) { //

strBuffer\_tLogRow\_2.append(String.valueOf(stats.pid));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.father\_pid != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.father\_pid));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.root\_pid != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.root\_pid));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.system\_pid != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.system\_pid));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.project != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.project));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.job != **null**) { //

strBuffer\_tLogRow\_2.append(String.valueOf(stats.job));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.job\_repository\_id != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.job\_repository\_id));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.job\_version != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.job\_version));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.context != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.context));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.origin != **null**) { //

strBuffer\_tLogRow\_2

.append(String.valueOf(stats.origin));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.message\_type != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.message\_type));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.message != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.message));

} //

strBuffer\_tLogRow\_2.append("|");

**if** (stats.duration != **null**) { //

strBuffer\_tLogRow\_2.append(String

.valueOf(stats.duration));

} //

**if** (globalMap.get("tLogRow\_CONSOLE") != **null**) {

consoleOut\_tLogRow\_2 = (java.io.PrintStream) globalMap

.get("tLogRow\_CONSOLE");

} **else** {

consoleOut\_tLogRow\_2 = **new** java.io.PrintStream(

**new** java.io.BufferedOutputStream(System.out));

globalMap.put("tLogRow\_CONSOLE", consoleOut\_tLogRow\_2);

}

consoleOut\_tLogRow\_2

.println(strBuffer\_tLogRow\_2.toString());

consoleOut\_tLogRow\_2.flush();

nb\_line\_tLogRow\_2++;

// ////

// ////

// /////////////////////

tos\_count\_tLogRow\_2++;

/\*\*

\* [tLogRow\_2 main ] stop

\*/

/\*\*

\* [tLogRow\_2 process\_data\_begin ] start

\*/

currentComponent = "tLogRow\_2";

/\*\*

\* [tLogRow\_2 process\_data\_begin ] stop

\*/

/\*\*

\* [tLogRow\_2 process\_data\_end ] start

\*/

currentComponent = "tLogRow\_2";

/\*\*

\* [tLogRow\_2 process\_data\_end ] stop

\*/

/\*\*

\* [tStatCatcher\_1 process\_data\_end ] start

\*/

currentComponent = "tStatCatcher\_1";

/\*\*

\* [tStatCatcher\_1 process\_data\_end ] stop

\*/

/\*\*

\* [tStatCatcher\_1 end ] start

\*/

currentComponent = "tStatCatcher\_1";

}

ok\_Hash.put("tStatCatcher\_1", **true**);

end\_Hash.put("tStatCatcher\_1", System.currentTimeMillis());

/\*\*

\* [tStatCatcher\_1 end ] stop

\*/

/\*\*

\* [tLogRow\_2 end ] start

\*/

currentComponent = "tLogRow\_2";

// ////

// ////

globalMap.put("tLogRow\_2\_NB\_LINE", nb\_line\_tLogRow\_2);

// /////////////////////

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("stats" + iterateId, 2,

0);

}

}

ok\_Hash.put("tLogRow\_2", **true**);

end\_Hash.put("tLogRow\_2", System.currentTimeMillis());

/\*\*

\* [tLogRow\_2 end ] stop

\*/

}// end the resume

} **catch** (java.lang.Exception e) {

TalendException te = **new** TalendException(e, currentComponent,

globalMap);

**throw** te;

} **catch** (java.lang.Error error) {

runStat.stopThreadStat();

**throw** error;

} **finally** {

**try** {

/\*\*

\* [tStatCatcher\_1 finally ] start

\*/

currentComponent = "tStatCatcher\_1";

/\*\*

\* [tStatCatcher\_1 finally ] stop

\*/

/\*\*

\* [tLogRow\_2 finally ] start

\*/

currentComponent = "tLogRow\_2";

/\*\*

\* [tLogRow\_2 finally ] stop

\*/

} **catch** (java.lang.Exception e) {

// ignore

} **catch** (java.lang.Error error) {

// ignore

}

resourceMap = **null**;

}

globalMap.put("tStatCatcher\_1\_SUBPROCESS\_STATE", 1);

}

**public** **static** **class** cleaned\_dict\_lookupStruct

**implements**

routines.system.IPersistableComparableLookupRow<cleaned\_dict\_lookupStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**protected** **static** **final** **int** DEFAULT\_HASHCODE = 1;

**protected** **static** **final** **int** PRIME = 31;

**protected** **int** hashCode = DEFAULT\_HASHCODE;

**public** **boolean** hashCodeDirty = **true**;

**public** String loopKey;

**public** String Code\_ID;

**public** String getCode\_ID() {

**return** **this**.Code\_ID;

}

**public** String Code;

**public** String getCode() {

**return** **this**.Code;

}

**public** String Description;

**public** String getDescription() {

**return** **this**.Description;

}

**public** String Format\_Group;

**public** String getFormat\_Group() {

**return** **this**.Format\_Group;

}

**public** String Format\_Subgroup;

**public** String getFormat\_Subgroup() {

**return** **this**.Format\_Subgroup;

}

**public** String Category\_Group;

**public** String getCategory\_Group() {

**return** **this**.Category\_Group;

}

**public** String Category\_Subgroup;

**public** String getCategory\_Subgroup() {

**return** **this**.Category\_Subgroup;

}

@Override

**public** **int** hashCode() {

**if** (**this**.hashCodeDirty) {

**final** **int** prime = PRIME;

**int** result = DEFAULT\_HASHCODE;

result = prime \* result

+ ((**this**.Code == **null**) ? 0 : **this**.Code.hashCode());

**this**.hashCode = result;

**this**.hashCodeDirty = **false**;

}

**return** **this**.hashCode;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

**final** cleaned\_dict\_lookupStruct other = (cleaned\_dict\_lookupStruct) obj;

**if** (**this**.Code == **null**) {

**if** (other.Code != **null**)

**return** **false**;

} **else** **if** (!**this**.Code.equals(other.Code))

**return** **false**;

**return** **true**;

}

**public** **void** copyDataTo(cleaned\_dict\_lookupStruct other) {

other.Code\_ID = **this**.Code\_ID;

other.Code = **this**.Code;

other.Description = **this**.Description;

other.Format\_Group = **this**.Format\_Group;

other.Format\_Subgroup = **this**.Format\_Subgroup;

other.Category\_Group = **this**.Category\_Group;

other.Category\_Subgroup = **this**.Category\_Subgroup;

}

**public** **void** copyKeysDataTo(cleaned\_dict\_lookupStruct other) {

other.Code = **this**.Code;

}

**private** String readString(DataInputStream dis, ObjectInputStream ois)

**throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**byte**[] byteArray = **new** **byte**[length];

dis.read(byteArray);

strReturn = **new** String(byteArray, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, DataOutputStream dos,

ObjectOutputStream oos) **throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readKeysData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.Code = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeKeysData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.Code, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

/\*\*

\* Fill Values data by reading ObjectInputStream.

\*/

**public** **void** readValuesData(DataInputStream dis, ObjectInputStream ois) {

**try** {

**int** length = 0;

**this**.Code\_ID = readString(dis, ois);

**this**.Description = readString(dis, ois);

**this**.Format\_Group = readString(dis, ois);

**this**.Format\_Subgroup = readString(dis, ois);

**this**.Category\_Group = readString(dis, ois);

**this**.Category\_Subgroup = readString(dis, ois);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

/\*\*

\* Return a byte array which represents Values data.

\*/

**public** **void** writeValuesData(DataOutputStream dos, ObjectOutputStream oos) {

**try** {

writeString(**this**.Code\_ID, dos, oos);

writeString(**this**.Description, dos, oos);

writeString(**this**.Format\_Group, dos, oos);

writeString(**this**.Format\_Subgroup, dos, oos);

writeString(**this**.Category\_Group, dos, oos);

writeString(**this**.Category\_Subgroup, dos, oos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("Code\_ID=" + Code\_ID);

sb.append(",Code=" + Code);

sb.append(",Description=" + Description);

sb.append(",Format\_Group=" + Format\_Group);

sb.append(",Format\_Subgroup=" + Format\_Subgroup);

sb.append(",Category\_Group=" + Category\_Group);

sb.append(",Category\_Subgroup=" + Category\_Subgroup);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(cleaned\_dict\_lookupStruct other) {

**int** returnValue = -1;

returnValue = checkNullsAndCompare(**this**.Code, other.Code);

**if** (returnValue != 0) {

**return** returnValue;

}

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** cleaned\_data\_dictStruct **implements**

routines.system.IPersistableRow<cleaned\_data\_dictStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String Code\_ID;

**public** String getCode\_ID() {

**return** **this**.Code\_ID;

}

**public** String Code;

**public** String getCode() {

**return** **this**.Code;

}

**public** String Description;

**public** String getDescription() {

**return** **this**.Description;

}

**public** String Format\_Group;

**public** String getFormat\_Group() {

**return** **this**.Format\_Group;

}

**public** String Format\_Subgroup;

**public** String getFormat\_Subgroup() {

**return** **this**.Format\_Subgroup;

}

**public** String Category\_Group;

**public** String getCategory\_Group() {

**return** **this**.Category\_Group;

}

**public** String Category\_Subgroup;

**public** String getCategory\_Subgroup() {

**return** **this**.Category\_Subgroup;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.Code\_ID = readString(dis);

**this**.Code = readString(dis);

**this**.Description = readString(dis);

**this**.Format\_Group = readString(dis);

**this**.Format\_Subgroup = readString(dis);

**this**.Category\_Group = readString(dis);

**this**.Category\_Subgroup = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.Code\_ID, dos);

// String

writeString(**this**.Code, dos);

// String

writeString(**this**.Description, dos);

// String

writeString(**this**.Format\_Group, dos);

// String

writeString(**this**.Format\_Subgroup, dos);

// String

writeString(**this**.Category\_Group, dos);

// String

writeString(**this**.Category\_Subgroup, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("Code\_ID=" + Code\_ID);

sb.append(",Code=" + Code);

sb.append(",Description=" + Description);

sb.append(",Format\_Group=" + Format\_Group);

sb.append(",Format\_Subgroup=" + Format\_Subgroup);

sb.append(",Category\_Group=" + Category\_Group);

sb.append(",Category\_Subgroup=" + Category\_Subgroup);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(cleaned\_data\_dictStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** deduped\_raw\_data\_dictStruct **implements**

routines.system.IPersistableRow<deduped\_raw\_data\_dictStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String Code;

**public** String getCode() {

**return** **this**.Code;

}

**public** String Description;

**public** String getDescription() {

**return** **this**.Description;

}

**public** String Code\_Type;

**public** String getCode\_Type() {

**return** **this**.Code\_Type;

}

**public** String Format\_Group;

**public** String getFormat\_Group() {

**return** **this**.Format\_Group;

}

**public** String Format\_Subgroup;

**public** String getFormat\_Subgroup() {

**return** **this**.Format\_Subgroup;

}

**public** String Category\_Group;

**public** String getCategory\_Group() {

**return** **this**.Category\_Group;

}

**public** String Category\_Subgroup;

**public** String getCategory\_Subgroup() {

**return** **this**.Category\_Subgroup;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.Code = readString(dis);

**this**.Description = readString(dis);

**this**.Code\_Type = readString(dis);

**this**.Format\_Group = readString(dis);

**this**.Format\_Subgroup = readString(dis);

**this**.Category\_Group = readString(dis);

**this**.Category\_Subgroup = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.Code, dos);

// String

writeString(**this**.Description, dos);

// String

writeString(**this**.Code\_Type, dos);

// String

writeString(**this**.Format\_Group, dos);

// String

writeString(**this**.Format\_Subgroup, dos);

// String

writeString(**this**.Category\_Group, dos);

// String

writeString(**this**.Category\_Subgroup, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("Code=" + Code);

sb.append(",Description=" + Description);

sb.append(",Code\_Type=" + Code\_Type);

sb.append(",Format\_Group=" + Format\_Group);

sb.append(",Format\_Subgroup=" + Format\_Subgroup);

sb.append(",Category\_Group=" + Category\_Group);

sb.append(",Category\_Subgroup=" + Category\_Subgroup);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(deduped\_raw\_data\_dictStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** raw\_data\_dictStruct **implements**

routines.system.IPersistableRow<raw\_data\_dictStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String Code;

**public** String getCode() {

**return** **this**.Code;

}

**public** String Description;

**public** String getDescription() {

**return** **this**.Description;

}

**public** String Code\_Type;

**public** String getCode\_Type() {

**return** **this**.Code\_Type;

}

**public** String Format\_Group;

**public** String getFormat\_Group() {

**return** **this**.Format\_Group;

}

**public** String Format\_Subgroup;

**public** String getFormat\_Subgroup() {

**return** **this**.Format\_Subgroup;

}

**public** String Category\_Group;

**public** String getCategory\_Group() {

**return** **this**.Category\_Group;

}

**public** String Category\_Subgroup;

**public** String getCategory\_Subgroup() {

**return** **this**.Category\_Subgroup;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.Code = readString(dis);

**this**.Description = readString(dis);

**this**.Code\_Type = readString(dis);

**this**.Format\_Group = readString(dis);

**this**.Format\_Subgroup = readString(dis);

**this**.Category\_Group = readString(dis);

**this**.Category\_Subgroup = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.Code, dos);

// String

writeString(**this**.Description, dos);

// String

writeString(**this**.Code\_Type, dos);

// String

writeString(**this**.Format\_Group, dos);

// String

writeString(**this**.Format\_Subgroup, dos);

// String

writeString(**this**.Category\_Group, dos);

// String

writeString(**this**.Category\_Subgroup, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("Code=" + Code);

sb.append(",Description=" + Description);

sb.append(",Code\_Type=" + Code\_Type);

sb.append(",Format\_Group=" + Format\_Group);

sb.append(",Format\_Subgroup=" + Format\_Subgroup);

sb.append(",Category\_Group=" + Category\_Group);

sb.append(",Category\_Subgroup=" + Category\_Subgroup);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(raw\_data\_dictStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **void** tFileInputDelimited\_2Process(

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

globalMap.put("tFileInputDelimited\_2\_SUBPROCESS\_STATE", 0);

**final** **boolean** execStat = **this**.execStat;

String iterateId = "";

String currentComponent = "";

java.util.Map<String, Object> resourceMap = **new** java.util.HashMap<String, Object>();

**try** {

// TDI-39566 avoid throwing an useless Exception

**boolean** resumeIt = **true**;

**if** (globalResumeTicket == **false** && resumeEntryMethodName != **null**) {

String currentMethodName = **new** java.lang.Exception()

.getStackTrace()[0].getMethodName();

resumeIt = resumeEntryMethodName.equals(currentMethodName);

}

**if** (resumeIt || globalResumeTicket) { // start the resume

globalResumeTicket = **true**;

raw\_data\_dictStruct raw\_data\_dict = **new** raw\_data\_dictStruct();

deduped\_raw\_data\_dictStruct deduped\_raw\_data\_dict = **new** deduped\_raw\_data\_dictStruct();

cleaned\_data\_dictStruct cleaned\_data\_dict = **new** cleaned\_data\_dictStruct();

cleaned\_data\_dictStruct cleaned\_dict\_lookup = cleaned\_data\_dict;

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup begin ] start

\*/

ok\_Hash.put("tAdvancedHash\_cleaned\_dict\_lookup", **false**);

start\_Hash.put("tAdvancedHash\_cleaned\_dict\_lookup",

System.currentTimeMillis());

currentComponent = "tAdvancedHash\_cleaned\_dict\_lookup";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("cleaned\_dict\_lookup"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tAdvancedHash\_cleaned\_dict\_lookup = 0;

**class** BytesLimit65535\_tAdvancedHash\_cleaned\_dict\_lookup {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tAdvancedHash\_cleaned\_dict\_lookup()

.limitLog4jByte();

// connection name:cleaned\_dict\_lookup

// source node:tFileOutputDelimited\_2 -

// inputs:(cleaned\_data\_dict)

// outputs:(cleaned\_dict\_lookup,cleaned\_dict\_lookup) | target

// node:tAdvancedHash\_cleaned\_dict\_lookup -

// inputs:(cleaned\_dict\_lookup) outputs:()

// linked node: tMap\_1 -

// inputs:(raw\_checkouts,cleaned\_dict\_lookup)

// outputs:(normalized\_checkouts,invalid\_checkouts)

org.talend.designer.components.lookup.common.ICommonLookup.MATCHING\_MODE matchingModeEnum\_cleaned\_dict\_lookup = org.talend.designer.components.lookup.common.ICommonLookup.MATCHING\_MODE.ALL\_MATCHES;

org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_dict\_lookupStruct> tHash\_Lookup\_cleaned\_dict\_lookup = org.talend.designer.components.lookup.memory.AdvancedMemoryLookup

.<cleaned\_dict\_lookupStruct> getLookup(matchingModeEnum\_cleaned\_dict\_lookup);

globalMap.put("tHash\_Lookup\_cleaned\_dict\_lookup",

tHash\_Lookup\_cleaned\_dict\_lookup);

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup begin ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_2 begin ] start

\*/

ok\_Hash.put("tFileOutputDelimited\_2", **false**);

start\_Hash.put("tFileOutputDelimited\_2",

System.currentTimeMillis());

currentComponent = "tFileOutputDelimited\_2";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("cleaned\_data\_dict"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tFileOutputDelimited\_2 = 0;

**class** BytesLimit65535\_tFileOutputDelimited\_2 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileOutputDelimited\_2().limitLog4jByte();

String fileName\_tFileOutputDelimited\_2 = "";

fileName\_tFileOutputDelimited\_2 = (**new** java.io.File(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/datadict/out.csv"))

.getAbsolutePath().replace("\\", "/");

String fullName\_tFileOutputDelimited\_2 = **null**;

String extension\_tFileOutputDelimited\_2 = **null**;

String directory\_tFileOutputDelimited\_2 = **null**;

**if** ((fileName\_tFileOutputDelimited\_2.indexOf("/") != -1)) {

**if** (fileName\_tFileOutputDelimited\_2.lastIndexOf(".") < fileName\_tFileOutputDelimited\_2

.lastIndexOf("/")) {

fullName\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2;

extension\_tFileOutputDelimited\_2 = "";

} **else** {

fullName\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2

.substring(0, fileName\_tFileOutputDelimited\_2

.lastIndexOf("."));

extension\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2

.substring(fileName\_tFileOutputDelimited\_2

.lastIndexOf("."));

}

directory\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2

.substring(0, fileName\_tFileOutputDelimited\_2

.lastIndexOf("/"));

} **else** {

**if** (fileName\_tFileOutputDelimited\_2.lastIndexOf(".") != -1) {

fullName\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2

.substring(0, fileName\_tFileOutputDelimited\_2

.lastIndexOf("."));

extension\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2

.substring(fileName\_tFileOutputDelimited\_2

.lastIndexOf("."));

} **else** {

fullName\_tFileOutputDelimited\_2 = fileName\_tFileOutputDelimited\_2;

extension\_tFileOutputDelimited\_2 = "";

}

directory\_tFileOutputDelimited\_2 = "";

}

**boolean** isFileGenerated\_tFileOutputDelimited\_2 = **true**;

java.io.File filetFileOutputDelimited\_2 = **new** java.io.File(

fileName\_tFileOutputDelimited\_2);

globalMap.put("tFileOutputDelimited\_2\_FILE\_NAME",

fileName\_tFileOutputDelimited\_2);

**int** nb\_line\_tFileOutputDelimited\_2 = 0;

**int** splitedFileNo\_tFileOutputDelimited\_2 = 0;

**int** currentRow\_tFileOutputDelimited\_2 = 0;

**final** String OUT\_DELIM\_tFileOutputDelimited\_2 = /\*\*

\* Start field

\* tFileOutputDelimited\_2:FIELDSEPARATOR

\*/

";"/\*\* End field tFileOutputDelimited\_2:FIELDSEPARATOR \*/

;

**final** String OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_2 = /\*\*

\* Start

\* field tFileOutputDelimited\_2:ROWSEPARATOR

\*/

"\n"/\*\* End field tFileOutputDelimited\_2:ROWSEPARATOR \*/

;

// create directory only if not exists

**if** (directory\_tFileOutputDelimited\_2 != **null**

&& directory\_tFileOutputDelimited\_2.trim().length() != 0) {

java.io.File dir\_tFileOutputDelimited\_2 = **new** java.io.File(

directory\_tFileOutputDelimited\_2);

**if** (!dir\_tFileOutputDelimited\_2.exists()) {

dir\_tFileOutputDelimited\_2.mkdirs();

}

}

// routines.system.Row

java.io.Writer outtFileOutputDelimited\_2 = **null**;

java.io.File fileToDelete\_tFileOutputDelimited\_2 = **new** java.io.File(

fileName\_tFileOutputDelimited\_2);

**if** (fileToDelete\_tFileOutputDelimited\_2.exists()) {

fileToDelete\_tFileOutputDelimited\_2.delete();

}

outtFileOutputDelimited\_2 = **new** java.io.BufferedWriter(

**new** java.io.OutputStreamWriter(

**new** java.io.FileOutputStream(

fileName\_tFileOutputDelimited\_2, **false**),

"ISO-8859-15"));

**if** (filetFileOutputDelimited\_2.length() == 0) {

outtFileOutputDelimited\_2.write("Code\_ID");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.write("Code");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.write("Description");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.write("Format\_Group");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.write("Format\_Subgroup");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.write("Category\_Group");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.write("Category\_Subgroup");

outtFileOutputDelimited\_2

.write(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2.flush();

}

resourceMap.put("out\_tFileOutputDelimited\_2",

outtFileOutputDelimited\_2);

resourceMap.put("nb\_line\_tFileOutputDelimited\_2",

nb\_line\_tFileOutputDelimited\_2);

/\*\*

\* [tFileOutputDelimited\_2 begin ] stop

\*/

/\*\*

\* [tMap\_3 begin ] start

\*/

ok\_Hash.put("tMap\_3", **false**);

start\_Hash.put("tMap\_3", System.currentTimeMillis());

currentComponent = "tMap\_3";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("deduped\_raw\_data\_dict"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tMap\_3 = 0;

**class** BytesLimit65535\_tMap\_3 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tMap\_3().limitLog4jByte();

// ###############################

// # Lookup's keys initialization

// ###############################

// ###############################

// # Vars initialization

**class** Var\_\_tMap\_3\_\_Struct {

}

Var\_\_tMap\_3\_\_Struct Var\_\_tMap\_3 = **new** Var\_\_tMap\_3\_\_Struct();

// ###############################

// ###############################

// # Outputs initialization

cleaned\_data\_dictStruct cleaned\_data\_dict\_tmp = **new** cleaned\_data\_dictStruct();

// ###############################

/\*\*

\* [tMap\_3 begin ] stop

\*/

/\*\*

\* [tUniqRow\_1 begin ] start

\*/

ok\_Hash.put("tUniqRow\_1", **false**);

start\_Hash.put("tUniqRow\_1", System.currentTimeMillis());

currentComponent = "tUniqRow\_1";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("raw\_data\_dict"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tUniqRow\_1 = 0;

**class** BytesLimit65535\_tUniqRow\_1 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tUniqRow\_1().limitLog4jByte();

**class** KeyStruct\_tUniqRow\_1 {

**private** **static** **final** **int** DEFAULT\_HASHCODE = 1;

**private** **static** **final** **int** PRIME = 31;

**private** **int** hashCode = DEFAULT\_HASHCODE;

**public** **boolean** hashCodeDirty = **true**;

String Code;

@Override

**public** **int** hashCode() {

**if** (**this**.hashCodeDirty) {

**final** **int** prime = PRIME;

**int** result = DEFAULT\_HASHCODE;

result = prime

\* result

+ ((**this**.Code == **null**) ? 0 : **this**.Code

.hashCode());

**this**.hashCode = result;

**this**.hashCodeDirty = **false**;

}

**return** **this**.hashCode;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

**final** KeyStruct\_tUniqRow\_1 other = (KeyStruct\_tUniqRow\_1) obj;

**if** (**this**.Code == **null**) {

**if** (other.Code != **null**)

**return** **false**;

} **else** **if** (!**this**.Code.equals(other.Code))

**return** **false**;

**return** **true**;

}

}

**int** nb\_uniques\_tUniqRow\_1 = 0;

**int** nb\_duplicates\_tUniqRow\_1 = 0;

KeyStruct\_tUniqRow\_1 finder\_tUniqRow\_1 = **new** KeyStruct\_tUniqRow\_1();

java.util.Set<KeyStruct\_tUniqRow\_1> keystUniqRow\_1 = **new** java.util.HashSet<KeyStruct\_tUniqRow\_1>();

/\*\*

\* [tUniqRow\_1 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_2 begin ] start

\*/

ok\_Hash.put("tFileInputDelimited\_2", **false**);

start\_Hash.put("tFileInputDelimited\_2",

System.currentTimeMillis());

currentComponent = "tFileInputDelimited\_2";

**int** tos\_count\_tFileInputDelimited\_2 = 0;

**class** BytesLimit65535\_tFileInputDelimited\_2 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileInputDelimited\_2().limitLog4jByte();

**final** routines.system.RowState rowstate\_tFileInputDelimited\_2 = **new** routines.system.RowState();

**int** nb\_line\_tFileInputDelimited\_2 = 0;

org.talend.fileprocess.FileInputDelimited fid\_tFileInputDelimited\_2 = **null**;

**try** {

Object filename\_tFileInputDelimited\_2 = "E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/Integrated\_Library\_System\_\_ILS\_\_Data\_Dictionary.csv";

**if** (filename\_tFileInputDelimited\_2 **instanceof** java.io.InputStream) {

**int** footer\_value\_tFileInputDelimited\_2 = 0, random\_value\_tFileInputDelimited\_2 = -1;

**if** (footer\_value\_tFileInputDelimited\_2 > 0

|| random\_value\_tFileInputDelimited\_2 > 0) {

**throw** **new** java.lang.Exception(

"When the input source is a stream,footer and random shouldn't be bigger than 0.");

}

}

**try** {

fid\_tFileInputDelimited\_2 = **new** org.talend.fileprocess.FileInputDelimited(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/Integrated\_Library\_System\_\_ILS\_\_Data\_Dictionary.csv",

"US-ASCII", ",", "\n", **false**, 1, 0, -1, -1,

**false**);

} **catch** (java.lang.Exception e) {

System.err.println(e.getMessage());

}

**while** (fid\_tFileInputDelimited\_2 != **null**

&& fid\_tFileInputDelimited\_2.nextRecord()) {

rowstate\_tFileInputDelimited\_2.reset();

raw\_data\_dict = **null**;

**boolean** whetherReject\_tFileInputDelimited\_2 = **false**;

raw\_data\_dict = **new** raw\_data\_dictStruct();

**try** {

**int** columnIndexWithD\_tFileInputDelimited\_2 = 0;

columnIndexWithD\_tFileInputDelimited\_2 = 0;

raw\_data\_dict.Code = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

columnIndexWithD\_tFileInputDelimited\_2 = 1;

raw\_data\_dict.Description = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

columnIndexWithD\_tFileInputDelimited\_2 = 2;

raw\_data\_dict.Code\_Type = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

columnIndexWithD\_tFileInputDelimited\_2 = 3;

raw\_data\_dict.Format\_Group = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

columnIndexWithD\_tFileInputDelimited\_2 = 4;

raw\_data\_dict.Format\_Subgroup = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

columnIndexWithD\_tFileInputDelimited\_2 = 5;

raw\_data\_dict.Category\_Group = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

columnIndexWithD\_tFileInputDelimited\_2 = 6;

raw\_data\_dict.Category\_Subgroup = fid\_tFileInputDelimited\_2

.get(columnIndexWithD\_tFileInputDelimited\_2);

**if** (rowstate\_tFileInputDelimited\_2.getException() != **null**) {

**throw** rowstate\_tFileInputDelimited\_2

.getException();

}

} **catch** (java.lang.Exception e) {

whetherReject\_tFileInputDelimited\_2 = **true**;

System.err.println(e.getMessage());

raw\_data\_dict = **null**;

}

/\*\*

\* [tFileInputDelimited\_2 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_2 main ] start

\*/

currentComponent = "tFileInputDelimited\_2";

tos\_count\_tFileInputDelimited\_2++;

/\*\*

\* [tFileInputDelimited\_2 main ] stop

\*/

/\*\*

\* [tFileInputDelimited\_2 process\_data\_begin ] start

\*/

currentComponent = "tFileInputDelimited\_2";

/\*\*

\* [tFileInputDelimited\_2 process\_data\_begin ] stop

\*/

// Start of branch "raw\_data\_dict"

**if** (raw\_data\_dict != **null**) {

/\*\*

\* [tUniqRow\_1 main ] start

\*/

currentComponent = "tUniqRow\_1";

// raw\_data\_dict

// raw\_data\_dict

**if** (execStat) {

runStat.updateStatOnConnection("raw\_data\_dict"

+ iterateId, 1, 1);

}

deduped\_raw\_data\_dict = **null**;

**if** (raw\_data\_dict.Code == **null**) {

finder\_tUniqRow\_1.Code = **null**;

} **else** {

finder\_tUniqRow\_1.Code = raw\_data\_dict.Code

.toLowerCase();

}

finder\_tUniqRow\_1.hashCodeDirty = **true**;

**if** (!keystUniqRow\_1.contains(finder\_tUniqRow\_1)) {

KeyStruct\_tUniqRow\_1 new\_tUniqRow\_1 = **new** KeyStruct\_tUniqRow\_1();

**if** (raw\_data\_dict.Code == **null**) {

new\_tUniqRow\_1.Code = **null**;

} **else** {

new\_tUniqRow\_1.Code = raw\_data\_dict.Code

.toLowerCase();

}

keystUniqRow\_1.add(new\_tUniqRow\_1);

**if** (deduped\_raw\_data\_dict == **null**) {

deduped\_raw\_data\_dict = **new** deduped\_raw\_data\_dictStruct();

}

deduped\_raw\_data\_dict.Code = raw\_data\_dict.Code;

deduped\_raw\_data\_dict.Description = raw\_data\_dict.Description;

deduped\_raw\_data\_dict.Code\_Type = raw\_data\_dict.Code\_Type;

deduped\_raw\_data\_dict.Format\_Group = raw\_data\_dict.Format\_Group;

deduped\_raw\_data\_dict.Format\_Subgroup = raw\_data\_dict.Format\_Subgroup;

deduped\_raw\_data\_dict.Category\_Group = raw\_data\_dict.Category\_Group;

deduped\_raw\_data\_dict.Category\_Subgroup = raw\_data\_dict.Category\_Subgroup;

nb\_uniques\_tUniqRow\_1++;

} **else** {

nb\_duplicates\_tUniqRow\_1++;

}

tos\_count\_tUniqRow\_1++;

/\*\*

\* [tUniqRow\_1 main ] stop

\*/

/\*\*

\* [tUniqRow\_1 process\_data\_begin ] start

\*/

currentComponent = "tUniqRow\_1";

/\*\*

\* [tUniqRow\_1 process\_data\_begin ] stop

\*/

// Start of branch "deduped\_raw\_data\_dict"

**if** (deduped\_raw\_data\_dict != **null**) {

/\*\*

\* [tMap\_3 main ] start

\*/

currentComponent = "tMap\_3";

// deduped\_raw\_data\_dict

// deduped\_raw\_data\_dict

**if** (execStat) {

runStat.updateStatOnConnection(

"deduped\_raw\_data\_dict" + iterateId,

1, 1);

}

**boolean** hasCasePrimitiveKeyWithNull\_tMap\_3 = **false**;

// ###############################

// # Input tables (lookups)

**boolean** rejectedInnerJoin\_tMap\_3 = **false**;

**boolean** mainRowRejected\_tMap\_3 = **false**;

// ###############################

{ // start of Var scope

// ###############################

// # Vars tables

Var\_\_tMap\_3\_\_Struct Var = Var\_\_tMap\_3;// ###############################

// ###############################

// # Output tables

cleaned\_data\_dict = **null**;

// # Output table : 'cleaned\_data\_dict'

cleaned\_data\_dict\_tmp.Code\_ID = Numeric

.sequence("s2", 1, 1) + "";

cleaned\_data\_dict\_tmp.Code = deduped\_raw\_data\_dict.Code;

cleaned\_data\_dict\_tmp.Description = deduped\_raw\_data\_dict.Description;

cleaned\_data\_dict\_tmp.Format\_Group = deduped\_raw\_data\_dict.Format\_Group;

cleaned\_data\_dict\_tmp.Format\_Subgroup = deduped\_raw\_data\_dict.Format\_Subgroup;

cleaned\_data\_dict\_tmp.Category\_Group = deduped\_raw\_data\_dict.Category\_Group;

cleaned\_data\_dict\_tmp.Category\_Subgroup = deduped\_raw\_data\_dict.Category\_Subgroup;

cleaned\_data\_dict = cleaned\_data\_dict\_tmp;

// ###############################

} // end of Var scope

rejectedInnerJoin\_tMap\_3 = **false**;

tos\_count\_tMap\_3++;

/\*\*

\* [tMap\_3 main ] stop

\*/

/\*\*

\* [tMap\_3 process\_data\_begin ] start

\*/

currentComponent = "tMap\_3";

/\*\*

\* [tMap\_3 process\_data\_begin ] stop

\*/

// Start of branch "cleaned\_data\_dict"

**if** (cleaned\_data\_dict != **null**) {

/\*\*

\* [tFileOutputDelimited\_2 main ] start

\*/

currentComponent = "tFileOutputDelimited\_2";

// cleaned\_data\_dict

// cleaned\_data\_dict

**if** (execStat) {

runStat.updateStatOnConnection(

"cleaned\_data\_dict" + iterateId,

1, 1);

}

StringBuilder sb\_tFileOutputDelimited\_2 = **new** StringBuilder();

**if** (cleaned\_data\_dict.Code\_ID != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Code\_ID);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_tFileOutputDelimited\_2);

**if** (cleaned\_data\_dict.Code != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Code);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_tFileOutputDelimited\_2);

**if** (cleaned\_data\_dict.Description != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Description);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_tFileOutputDelimited\_2);

**if** (cleaned\_data\_dict.Format\_Group != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Format\_Group);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_tFileOutputDelimited\_2);

**if** (cleaned\_data\_dict.Format\_Subgroup != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Format\_Subgroup);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_tFileOutputDelimited\_2);

**if** (cleaned\_data\_dict.Category\_Group != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Category\_Group);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_tFileOutputDelimited\_2);

**if** (cleaned\_data\_dict.Category\_Subgroup != **null**) {

sb\_tFileOutputDelimited\_2

.append(cleaned\_data\_dict.Category\_Subgroup);

}

sb\_tFileOutputDelimited\_2

.append(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_2);

nb\_line\_tFileOutputDelimited\_2++;

resourceMap.put(

"nb\_line\_tFileOutputDelimited\_2",

nb\_line\_tFileOutputDelimited\_2);

outtFileOutputDelimited\_2

.write(sb\_tFileOutputDelimited\_2

.toString());

cleaned\_dict\_lookup = cleaned\_data\_dict;

tos\_count\_tFileOutputDelimited\_2++;

/\*\*

\* [tFileOutputDelimited\_2 main ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_2

\* process\_data\_begin ] start

\*/

currentComponent = "tFileOutputDelimited\_2";

/\*\*

\* [tFileOutputDelimited\_2

\* process\_data\_begin ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup main ]

\* start

\*/

currentComponent = "tAdvancedHash\_cleaned\_dict\_lookup";

// cleaned\_dict\_lookup

// cleaned\_dict\_lookup

**if** (execStat) {

runStat.updateStatOnConnection(

"cleaned\_dict\_lookup"

+ iterateId, 1, 1);

}

cleaned\_dict\_lookupStruct cleaned\_dict\_lookup\_HashRow = **new** cleaned\_dict\_lookupStruct();

cleaned\_dict\_lookup\_HashRow.Code\_ID = cleaned\_dict\_lookup.Code\_ID;

cleaned\_dict\_lookup\_HashRow.Code = cleaned\_dict\_lookup.Code;

cleaned\_dict\_lookup\_HashRow.Description = cleaned\_dict\_lookup.Description;

cleaned\_dict\_lookup\_HashRow.Format\_Group = cleaned\_dict\_lookup.Format\_Group;

cleaned\_dict\_lookup\_HashRow.Format\_Subgroup = cleaned\_dict\_lookup.Format\_Subgroup;

cleaned\_dict\_lookup\_HashRow.Category\_Group = cleaned\_dict\_lookup.Category\_Group;

cleaned\_dict\_lookup\_HashRow.Category\_Subgroup = cleaned\_dict\_lookup.Category\_Subgroup;

tHash\_Lookup\_cleaned\_dict\_lookup

.put(cleaned\_dict\_lookup\_HashRow);

tos\_count\_tAdvancedHash\_cleaned\_dict\_lookup++;

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup main ]

\* stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup

\* process\_data\_begin ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_dict\_lookup";

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup

\* process\_data\_begin ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup

\* process\_data\_end ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_dict\_lookup";

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup

\* process\_data\_end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_2 process\_data\_end

\* ] start

\*/

currentComponent = "tFileOutputDelimited\_2";

/\*\*

\* [tFileOutputDelimited\_2 process\_data\_end

\* ] stop

\*/

} // End of branch "cleaned\_data\_dict"

/\*\*

\* [tMap\_3 process\_data\_end ] start

\*/

currentComponent = "tMap\_3";

/\*\*

\* [tMap\_3 process\_data\_end ] stop

\*/

} // End of branch "deduped\_raw\_data\_dict"

/\*\*

\* [tUniqRow\_1 process\_data\_end ] start

\*/

currentComponent = "tUniqRow\_1";

/\*\*

\* [tUniqRow\_1 process\_data\_end ] stop

\*/

} // End of branch "raw\_data\_dict"

/\*\*

\* [tFileInputDelimited\_2 process\_data\_end ] start

\*/

currentComponent = "tFileInputDelimited\_2";

/\*\*

\* [tFileInputDelimited\_2 process\_data\_end ] stop

\*/

/\*\*

\* [tFileInputDelimited\_2 end ] start

\*/

currentComponent = "tFileInputDelimited\_2";

}

} **finally** {

**if** (!((Object) ("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/Integrated\_Library\_System\_\_ILS\_\_Data\_Dictionary.csv") **instanceof** java.io.InputStream)) {

**if** (fid\_tFileInputDelimited\_2 != **null**) {

fid\_tFileInputDelimited\_2.close();

}

}

**if** (fid\_tFileInputDelimited\_2 != **null**) {

globalMap.put("tFileInputDelimited\_2\_NB\_LINE",

fid\_tFileInputDelimited\_2.getRowNumber());

}

}

ok\_Hash.put("tFileInputDelimited\_2", **true**);

end\_Hash.put("tFileInputDelimited\_2",

System.currentTimeMillis());

/\*\*

\* [tFileInputDelimited\_2 end ] stop

\*/

/\*\*

\* [tUniqRow\_1 end ] start

\*/

currentComponent = "tUniqRow\_1";

globalMap.put("tUniqRow\_1\_NB\_UNIQUES", nb\_uniques\_tUniqRow\_1);

globalMap.put("tUniqRow\_1\_NB\_DUPLICATES",

nb\_duplicates\_tUniqRow\_1);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("raw\_data\_dict"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tUniqRow\_1", **true**);

end\_Hash.put("tUniqRow\_1", System.currentTimeMillis());

/\*\*

\* [tUniqRow\_1 end ] stop

\*/

/\*\*

\* [tMap\_3 end ] start

\*/

currentComponent = "tMap\_3";

// ###############################

// # Lookup hashes releasing

// ###############################

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("deduped\_raw\_data\_dict"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tMap\_3", **true**);

end\_Hash.put("tMap\_3", System.currentTimeMillis());

/\*\*

\* [tMap\_3 end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_2 end ] start

\*/

currentComponent = "tFileOutputDelimited\_2";

**if** (outtFileOutputDelimited\_2 != **null**) {

outtFileOutputDelimited\_2.flush();

outtFileOutputDelimited\_2.close();

}

globalMap.put("tFileOutputDelimited\_2\_NB\_LINE",

nb\_line\_tFileOutputDelimited\_2);

globalMap.put("tFileOutputDelimited\_2\_FILE\_NAME",

fileName\_tFileOutputDelimited\_2);

resourceMap.put("finish\_tFileOutputDelimited\_2", **true**);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("cleaned\_data\_dict"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tFileOutputDelimited\_2", **true**);

end\_Hash.put("tFileOutputDelimited\_2",

System.currentTimeMillis());

/\*\*

\* [tFileOutputDelimited\_2 end ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup end ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_dict\_lookup";

tHash\_Lookup\_cleaned\_dict\_lookup.endPut();

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("cleaned\_dict\_lookup"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tAdvancedHash\_cleaned\_dict\_lookup", **true**);

end\_Hash.put("tAdvancedHash\_cleaned\_dict\_lookup",

System.currentTimeMillis());

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup end ] stop

\*/

}// end the resume

} **catch** (java.lang.Exception e) {

TalendException te = **new** TalendException(e, currentComponent,

globalMap);

**throw** te;

} **catch** (java.lang.Error error) {

runStat.stopThreadStat();

**throw** error;

} **finally** {

**try** {

/\*\*

\* [tFileInputDelimited\_2 finally ] start

\*/

currentComponent = "tFileInputDelimited\_2";

/\*\*

\* [tFileInputDelimited\_2 finally ] stop

\*/

/\*\*

\* [tUniqRow\_1 finally ] start

\*/

currentComponent = "tUniqRow\_1";

/\*\*

\* [tUniqRow\_1 finally ] stop

\*/

/\*\*

\* [tMap\_3 finally ] start

\*/

currentComponent = "tMap\_3";

/\*\*

\* [tMap\_3 finally ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_2 finally ] start

\*/

currentComponent = "tFileOutputDelimited\_2";

**if** (resourceMap.get("finish\_tFileOutputDelimited\_2") == **null**) {

java.io.Writer outtFileOutputDelimited\_2 = (java.io.Writer) resourceMap

.get("out\_tFileOutputDelimited\_2");

**if** (outtFileOutputDelimited\_2 != **null**) {

outtFileOutputDelimited\_2.flush();

outtFileOutputDelimited\_2.close();

}

}

/\*\*

\* [tFileOutputDelimited\_2 finally ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup finally ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_dict\_lookup";

/\*\*

\* [tAdvancedHash\_cleaned\_dict\_lookup finally ] stop

\*/

} **catch** (java.lang.Exception e) {

// ignore

} **catch** (java.lang.Error error) {

// ignore

}

resourceMap = **null**;

}

globalMap.put("tFileInputDelimited\_2\_SUBPROCESS\_STATE", 1);

}

**public** **static** **class** deduped\_unclean\_invStruct **implements**

routines.system.IPersistableRow<deduped\_unclean\_invStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String BibNum;

**public** String getBibNum() {

**return** **this**.BibNum;

}

**public** String Title;

**public** String getTitle() {

**return** **this**.Title;

}

**public** String Author;

**public** String getAuthor() {

**return** **this**.Author;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.BibNum = readString(dis);

**this**.Title = readString(dis);

**this**.Author = readString(dis);

**this**.Subjects = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.BibNum, dos);

// String

writeString(**this**.Title, dos);

// String

writeString(**this**.Author, dos);

// String

writeString(**this**.Subjects, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("BibNum=" + BibNum);

sb.append(",Title=" + Title);

sb.append(",Author=" + Author);

sb.append(",Subjects=" + Subjects);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(deduped\_unclean\_invStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** deduped\_invStruct **implements**

routines.system.IPersistableRow<deduped\_invStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String BibNum;

**public** String getBibNum() {

**return** **this**.BibNum;

}

**public** String Title;

**public** String getTitle() {

**return** **this**.Title;

}

**public** String Author;

**public** String getAuthor() {

**return** **this**.Author;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.BibNum = readString(dis);

**this**.Title = readString(dis);

**this**.Author = readString(dis);

**this**.Subjects = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.BibNum, dos);

// String

writeString(**this**.Title, dos);

// String

writeString(**this**.Author, dos);

// String

writeString(**this**.Subjects, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("BibNum=" + BibNum);

sb.append(",Title=" + Title);

sb.append(",Author=" + Author);

sb.append(",Subjects=" + Subjects);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(deduped\_invStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** decoded\_invStruct **implements**

routines.system.IPersistableRow<decoded\_invStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String BibNum;

**public** String getBibNum() {

**return** **this**.BibNum;

}

**public** String Title;

**public** String getTitle() {

**return** **this**.Title;

}

**public** String Author;

**public** String getAuthor() {

**return** **this**.Author;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.BibNum = readString(dis);

**this**.Title = readString(dis);

**this**.Author = readString(dis);

**this**.Subjects = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.BibNum, dos);

// String

writeString(**this**.Title, dos);

// String

writeString(**this**.Author, dos);

// String

writeString(**this**.Subjects, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("BibNum=" + BibNum);

sb.append(",Title=" + Title);

sb.append(",Author=" + Author);

sb.append(",Subjects=" + Subjects);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(decoded\_invStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **static** **class** raw\_invStruct **implements**

routines.system.IPersistableRow<raw\_invStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**public** String BibNum;

**public** String getBibNum() {

**return** **this**.BibNum;

}

**public** String Title;

**public** String getTitle() {

**return** **this**.Title;

}

**public** String Author;

**public** String getAuthor() {

**return** **this**.Author;

}

**public** String ISBN;

**public** String getISBN() {

**return** **this**.ISBN;

}

**public** String PublicationYear;

**public** String getPublicationYear() {

**return** **this**.PublicationYear;

}

**public** String Publisher;

**public** String getPublisher() {

**return** **this**.Publisher;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

**public** String ItemType;

**public** String getItemType() {

**return** **this**.ItemType;

}

**public** String ItemCollection;

**public** String getItemCollection() {

**return** **this**.ItemCollection;

}

**public** String FloatingItem;

**public** String getFloatingItem() {

**return** **this**.FloatingItem;

}

**public** String ItemLocation;

**public** String getItemLocation() {

**return** **this**.ItemLocation;

}

**public** String ReportDate;

**public** String getReportDate() {

**return** **this**.ReportDate;

}

**public** Integer ItemCount;

**public** Integer getItemCount() {

**return** **this**.ItemCount;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** Integer readInteger(ObjectInputStream dis) **throws** IOException {

Integer intReturn;

**int** length = 0;

length = dis.readByte();

**if** (length == -1) {

intReturn = **null**;

} **else** {

intReturn = dis.readInt();

}

**return** intReturn;

}

**private** **void** writeInteger(Integer intNum, ObjectOutputStream dos)

**throws** IOException {

**if** (intNum == **null**) {

dos.writeByte(-1);

} **else** {

dos.writeByte(0);

dos.writeInt(intNum);

}

}

**public** **void** readData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.BibNum = readString(dis);

**this**.Title = readString(dis);

**this**.Author = readString(dis);

**this**.ISBN = readString(dis);

**this**.PublicationYear = readString(dis);

**this**.Publisher = readString(dis);

**this**.Subjects = readString(dis);

**this**.ItemType = readString(dis);

**this**.ItemCollection = readString(dis);

**this**.FloatingItem = readString(dis);

**this**.ItemLocation = readString(dis);

**this**.ReportDate = readString(dis);

**this**.ItemCount = readInteger(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.BibNum, dos);

// String

writeString(**this**.Title, dos);

// String

writeString(**this**.Author, dos);

// String

writeString(**this**.ISBN, dos);

// String

writeString(**this**.PublicationYear, dos);

// String

writeString(**this**.Publisher, dos);

// String

writeString(**this**.Subjects, dos);

// String

writeString(**this**.ItemType, dos);

// String

writeString(**this**.ItemCollection, dos);

// String

writeString(**this**.FloatingItem, dos);

// String

writeString(**this**.ItemLocation, dos);

// String

writeString(**this**.ReportDate, dos);

// Integer

writeInteger(**this**.ItemCount, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("BibNum=" + BibNum);

sb.append(",Title=" + Title);

sb.append(",Author=" + Author);

sb.append(",ISBN=" + ISBN);

sb.append(",PublicationYear=" + PublicationYear);

sb.append(",Publisher=" + Publisher);

sb.append(",Subjects=" + Subjects);

sb.append(",ItemType=" + ItemType);

sb.append(",ItemCollection=" + ItemCollection);

sb.append(",FloatingItem=" + FloatingItem);

sb.append(",ItemLocation=" + ItemLocation);

sb.append(",ReportDate=" + ReportDate);

sb.append(",ItemCount=" + String.valueOf(ItemCount));

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(raw\_invStruct other) {

**int** returnValue = -1;

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **void** tFileInputDelimited\_3Process(

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

globalMap.put("tFileInputDelimited\_3\_SUBPROCESS\_STATE", 0);

**final** **boolean** execStat = **this**.execStat;

String iterateId = "";

String currentComponent = "";

java.util.Map<String, Object> resourceMap = **new** java.util.HashMap<String, Object>();

**try** {

// TDI-39566 avoid throwing an useless Exception

**boolean** resumeIt = **true**;

**if** (globalResumeTicket == **false** && resumeEntryMethodName != **null**) {

String currentMethodName = **new** java.lang.Exception()

.getStackTrace()[0].getMethodName();

resumeIt = resumeEntryMethodName.equals(currentMethodName);

}

**if** (resumeIt || globalResumeTicket) { // start the resume

globalResumeTicket = **true**;

raw\_invStruct raw\_inv = **new** raw\_invStruct();

decoded\_invStruct decoded\_inv = **new** decoded\_invStruct();

deduped\_invStruct deduped\_inv = **new** deduped\_invStruct();

deduped\_unclean\_invStruct deduped\_unclean\_inv = **new** deduped\_unclean\_invStruct();

/\*\*

\* [tFileOutputDelimited\_4 begin ] start

\*/

ok\_Hash.put("tFileOutputDelimited\_4", **false**);

start\_Hash.put("tFileOutputDelimited\_4",

System.currentTimeMillis());

currentComponent = "tFileOutputDelimited\_4";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("deduped\_unclean\_inv"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tFileOutputDelimited\_4 = 0;

**class** BytesLimit65535\_tFileOutputDelimited\_4 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileOutputDelimited\_4().limitLog4jByte();

String fileName\_tFileOutputDelimited\_4 = "";

fileName\_tFileOutputDelimited\_4 = (**new** java.io.File(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/Inv/out\_accept\_inv.csv"))

.getAbsolutePath().replace("\\", "/");

String fullName\_tFileOutputDelimited\_4 = **null**;

String extension\_tFileOutputDelimited\_4 = **null**;

String directory\_tFileOutputDelimited\_4 = **null**;

**if** ((fileName\_tFileOutputDelimited\_4.indexOf("/") != -1)) {

**if** (fileName\_tFileOutputDelimited\_4.lastIndexOf(".") < fileName\_tFileOutputDelimited\_4

.lastIndexOf("/")) {

fullName\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4;

extension\_tFileOutputDelimited\_4 = "";

} **else** {

fullName\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4

.substring(0, fileName\_tFileOutputDelimited\_4

.lastIndexOf("."));

extension\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4

.substring(fileName\_tFileOutputDelimited\_4

.lastIndexOf("."));

}

directory\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4

.substring(0, fileName\_tFileOutputDelimited\_4

.lastIndexOf("/"));

} **else** {

**if** (fileName\_tFileOutputDelimited\_4.lastIndexOf(".") != -1) {

fullName\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4

.substring(0, fileName\_tFileOutputDelimited\_4

.lastIndexOf("."));

extension\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4

.substring(fileName\_tFileOutputDelimited\_4

.lastIndexOf("."));

} **else** {

fullName\_tFileOutputDelimited\_4 = fileName\_tFileOutputDelimited\_4;

extension\_tFileOutputDelimited\_4 = "";

}

directory\_tFileOutputDelimited\_4 = "";

}

**boolean** isFileGenerated\_tFileOutputDelimited\_4 = **true**;

java.io.File filetFileOutputDelimited\_4 = **new** java.io.File(

fileName\_tFileOutputDelimited\_4);

globalMap.put("tFileOutputDelimited\_4\_FILE\_NAME",

fileName\_tFileOutputDelimited\_4);

**int** nb\_line\_tFileOutputDelimited\_4 = 0;

**int** splitedFileNo\_tFileOutputDelimited\_4 = 0;

**int** currentRow\_tFileOutputDelimited\_4 = 0;

**final** String OUT\_DELIM\_tFileOutputDelimited\_4 = /\*\*

\* Start field

\* tFileOutputDelimited\_4:FIELDSEPARATOR

\*/

"^"/\*\* End field tFileOutputDelimited\_4:FIELDSEPARATOR \*/

;

**final** String OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_4 = /\*\*

\* Start

\* field tFileOutputDelimited\_4:ROWSEPARATOR

\*/

"\n"/\*\* End field tFileOutputDelimited\_4:ROWSEPARATOR \*/

;

// create directory only if not exists

**if** (directory\_tFileOutputDelimited\_4 != **null**

&& directory\_tFileOutputDelimited\_4.trim().length() != 0) {

java.io.File dir\_tFileOutputDelimited\_4 = **new** java.io.File(

directory\_tFileOutputDelimited\_4);

**if** (!dir\_tFileOutputDelimited\_4.exists()) {

dir\_tFileOutputDelimited\_4.mkdirs();

}

}

// routines.system.Row

java.io.Writer outtFileOutputDelimited\_4 = **null**;

java.io.File fileToDelete\_tFileOutputDelimited\_4 = **new** java.io.File(

fileName\_tFileOutputDelimited\_4);

**if** (fileToDelete\_tFileOutputDelimited\_4.exists()) {

fileToDelete\_tFileOutputDelimited\_4.delete();

}

outtFileOutputDelimited\_4 = **new** java.io.BufferedWriter(

**new** java.io.OutputStreamWriter(

**new** java.io.FileOutputStream(

fileName\_tFileOutputDelimited\_4, **false**),

"ISO-8859-15"));

**if** (filetFileOutputDelimited\_4.length() == 0) {

outtFileOutputDelimited\_4.write("BibNum");

outtFileOutputDelimited\_4

.write(OUT\_DELIM\_tFileOutputDelimited\_4);

outtFileOutputDelimited\_4.write("Title");

outtFileOutputDelimited\_4

.write(OUT\_DELIM\_tFileOutputDelimited\_4);

outtFileOutputDelimited\_4.write("Author");

outtFileOutputDelimited\_4

.write(OUT\_DELIM\_tFileOutputDelimited\_4);

outtFileOutputDelimited\_4.write("Subjects");

outtFileOutputDelimited\_4

.write(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_4);

outtFileOutputDelimited\_4.flush();

}

resourceMap.put("out\_tFileOutputDelimited\_4",

outtFileOutputDelimited\_4);

resourceMap.put("nb\_line\_tFileOutputDelimited\_4",

nb\_line\_tFileOutputDelimited\_4);

/\*\*

\* [tFileOutputDelimited\_4 begin ] stop

\*/

/\*\*

\* [tMap\_4 begin ] start

\*/

ok\_Hash.put("tMap\_4", **false**);

start\_Hash.put("tMap\_4", System.currentTimeMillis());

currentComponent = "tMap\_4";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("deduped\_inv"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tMap\_4 = 0;

**class** BytesLimit65535\_tMap\_4 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tMap\_4().limitLog4jByte();

// ###############################

// # Lookup's keys initialization

// ###############################

// ###############################

// # Vars initialization

**class** Var\_\_tMap\_4\_\_Struct {

}

Var\_\_tMap\_4\_\_Struct Var\_\_tMap\_4 = **new** Var\_\_tMap\_4\_\_Struct();

// ###############################

// ###############################

// # Outputs initialization

deduped\_unclean\_invStruct deduped\_unclean\_inv\_tmp = **new** deduped\_unclean\_invStruct();

// ###############################

/\*\*

\* [tMap\_4 begin ] stop

\*/

/\*\*

\* [tUniqRow\_2 begin ] start

\*/

ok\_Hash.put("tUniqRow\_2", **false**);

start\_Hash.put("tUniqRow\_2", System.currentTimeMillis());

currentComponent = "tUniqRow\_2";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("decoded\_inv"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tUniqRow\_2 = 0;

**class** BytesLimit65535\_tUniqRow\_2 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tUniqRow\_2().limitLog4jByte();

**class** KeyStruct\_tUniqRow\_2 {

**private** **static** **final** **int** DEFAULT\_HASHCODE = 1;

**private** **static** **final** **int** PRIME = 31;

**private** **int** hashCode = DEFAULT\_HASHCODE;

**public** **boolean** hashCodeDirty = **true**;

String BibNum;

String Title;

String Author;

@Override

**public** **int** hashCode() {

**if** (**this**.hashCodeDirty) {

**final** **int** prime = PRIME;

**int** result = DEFAULT\_HASHCODE;

result = prime

\* result

+ ((**this**.BibNum == **null**) ? 0 : **this**.BibNum

.hashCode());

result = prime

\* result

+ ((**this**.Title == **null**) ? 0 : **this**.Title

.hashCode());

result = prime

\* result

+ ((**this**.Author == **null**) ? 0 : **this**.Author

.hashCode());

**this**.hashCode = result;

**this**.hashCodeDirty = **false**;

}

**return** **this**.hashCode;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

**final** KeyStruct\_tUniqRow\_2 other = (KeyStruct\_tUniqRow\_2) obj;

**if** (**this**.BibNum == **null**) {

**if** (other.BibNum != **null**)

**return** **false**;

} **else** **if** (!**this**.BibNum.equals(other.BibNum))

**return** **false**;

**if** (**this**.Title == **null**) {

**if** (other.Title != **null**)

**return** **false**;

} **else** **if** (!**this**.Title.equals(other.Title))

**return** **false**;

**if** (**this**.Author == **null**) {

**if** (other.Author != **null**)

**return** **false**;

} **else** **if** (!**this**.Author.equals(other.Author))

**return** **false**;

**return** **true**;

}

}

**int** nb\_uniques\_tUniqRow\_2 = 0;

**int** nb\_duplicates\_tUniqRow\_2 = 0;

KeyStruct\_tUniqRow\_2 finder\_tUniqRow\_2 = **new** KeyStruct\_tUniqRow\_2();

java.util.Set<KeyStruct\_tUniqRow\_2> keystUniqRow\_2 = **new** java.util.HashSet<KeyStruct\_tUniqRow\_2>();

/\*\*

\* [tUniqRow\_2 begin ] stop

\*/

/\*\*

\* [tMap\_2 begin ] start

\*/

ok\_Hash.put("tMap\_2", **false**);

start\_Hash.put("tMap\_2", System.currentTimeMillis());

currentComponent = "tMap\_2";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("raw\_inv" + iterateId,

0, 0);

}

}

**int** tos\_count\_tMap\_2 = 0;

**class** BytesLimit65535\_tMap\_2 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tMap\_2().limitLog4jByte();

// ###############################

// # Lookup's keys initialization

// ###############################

// ###############################

// # Vars initialization

**class** Var\_\_tMap\_2\_\_Struct {

}

Var\_\_tMap\_2\_\_Struct Var\_\_tMap\_2 = **new** Var\_\_tMap\_2\_\_Struct();

// ###############################

// ###############################

// # Outputs initialization

decoded\_invStruct decoded\_inv\_tmp = **new** decoded\_invStruct();

// ###############################

/\*\*

\* [tMap\_2 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_3 begin ] start

\*/

ok\_Hash.put("tFileInputDelimited\_3", **false**);

start\_Hash.put("tFileInputDelimited\_3",

System.currentTimeMillis());

currentComponent = "tFileInputDelimited\_3";

**int** tos\_count\_tFileInputDelimited\_3 = 0;

**class** BytesLimit65535\_tFileInputDelimited\_3 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileInputDelimited\_3().limitLog4jByte();

**final** routines.system.RowState rowstate\_tFileInputDelimited\_3 = **new** routines.system.RowState();

**int** nb\_line\_tFileInputDelimited\_3 = 0;

org.talend.fileprocess.FileInputDelimited fid\_tFileInputDelimited\_3 = **null**;

**try** {

Object filename\_tFileInputDelimited\_3 = "E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/export\_dataframe.csv";

**if** (filename\_tFileInputDelimited\_3 **instanceof** java.io.InputStream) {

**int** footer\_value\_tFileInputDelimited\_3 = 0, random\_value\_tFileInputDelimited\_3 = -1;

**if** (footer\_value\_tFileInputDelimited\_3 > 0

|| random\_value\_tFileInputDelimited\_3 > 0) {

**throw** **new** java.lang.Exception(

"When the input source is a stream,footer and random shouldn't be bigger than 0.");

}

}

**try** {

fid\_tFileInputDelimited\_3 = **new** org.talend.fileprocess.FileInputDelimited(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/export\_dataframe.csv",

"US-ASCII", "^", "\n", **false**, 1, 0, -1, -1,

**false**);

} **catch** (java.lang.Exception e) {

System.err.println(e.getMessage());

}

**while** (fid\_tFileInputDelimited\_3 != **null**

&& fid\_tFileInputDelimited\_3.nextRecord()) {

rowstate\_tFileInputDelimited\_3.reset();

raw\_inv = **null**;

**boolean** whetherReject\_tFileInputDelimited\_3 = **false**;

raw\_inv = **new** raw\_invStruct();

**try** {

**int** columnIndexWithD\_tFileInputDelimited\_3 = 0;

String temp = "";

columnIndexWithD\_tFileInputDelimited\_3 = 0;

raw\_inv.BibNum = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 1;

raw\_inv.Title = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 2;

raw\_inv.Author = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 3;

raw\_inv.ISBN = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 4;

raw\_inv.PublicationYear = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 5;

raw\_inv.Publisher = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 6;

raw\_inv.Subjects = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 7;

raw\_inv.ItemType = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 8;

raw\_inv.ItemCollection = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 9;

raw\_inv.FloatingItem = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 10;

raw\_inv.ItemLocation = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 11;

raw\_inv.ReportDate = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

columnIndexWithD\_tFileInputDelimited\_3 = 12;

temp = fid\_tFileInputDelimited\_3

.get(columnIndexWithD\_tFileInputDelimited\_3);

**if** (temp.length() > 0) {

**try** {

raw\_inv.ItemCount = ParserUtils

.parseTo\_Integer(temp);

} **catch** (java.lang.Exception ex\_tFileInputDelimited\_3) {

rowstate\_tFileInputDelimited\_3

.setException(**new** RuntimeException(

String.format(

"Couldn't parse value for column '%s' in '%s', value is '%s'. Details: %s",

"ItemCount",

"raw\_inv", temp,

ex\_tFileInputDelimited\_3),

ex\_tFileInputDelimited\_3));

}

} **else** {

raw\_inv.ItemCount = **null**;

}

**if** (rowstate\_tFileInputDelimited\_3.getException() != **null**) {

**throw** rowstate\_tFileInputDelimited\_3

.getException();

}

} **catch** (java.lang.Exception e) {

whetherReject\_tFileInputDelimited\_3 = **true**;

System.err.println(e.getMessage());

raw\_inv = **null**;

}

/\*\*

\* [tFileInputDelimited\_3 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_3 main ] start

\*/

currentComponent = "tFileInputDelimited\_3";

tos\_count\_tFileInputDelimited\_3++;

/\*\*

\* [tFileInputDelimited\_3 main ] stop

\*/

/\*\*

\* [tFileInputDelimited\_3 process\_data\_begin ] start

\*/

currentComponent = "tFileInputDelimited\_3";

/\*\*

\* [tFileInputDelimited\_3 process\_data\_begin ] stop

\*/

// Start of branch "raw\_inv"

**if** (raw\_inv != **null**) {

/\*\*

\* [tMap\_2 main ] start

\*/

currentComponent = "tMap\_2";

// raw\_inv

// raw\_inv

**if** (execStat) {

runStat.updateStatOnConnection("raw\_inv"

+ iterateId, 1, 1);

}

**boolean** hasCasePrimitiveKeyWithNull\_tMap\_2 = **false**;

// ###############################

// # Input tables (lookups)

**boolean** rejectedInnerJoin\_tMap\_2 = **false**;

**boolean** mainRowRejected\_tMap\_2 = **false**;

// ###############################

{ // start of Var scope

// ###############################

// # Vars tables

Var\_\_tMap\_2\_\_Struct Var = Var\_\_tMap\_2;// ###############################

// ###############################

// # Output tables

decoded\_inv = **null**;

// # Output table : 'decoded\_inv'

decoded\_inv\_tmp.BibNum = raw\_inv.BibNum;

decoded\_inv\_tmp.Title = DecodeString

.decoder(raw\_inv.Title);

decoded\_inv\_tmp.Author = DecodeString

.decoder(raw\_inv.Author);

decoded\_inv\_tmp.Subjects = DecodeString

.decoder(raw\_inv.Subjects);

decoded\_inv = decoded\_inv\_tmp;

// ###############################

} // end of Var scope

rejectedInnerJoin\_tMap\_2 = **false**;

tos\_count\_tMap\_2++;

/\*\*

\* [tMap\_2 main ] stop

\*/

/\*\*

\* [tMap\_2 process\_data\_begin ] start

\*/

currentComponent = "tMap\_2";

/\*\*

\* [tMap\_2 process\_data\_begin ] stop

\*/

// Start of branch "decoded\_inv"

**if** (decoded\_inv != **null**) {

/\*\*

\* [tUniqRow\_2 main ] start

\*/

currentComponent = "tUniqRow\_2";

// decoded\_inv

// decoded\_inv

**if** (execStat) {

runStat.updateStatOnConnection(

"decoded\_inv" + iterateId, 1, 1);

}

deduped\_inv = **null**;

**if** (decoded\_inv.BibNum == **null**) {

finder\_tUniqRow\_2.BibNum = **null**;

} **else** {

finder\_tUniqRow\_2.BibNum = decoded\_inv.BibNum

.toLowerCase();

}

**if** (decoded\_inv.Title == **null**) {

finder\_tUniqRow\_2.Title = **null**;

} **else** {

finder\_tUniqRow\_2.Title = decoded\_inv.Title

.toLowerCase();

}

**if** (decoded\_inv.Author == **null**) {

finder\_tUniqRow\_2.Author = **null**;

} **else** {

finder\_tUniqRow\_2.Author = decoded\_inv.Author

.toLowerCase();

}

finder\_tUniqRow\_2.hashCodeDirty = **true**;

**if** (!keystUniqRow\_2.contains(finder\_tUniqRow\_2)) {

KeyStruct\_tUniqRow\_2 new\_tUniqRow\_2 = **new** KeyStruct\_tUniqRow\_2();

**if** (decoded\_inv.BibNum == **null**) {

new\_tUniqRow\_2.BibNum = **null**;

} **else** {

new\_tUniqRow\_2.BibNum = decoded\_inv.BibNum

.toLowerCase();

}

**if** (decoded\_inv.Title == **null**) {

new\_tUniqRow\_2.Title = **null**;

} **else** {

new\_tUniqRow\_2.Title = decoded\_inv.Title

.toLowerCase();

}

**if** (decoded\_inv.Author == **null**) {

new\_tUniqRow\_2.Author = **null**;

} **else** {

new\_tUniqRow\_2.Author = decoded\_inv.Author

.toLowerCase();

}

keystUniqRow\_2.add(new\_tUniqRow\_2);

**if** (deduped\_inv == **null**) {

deduped\_inv = **new** deduped\_invStruct();

}

deduped\_inv.BibNum = decoded\_inv.BibNum;

deduped\_inv.Title = decoded\_inv.Title;

deduped\_inv.Author = decoded\_inv.Author;

deduped\_inv.Subjects = decoded\_inv.Subjects;

nb\_uniques\_tUniqRow\_2++;

} **else** {

nb\_duplicates\_tUniqRow\_2++;

}

tos\_count\_tUniqRow\_2++;

/\*\*

\* [tUniqRow\_2 main ] stop

\*/

/\*\*

\* [tUniqRow\_2 process\_data\_begin ] start

\*/

currentComponent = "tUniqRow\_2";

/\*\*

\* [tUniqRow\_2 process\_data\_begin ] stop

\*/

// Start of branch "deduped\_inv"

**if** (deduped\_inv != **null**) {

/\*\*

\* [tMap\_4 main ] start

\*/

currentComponent = "tMap\_4";

// deduped\_inv

// deduped\_inv

**if** (execStat) {

runStat.updateStatOnConnection(

"deduped\_inv" + iterateId, 1, 1);

}

**boolean** hasCasePrimitiveKeyWithNull\_tMap\_4 = **false**;

// ###############################

// # Input tables (lookups)

**boolean** rejectedInnerJoin\_tMap\_4 = **false**;

**boolean** mainRowRejected\_tMap\_4 = **false**;

// ###############################

{ // start of Var scope

// ###############################

// # Vars tables

Var\_\_tMap\_4\_\_Struct Var = Var\_\_tMap\_4;// ###############################

// ###############################

// # Output tables

deduped\_unclean\_inv = **null**;

// # Output table :

// 'deduped\_unclean\_inv'

// # Filter conditions

**if** (

!Relational.ISNULL(deduped\_inv.Title) ? (deduped\_inv.Title

.trim() != "" ? **true** : **false**)

: **false**

) {

deduped\_unclean\_inv\_tmp.BibNum = deduped\_inv.BibNum;

deduped\_unclean\_inv\_tmp.Title = deduped\_inv.Title

.contains("/") ? deduped\_inv.Title

.substring(

0,

StringHandling

.INDEX(deduped\_inv.Title,

"/"))

.trim()

: deduped\_inv.Title.trim();

deduped\_unclean\_inv\_tmp.Author = !Relational

.ISNULL(deduped\_inv.Author) ? (deduped\_inv.Author

.trim() == "" ? "No Author Data"

: StringHandling

.EREPLACE(

deduped\_inv.Author,

"([,][^A-Za-z][^A-Za-z ]+)",

""))

: "No Author Data";

deduped\_unclean\_inv\_tmp.Subjects = deduped\_inv.Subjects;

deduped\_unclean\_inv = deduped\_unclean\_inv\_tmp;

} // closing filter/reject

// ###############################

} // end of Var scope

rejectedInnerJoin\_tMap\_4 = **false**;

tos\_count\_tMap\_4++;

/\*\*

\* [tMap\_4 main ] stop

\*/

/\*\*

\* [tMap\_4 process\_data\_begin ] start

\*/

currentComponent = "tMap\_4";

/\*\*

\* [tMap\_4 process\_data\_begin ] stop

\*/

// Start of branch "deduped\_unclean\_inv"

**if** (deduped\_unclean\_inv != **null**) {

/\*\*

\* [tFileOutputDelimited\_4 main ] start

\*/

currentComponent = "tFileOutputDelimited\_4";

// deduped\_unclean\_inv

// deduped\_unclean\_inv

**if** (execStat) {

runStat.updateStatOnConnection(

"deduped\_unclean\_inv"

+ iterateId, 1, 1);

}

StringBuilder sb\_tFileOutputDelimited\_4 = **new** StringBuilder();

**if** (deduped\_unclean\_inv.BibNum != **null**) {

sb\_tFileOutputDelimited\_4

.append(deduped\_unclean\_inv.BibNum);

}

sb\_tFileOutputDelimited\_4

.append(OUT\_DELIM\_tFileOutputDelimited\_4);

**if** (deduped\_unclean\_inv.Title != **null**) {

sb\_tFileOutputDelimited\_4

.append(deduped\_unclean\_inv.Title);

}

sb\_tFileOutputDelimited\_4

.append(OUT\_DELIM\_tFileOutputDelimited\_4);

**if** (deduped\_unclean\_inv.Author != **null**) {

sb\_tFileOutputDelimited\_4

.append(deduped\_unclean\_inv.Author);

}

sb\_tFileOutputDelimited\_4

.append(OUT\_DELIM\_tFileOutputDelimited\_4);

**if** (deduped\_unclean\_inv.Subjects != **null**) {

sb\_tFileOutputDelimited\_4

.append(deduped\_unclean\_inv.Subjects);

}

sb\_tFileOutputDelimited\_4

.append(OUT\_DELIM\_ROWSEP\_tFileOutputDelimited\_4);

nb\_line\_tFileOutputDelimited\_4++;

resourceMap

.put("nb\_line\_tFileOutputDelimited\_4",

nb\_line\_tFileOutputDelimited\_4);

outtFileOutputDelimited\_4

.write(sb\_tFileOutputDelimited\_4

.toString());

tos\_count\_tFileOutputDelimited\_4++;

/\*\*

\* [tFileOutputDelimited\_4 main ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_4

\* process\_data\_begin ] start

\*/

currentComponent = "tFileOutputDelimited\_4";

/\*\*

\* [tFileOutputDelimited\_4

\* process\_data\_begin ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_4

\* process\_data\_end ] start

\*/

currentComponent = "tFileOutputDelimited\_4";

/\*\*

\* [tFileOutputDelimited\_4

\* process\_data\_end ] stop

\*/

} // End of branch "deduped\_unclean\_inv"

/\*\*

\* [tMap\_4 process\_data\_end ] start

\*/

currentComponent = "tMap\_4";

/\*\*

\* [tMap\_4 process\_data\_end ] stop

\*/

} // End of branch "deduped\_inv"

/\*\*

\* [tUniqRow\_2 process\_data\_end ] start

\*/

currentComponent = "tUniqRow\_2";

/\*\*

\* [tUniqRow\_2 process\_data\_end ] stop

\*/

} // End of branch "decoded\_inv"

/\*\*

\* [tMap\_2 process\_data\_end ] start

\*/

currentComponent = "tMap\_2";

/\*\*

\* [tMap\_2 process\_data\_end ] stop

\*/

} // End of branch "raw\_inv"

/\*\*

\* [tFileInputDelimited\_3 process\_data\_end ] start

\*/

currentComponent = "tFileInputDelimited\_3";

/\*\*

\* [tFileInputDelimited\_3 process\_data\_end ] stop

\*/

/\*\*

\* [tFileInputDelimited\_3 end ] start

\*/

currentComponent = "tFileInputDelimited\_3";

}

} **finally** {

**if** (!((Object) ("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/export\_dataframe.csv") **instanceof** java.io.InputStream)) {

**if** (fid\_tFileInputDelimited\_3 != **null**) {

fid\_tFileInputDelimited\_3.close();

}

}

**if** (fid\_tFileInputDelimited\_3 != **null**) {

globalMap.put("tFileInputDelimited\_3\_NB\_LINE",

fid\_tFileInputDelimited\_3.getRowNumber());

}

}

ok\_Hash.put("tFileInputDelimited\_3", **true**);

end\_Hash.put("tFileInputDelimited\_3",

System.currentTimeMillis());

/\*\*

\* [tFileInputDelimited\_3 end ] stop

\*/

/\*\*

\* [tMap\_2 end ] start

\*/

currentComponent = "tMap\_2";

// ###############################

// # Lookup hashes releasing

// ###############################

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("raw\_inv" + iterateId,

2, 0);

}

}

ok\_Hash.put("tMap\_2", **true**);

end\_Hash.put("tMap\_2", System.currentTimeMillis());

/\*\*

\* [tMap\_2 end ] stop

\*/

/\*\*

\* [tUniqRow\_2 end ] start

\*/

currentComponent = "tUniqRow\_2";

globalMap.put("tUniqRow\_2\_NB\_UNIQUES", nb\_uniques\_tUniqRow\_2);

globalMap.put("tUniqRow\_2\_NB\_DUPLICATES",

nb\_duplicates\_tUniqRow\_2);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("decoded\_inv"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tUniqRow\_2", **true**);

end\_Hash.put("tUniqRow\_2", System.currentTimeMillis());

/\*\*

\* [tUniqRow\_2 end ] stop

\*/

/\*\*

\* [tMap\_4 end ] start

\*/

currentComponent = "tMap\_4";

// ###############################

// # Lookup hashes releasing

// ###############################

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("deduped\_inv"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tMap\_4", **true**);

end\_Hash.put("tMap\_4", System.currentTimeMillis());

/\*\*

\* [tMap\_4 end ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_4 end ] start

\*/

currentComponent = "tFileOutputDelimited\_4";

**if** (outtFileOutputDelimited\_4 != **null**) {

outtFileOutputDelimited\_4.flush();

outtFileOutputDelimited\_4.close();

}

globalMap.put("tFileOutputDelimited\_4\_NB\_LINE",

nb\_line\_tFileOutputDelimited\_4);

globalMap.put("tFileOutputDelimited\_4\_FILE\_NAME",

fileName\_tFileOutputDelimited\_4);

resourceMap.put("finish\_tFileOutputDelimited\_4", **true**);

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("deduped\_unclean\_inv"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tFileOutputDelimited\_4", **true**);

end\_Hash.put("tFileOutputDelimited\_4",

System.currentTimeMillis());

/\*\*

\* [tFileOutputDelimited\_4 end ] stop

\*/

}// end the resume

**if** (resumeEntryMethodName == **null** || globalResumeTicket) {

resumeUtil

.addLog("CHECKPOINT",

"CONNECTION:SUBJOB\_OK:tFileInputDelimited\_3:OnSubjobOk",

"", Thread.currentThread().getId() + "", "",

"", "", "", "");

}

**if** (execStat) {

runStat.updateStatOnConnection("OnSubjobOk1", 0, "ok");

}

tFileInputDelimited\_2Process(globalMap);

} **catch** (java.lang.Exception e) {

TalendException te = **new** TalendException(e, currentComponent,

globalMap);

**throw** te;

} **catch** (java.lang.Error error) {

runStat.stopThreadStat();

**throw** error;

} **finally** {

**try** {

/\*\*

\* [tFileInputDelimited\_3 finally ] start

\*/

currentComponent = "tFileInputDelimited\_3";

/\*\*

\* [tFileInputDelimited\_3 finally ] stop

\*/

/\*\*

\* [tMap\_2 finally ] start

\*/

currentComponent = "tMap\_2";

/\*\*

\* [tMap\_2 finally ] stop

\*/

/\*\*

\* [tUniqRow\_2 finally ] start

\*/

currentComponent = "tUniqRow\_2";

/\*\*

\* [tUniqRow\_2 finally ] stop

\*/

/\*\*

\* [tMap\_4 finally ] start

\*/

currentComponent = "tMap\_4";

/\*\*

\* [tMap\_4 finally ] stop

\*/

/\*\*

\* [tFileOutputDelimited\_4 finally ] start

\*/

currentComponent = "tFileOutputDelimited\_4";

**if** (resourceMap.get("finish\_tFileOutputDelimited\_4") == **null**) {

java.io.Writer outtFileOutputDelimited\_4 = (java.io.Writer) resourceMap

.get("out\_tFileOutputDelimited\_4");

**if** (outtFileOutputDelimited\_4 != **null**) {

outtFileOutputDelimited\_4.flush();

outtFileOutputDelimited\_4.close();

}

}

/\*\*

\* [tFileOutputDelimited\_4 finally ] stop

\*/

} **catch** (java.lang.Exception e) {

// ignore

} **catch** (java.lang.Error error) {

// ignore

}

resourceMap = **null**;

}

globalMap.put("tFileInputDelimited\_3\_SUBPROCESS\_STATE", 1);

}

**public** **static** **class** cleaned\_sdl\_invStruct

**implements**

routines.system.IPersistableComparableLookupRow<cleaned\_sdl\_invStruct> {

**final** **static** **byte**[] commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**static** **byte**[] commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[0];

**protected** **static** **final** **int** DEFAULT\_HASHCODE = 1;

**protected** **static** **final** **int** PRIME = 31;

**protected** **int** hashCode = DEFAULT\_HASHCODE;

**public** **boolean** hashCodeDirty = **true**;

**public** String loopKey;

**public** String BibNum;

**public** String getBibNum() {

**return** **this**.BibNum;

}

**public** String Title;

**public** String getTitle() {

**return** **this**.Title;

}

**public** String Author;

**public** String getAuthor() {

**return** **this**.Author;

}

**public** String Subjects;

**public** String getSubjects() {

**return** **this**.Subjects;

}

@Override

**public** **int** hashCode() {

**if** (**this**.hashCodeDirty) {

**final** **int** prime = PRIME;

**int** result = DEFAULT\_HASHCODE;

result = prime \* result

+ ((**this**.BibNum == **null**) ? 0 : **this**.BibNum.hashCode());

**this**.hashCode = result;

**this**.hashCodeDirty = **false**;

}

**return** **this**.hashCode;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

**final** cleaned\_sdl\_invStruct other = (cleaned\_sdl\_invStruct) obj;

**if** (**this**.BibNum == **null**) {

**if** (other.BibNum != **null**)

**return** **false**;

} **else** **if** (!**this**.BibNum.equals(other.BibNum))

**return** **false**;

**return** **true**;

}

**public** **void** copyDataTo(cleaned\_sdl\_invStruct other) {

other.BibNum = **this**.BibNum;

other.Title = **this**.Title;

other.Author = **this**.Author;

other.Subjects = **this**.Subjects;

}

**public** **void** copyKeysDataTo(cleaned\_sdl\_invStruct other) {

other.BibNum = **this**.BibNum;

}

**private** String readString(ObjectInputStream dis) **throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**if** (length > commonByteArray\_ADM\_SDL\_cleanup\_transformation.length) {

**if** (length < 1024

&& commonByteArray\_ADM\_SDL\_cleanup\_transformation.length == 0) {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[1024];

} **else** {

commonByteArray\_ADM\_SDL\_cleanup\_transformation = **new** **byte**[2 \* length];

}

}

dis.readFully(commonByteArray\_ADM\_SDL\_cleanup\_transformation,

0, length);

strReturn = **new** String(

commonByteArray\_ADM\_SDL\_cleanup\_transformation, 0,

length, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, ObjectOutputStream dos)

**throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**private** String readString(DataInputStream dis, ObjectInputStream ois)

**throws** IOException {

String strReturn = **null**;

**int** length = 0;

length = dis.readInt();

**if** (length == -1) {

strReturn = **null**;

} **else** {

**byte**[] byteArray = **new** **byte**[length];

dis.read(byteArray);

strReturn = **new** String(byteArray, utf8Charset);

}

**return** strReturn;

}

**private** **void** writeString(String str, DataOutputStream dos,

ObjectOutputStream oos) **throws** IOException {

**if** (str == **null**) {

dos.writeInt(-1);

} **else** {

**byte**[] byteArray = str.getBytes(utf8Charset);

dos.writeInt(byteArray.length);

dos.write(byteArray);

}

}

**public** **void** readKeysData(ObjectInputStream dis) {

**synchronized** (commonByteArrayLock\_ADM\_SDL\_cleanup\_transformation) {

**try** {

**int** length = 0;

**this**.BibNum = readString(dis);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

}

**public** **void** writeKeysData(ObjectOutputStream dos) {

**try** {

// String

writeString(**this**.BibNum, dos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

/\*\*

\* Fill Values data by reading ObjectInputStream.

\*/

**public** **void** readValuesData(DataInputStream dis, ObjectInputStream ois) {

**try** {

**int** length = 0;

**this**.Title = readString(dis, ois);

**this**.Author = readString(dis, ois);

**this**.Subjects = readString(dis, ois);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

/\*\*

\* Return a byte array which represents Values data.

\*/

**public** **void** writeValuesData(DataOutputStream dos, ObjectOutputStream oos) {

**try** {

writeString(**this**.Title, dos, oos);

writeString(**this**.Author, dos, oos);

writeString(**this**.Subjects, dos, oos);

} **catch** (IOException e) {

**throw** **new** RuntimeException(e);

}

}

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append(**super**.toString());

sb.append("[");

sb.append("BibNum=" + BibNum);

sb.append(",Title=" + Title);

sb.append(",Author=" + Author);

sb.append(",Subjects=" + Subjects);

sb.append("]");

**return** sb.toString();

}

/\*\*

\* Compare keys

\*/

**public** **int** compareTo(cleaned\_sdl\_invStruct other) {

**int** returnValue = -1;

returnValue = checkNullsAndCompare(**this**.BibNum, other.BibNum);

**if** (returnValue != 0) {

**return** returnValue;

}

**return** returnValue;

}

**private** **int** checkNullsAndCompare(Object object1, Object object2) {

**int** returnValue = 0;

**if** (object1 **instanceof** Comparable && object2 **instanceof** Comparable) {

returnValue = ((Comparable) object1).compareTo(object2);

} **else** **if** (object1 != **null** && object2 != **null**) {

returnValue = compareStrings(object1.toString(),

object2.toString());

} **else** **if** (object1 == **null** && object2 != **null**) {

returnValue = 1;

} **else** **if** (object1 != **null** && object2 == **null**) {

returnValue = -1;

} **else** {

returnValue = 0;

}

**return** returnValue;

}

**private** **int** compareStrings(String string1, String string2) {

**return** string1.compareTo(string2);

}

}

**public** **void** tFileInputDelimited\_5Process(

**final** java.util.Map<String, Object> globalMap)

**throws** TalendException {

globalMap.put("tFileInputDelimited\_5\_SUBPROCESS\_STATE", 0);

**final** **boolean** execStat = **this**.execStat;

String iterateId = "";

String currentComponent = "";

java.util.Map<String, Object> resourceMap = **new** java.util.HashMap<String, Object>();

**try** {

// TDI-39566 avoid throwing an useless Exception

**boolean** resumeIt = **true**;

**if** (globalResumeTicket == **false** && resumeEntryMethodName != **null**) {

String currentMethodName = **new** java.lang.Exception()

.getStackTrace()[0].getMethodName();

resumeIt = resumeEntryMethodName.equals(currentMethodName);

}

**if** (resumeIt || globalResumeTicket) { // start the resume

globalResumeTicket = **true**;

cleaned\_sdl\_invStruct cleaned\_sdl\_inv = **new** cleaned\_sdl\_invStruct();

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv begin ] start

\*/

ok\_Hash.put("tAdvancedHash\_cleaned\_sdl\_inv", **false**);

start\_Hash.put("tAdvancedHash\_cleaned\_sdl\_inv",

System.currentTimeMillis());

currentComponent = "tAdvancedHash\_cleaned\_sdl\_inv";

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**) {

runStat.updateStatOnConnection("cleaned\_sdl\_inv"

+ iterateId, 0, 0);

}

}

**int** tos\_count\_tAdvancedHash\_cleaned\_sdl\_inv = 0;

**class** BytesLimit65535\_tAdvancedHash\_cleaned\_sdl\_inv {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tAdvancedHash\_cleaned\_sdl\_inv()

.limitLog4jByte();

// connection name:cleaned\_sdl\_inv

// source node:tFileInputDelimited\_5 -

// inputs:(after\_tFileInputDelimited\_4)

// outputs:(cleaned\_sdl\_inv,cleaned\_sdl\_inv) | target

// node:tAdvancedHash\_cleaned\_sdl\_inv - inputs:(cleaned\_sdl\_inv)

// outputs:()

// linked node: tMap\_5 - inputs:(norm\_checkouts,cleaned\_sdl\_inv)

// outputs:(validcheckouts,invalidcheckouts)

org.talend.designer.components.lookup.common.ICommonLookup.MATCHING\_MODE matchingModeEnum\_cleaned\_sdl\_inv = org.talend.designer.components.lookup.common.ICommonLookup.MATCHING\_MODE.UNIQUE\_MATCH;

org.talend.designer.components.lookup.memory.AdvancedMemoryLookup<cleaned\_sdl\_invStruct> tHash\_Lookup\_cleaned\_sdl\_inv = org.talend.designer.components.lookup.memory.AdvancedMemoryLookup

.<cleaned\_sdl\_invStruct> getLookup(matchingModeEnum\_cleaned\_sdl\_inv);

globalMap.put("tHash\_Lookup\_cleaned\_sdl\_inv",

tHash\_Lookup\_cleaned\_sdl\_inv);

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_5 begin ] start

\*/

ok\_Hash.put("tFileInputDelimited\_5", **false**);

start\_Hash.put("tFileInputDelimited\_5",

System.currentTimeMillis());

currentComponent = "tFileInputDelimited\_5";

**int** tos\_count\_tFileInputDelimited\_5 = 0;

**class** BytesLimit65535\_tFileInputDelimited\_5 {

**public** **void** limitLog4jByte() **throws** Exception {

}

}

**new** BytesLimit65535\_tFileInputDelimited\_5().limitLog4jByte();

**final** routines.system.RowState rowstate\_tFileInputDelimited\_5 = **new** routines.system.RowState();

**int** nb\_line\_tFileInputDelimited\_5 = 0;

org.talend.fileprocess.FileInputDelimited fid\_tFileInputDelimited\_5 = **null**;

**try** {

Object filename\_tFileInputDelimited\_5 = "E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/Inv/out\_accept\_inv.csv";

**if** (filename\_tFileInputDelimited\_5 **instanceof** java.io.InputStream) {

**int** footer\_value\_tFileInputDelimited\_5 = 0, random\_value\_tFileInputDelimited\_5 = -1;

**if** (footer\_value\_tFileInputDelimited\_5 > 0

|| random\_value\_tFileInputDelimited\_5 > 0) {

**throw** **new** java.lang.Exception(

"When the input source is a stream,footer and random shouldn't be bigger than 0.");

}

}

**try** {

fid\_tFileInputDelimited\_5 = **new** org.talend.fileprocess.FileInputDelimited(

"E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/Inv/out\_accept\_inv.csv",

"US-ASCII", "^", "\n", **false**, 1, 0, -1, -1,

**false**);

} **catch** (java.lang.Exception e) {

System.err.println(e.getMessage());

}

**while** (fid\_tFileInputDelimited\_5 != **null**

&& fid\_tFileInputDelimited\_5.nextRecord()) {

rowstate\_tFileInputDelimited\_5.reset();

cleaned\_sdl\_inv = **null**;

cleaned\_sdl\_inv = **null**;

**boolean** whetherReject\_tFileInputDelimited\_5 = **false**;

cleaned\_sdl\_inv = **new** cleaned\_sdl\_invStruct();

**try** {

**int** columnIndexWithD\_tFileInputDelimited\_5 = 0;

columnIndexWithD\_tFileInputDelimited\_5 = 0;

cleaned\_sdl\_inv.BibNum = fid\_tFileInputDelimited\_5

.get(columnIndexWithD\_tFileInputDelimited\_5);

columnIndexWithD\_tFileInputDelimited\_5 = 1;

cleaned\_sdl\_inv.Title = fid\_tFileInputDelimited\_5

.get(columnIndexWithD\_tFileInputDelimited\_5);

columnIndexWithD\_tFileInputDelimited\_5 = 2;

cleaned\_sdl\_inv.Author = fid\_tFileInputDelimited\_5

.get(columnIndexWithD\_tFileInputDelimited\_5);

columnIndexWithD\_tFileInputDelimited\_5 = 3;

cleaned\_sdl\_inv.Subjects = fid\_tFileInputDelimited\_5

.get(columnIndexWithD\_tFileInputDelimited\_5);

**if** (rowstate\_tFileInputDelimited\_5.getException() != **null**) {

**throw** rowstate\_tFileInputDelimited\_5

.getException();

}

} **catch** (java.lang.Exception e) {

whetherReject\_tFileInputDelimited\_5 = **true**;

System.err.println(e.getMessage());

cleaned\_sdl\_inv = **null**;

}

/\*\*

\* [tFileInputDelimited\_5 begin ] stop

\*/

/\*\*

\* [tFileInputDelimited\_5 main ] start

\*/

currentComponent = "tFileInputDelimited\_5";

tos\_count\_tFileInputDelimited\_5++;

/\*\*

\* [tFileInputDelimited\_5 main ] stop

\*/

/\*\*

\* [tFileInputDelimited\_5 process\_data\_begin ] start

\*/

currentComponent = "tFileInputDelimited\_5";

/\*\*

\* [tFileInputDelimited\_5 process\_data\_begin ] stop

\*/

// Start of branch "cleaned\_sdl\_inv"

**if** (cleaned\_sdl\_inv != **null**) {

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv main ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_sdl\_inv";

// cleaned\_sdl\_inv

// cleaned\_sdl\_inv

**if** (execStat) {

runStat.updateStatOnConnection(

"cleaned\_sdl\_inv" + iterateId, 1, 1);

}

cleaned\_sdl\_invStruct cleaned\_sdl\_inv\_HashRow = **new** cleaned\_sdl\_invStruct();

cleaned\_sdl\_inv\_HashRow.BibNum = cleaned\_sdl\_inv.BibNum;

cleaned\_sdl\_inv\_HashRow.Title = cleaned\_sdl\_inv.Title;

cleaned\_sdl\_inv\_HashRow.Author = cleaned\_sdl\_inv.Author;

cleaned\_sdl\_inv\_HashRow.Subjects = cleaned\_sdl\_inv.Subjects;

tHash\_Lookup\_cleaned\_sdl\_inv

.put(cleaned\_sdl\_inv\_HashRow);

tos\_count\_tAdvancedHash\_cleaned\_sdl\_inv++;

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv main ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv process\_data\_begin

\* ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_sdl\_inv";

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv process\_data\_begin

\* ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv process\_data\_end ]

\* start

\*/

currentComponent = "tAdvancedHash\_cleaned\_sdl\_inv";

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv process\_data\_end ]

\* stop

\*/

} // End of branch "cleaned\_sdl\_inv"

/\*\*

\* [tFileInputDelimited\_5 process\_data\_end ] start

\*/

currentComponent = "tFileInputDelimited\_5";

/\*\*

\* [tFileInputDelimited\_5 process\_data\_end ] stop

\*/

/\*\*

\* [tFileInputDelimited\_5 end ] start

\*/

currentComponent = "tFileInputDelimited\_5";

}

} **finally** {

**if** (!((Object) ("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/Inv/out\_accept\_inv.csv") **instanceof** java.io.InputStream)) {

**if** (fid\_tFileInputDelimited\_5 != **null**) {

fid\_tFileInputDelimited\_5.close();

}

}

**if** (fid\_tFileInputDelimited\_5 != **null**) {

globalMap.put("tFileInputDelimited\_5\_NB\_LINE",

fid\_tFileInputDelimited\_5.getRowNumber());

}

}

ok\_Hash.put("tFileInputDelimited\_5", **true**);

end\_Hash.put("tFileInputDelimited\_5",

System.currentTimeMillis());

/\*\*

\* [tFileInputDelimited\_5 end ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv end ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_sdl\_inv";

tHash\_Lookup\_cleaned\_sdl\_inv.endPut();

**if** (execStat) {

**if** (resourceMap.get("inIterateVComp") == **null**

|| !((Boolean) resourceMap.get("inIterateVComp"))) {

runStat.updateStatOnConnection("cleaned\_sdl\_inv"

+ iterateId, 2, 0);

}

}

ok\_Hash.put("tAdvancedHash\_cleaned\_sdl\_inv", **true**);

end\_Hash.put("tAdvancedHash\_cleaned\_sdl\_inv",

System.currentTimeMillis());

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv end ] stop

\*/

}// end the resume

} **catch** (java.lang.Exception e) {

TalendException te = **new** TalendException(e, currentComponent,

globalMap);

**throw** te;

} **catch** (java.lang.Error error) {

runStat.stopThreadStat();

**throw** error;

} **finally** {

**try** {

/\*\*

\* [tFileInputDelimited\_5 finally ] start

\*/

currentComponent = "tFileInputDelimited\_5";

/\*\*

\* [tFileInputDelimited\_5 finally ] stop

\*/

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv finally ] start

\*/

currentComponent = "tAdvancedHash\_cleaned\_sdl\_inv";

/\*\*

\* [tAdvancedHash\_cleaned\_sdl\_inv finally ] stop

\*/

} **catch** (java.lang.Exception e) {

// ignore

} **catch** (java.lang.Error error) {

// ignore

}

resourceMap = **null**;

}

globalMap.put("tFileInputDelimited\_5\_SUBPROCESS\_STATE", 1);

}

**public** String resuming\_logs\_dir\_path = **null**;

**public** String resuming\_checkpoint\_path = **null**;

**public** String parent\_part\_launcher = **null**;

**private** String resumeEntryMethodName = **null**;

**private** **boolean** globalResumeTicket = **false**;

**public** **boolean** watch = **false**;

// portStats is null, it means don't execute the statistics

**public** Integer portStats = **null**;

**public** **int** portTraces = 4334;

**public** String clientHost;

**public** String defaultClientHost = "localhost";

**public** String contextStr = "Default";

**public** **boolean** isDefaultContext = **true**;

**public** String pid = "0";

**public** String rootPid = **null**;

**public** String fatherPid = **null**;

**public** String fatherNode = **null**;

**public** **long** startTime = 0;

**public** **boolean** isChildJob = **false**;

**public** String log4jLevel = "";

**private** **boolean** execStat = **true**;

**private** ThreadLocal<java.util.Map<String, String>> threadLocal = **new** ThreadLocal<java.util.Map<String, String>>() {

**protected** java.util.Map<String, String> initialValue() {

java.util.Map<String, String> threadRunResultMap = **new** java.util.HashMap<String, String>();

threadRunResultMap.put("errorCode", **null**);

threadRunResultMap.put("status", "");

**return** threadRunResultMap;

};

};

**private** PropertiesWithType context\_param = **new** PropertiesWithType();

**public** java.util.Map<String, Object> parentContextMap = **new** java.util.HashMap<String, Object>();

**public** String status = "";

**public** **static** **void** main(String[] args) {

**final** SDL\_cleanup\_transformation SDL\_cleanup\_transformationClass = **new** SDL\_cleanup\_transformation();

**int** exitCode = SDL\_cleanup\_transformationClass.runJobInTOS(args);

System.exit(exitCode);

}

**public** String[][] runJob(String[] args) {

**int** exitCode = runJobInTOS(args);

String[][] bufferValue = **new** String[][] { { Integer.toString(exitCode) } };

**return** bufferValue;

}

**public** **boolean** hastBufferOutputComponent() {

**boolean** hastBufferOutput = **false**;

**return** hastBufferOutput;

}

**public** **int** runJobInTOS(String[] args) {

// reset status

status = "";

String lastStr = "";

**for** (String arg : args) {

**if** (arg.equalsIgnoreCase("--context\_param")) {

lastStr = arg;

} **else** **if** (lastStr.equals("")) {

evalParam(arg);

} **else** {

evalParam(lastStr + " " + arg);

lastStr = "";

}

}

**if** (clientHost == **null**) {

clientHost = defaultClientHost;

}

**if** (pid == **null** || "0".equals(pid)) {

pid = TalendString.getAsciiRandomString(6);

}

**if** (rootPid == **null**) {

rootPid = pid;

}

**if** (fatherPid == **null**) {

fatherPid = pid;

} **else** {

isChildJob = **true**;

}

**if** (portStats != **null**) {

// portStats = -1; //for testing

**if** (portStats < 0 || portStats > 65535) {

// issue:10869, the portStats is invalid, so this client socket

// can't open

System.err.println("The statistics socket port " + portStats

+ " is invalid.");

execStat = **false**;

}

} **else** {

execStat = **false**;

}

**try** {

// call job/subjob with an existing context, like:

// --context=production. if without this parameter, there will use

// the default context instead.

java.io.InputStream inContext = SDL\_cleanup\_transformation.**class**

.getClassLoader().getResourceAsStream(

"adm/sdl\_cleanup\_transformation\_0\_1/contexts/"

+ contextStr + ".properties");

**if** (inContext == **null**) {

inContext = SDL\_cleanup\_transformation.**class**

.getClassLoader()

.getResourceAsStream(

"config/contexts/" + contextStr + ".properties");

}

**if** (inContext != **null**) {

// defaultProps is in order to keep the original context value

defaultProps.load(inContext);

inContext.close();

context = **new** ContextProperties(defaultProps);

} **else** **if** (!isDefaultContext) {

// print info and job continue to run, for case: context\_param

// is not empty.

System.err.println("Could not find the context " + contextStr);

}

**if** (!context\_param.isEmpty()) {

context.putAll(context\_param);

// set types for params from parentJobs

**for** (Object key : context\_param.keySet()) {

String context\_key = key.toString();

String context\_type = context\_param

.getContextType(context\_key);

context.setContextType(context\_key, context\_type);

}

}

} **catch** (java.io.IOException ie) {

System.err.println("Could not load context " + contextStr);

ie.printStackTrace();

}

// get context value from parent directly

**if** (parentContextMap != **null** && !parentContextMap.isEmpty()) {

}

// Resume: init the resumeUtil

resumeEntryMethodName = ResumeUtil

.getResumeEntryMethodName(resuming\_checkpoint\_path);

resumeUtil = **new** ResumeUtil(resuming\_logs\_dir\_path, isChildJob, rootPid);

resumeUtil.initCommonInfo(pid, rootPid, fatherPid, projectName,

jobName, contextStr, jobVersion);

List<String> parametersToEncrypt = **new** java.util.ArrayList<String>();

// Resume: jobStart

resumeUtil.addLog("JOB\_STARTED", "JOB:" + jobName,

parent\_part\_launcher, Thread.currentThread().getId() + "", "",

"", "", "",

resumeUtil.convertToJsonText(context, parametersToEncrypt));

**if** (execStat) {

**try** {

runStat.openSocket(!isChildJob);

runStat.setAllPID(rootPid, fatherPid, pid, jobName);

runStat.startThreadStat(clientHost, portStats);

runStat.updateStatOnJob(RunStat.JOBSTART, fatherNode);

} **catch** (java.io.IOException ioException) {

ioException.printStackTrace();

}

}

java.util.concurrent.ConcurrentHashMap<Object, Object> concurrentHashMap = **new** java.util.concurrent.ConcurrentHashMap<Object, Object>();

globalMap.put("concurrentHashMap", concurrentHashMap);

**long** startUsedMemory = Runtime.getRuntime().totalMemory()

- Runtime.getRuntime().freeMemory();

**long** endUsedMemory = 0;

**long** end = 0;

startTime = System.currentTimeMillis();

tStatCatcher\_1.addMessage("begin");

**this**.globalResumeTicket = **true**;// to run tPreJob

**try** {

tStatCatcher\_1Process(globalMap);

} **catch** (java.lang.Exception e) {

e.printStackTrace();

}

**this**.globalResumeTicket = **false**;// to run others jobs

**try** {

errorCode = **null**;

tFileInputDelimited\_1Process(globalMap);

**if** (!"failure".equals(status)) {

status = "end";

}

} **catch** (TalendException e\_tFileInputDelimited\_1) {

globalMap.put("tFileInputDelimited\_1\_SUBPROCESS\_STATE", -1);

e\_tFileInputDelimited\_1.printStackTrace();

}

**this**.globalResumeTicket = **true**;// to run tPostJob

end = System.currentTimeMillis();

**if** (watch) {

System.out.println((end - startTime) + " milliseconds");

}

endUsedMemory = Runtime.getRuntime().totalMemory()

- Runtime.getRuntime().freeMemory();

**if** (**false**) {

System.out

.println((endUsedMemory - startUsedMemory)

+ " bytes memory increase when running : SDL\_cleanup\_transformation");

}

tStatCatcher\_1.addMessage(status == "" ? "end" : status,

(end - startTime));

**try** {

tStatCatcher\_1Process(globalMap);

} **catch** (java.lang.Exception e) {

e.printStackTrace();

}

**if** (execStat) {

runStat.updateStatOnJob(RunStat.JOBEND, fatherNode);

runStat.stopThreadStat();

}

**int** returnCode = 0;

**if** (errorCode == **null**) {

returnCode = status != **null** && status.equals("failure") ? 1 : 0;

} **else** {

returnCode = errorCode.intValue();

}

resumeUtil.addLog("JOB\_ENDED", "JOB:" + jobName, parent\_part\_launcher,

Thread.currentThread().getId() + "", "", "" + returnCode, "",

"", "");

**return** returnCode;

}

// only for OSGi env

**public** **void** destroy() {

}

**private** java.util.Map<String, Object> getSharedConnections4REST() {

java.util.Map<String, Object> connections = **new** java.util.HashMap<String, Object>();

**return** connections;

}

**private** **void** evalParam(String arg) {

**if** (arg.startsWith("--resuming\_logs\_dir\_path")) {

resuming\_logs\_dir\_path = arg.substring(25);

} **else** **if** (arg.startsWith("--resuming\_checkpoint\_path")) {

resuming\_checkpoint\_path = arg.substring(27);

} **else** **if** (arg.startsWith("--parent\_part\_launcher")) {

parent\_part\_launcher = arg.substring(23);

} **else** **if** (arg.startsWith("--watch")) {

watch = **true**;

} **else** **if** (arg.startsWith("--stat\_port=")) {

String portStatsStr = arg.substring(12);

**if** (portStatsStr != **null** && !portStatsStr.equals("null")) {

portStats = Integer.parseInt(portStatsStr);

}

} **else** **if** (arg.startsWith("--trace\_port=")) {

portTraces = Integer.parseInt(arg.substring(13));

} **else** **if** (arg.startsWith("--client\_host=")) {

clientHost = arg.substring(14);

} **else** **if** (arg.startsWith("--context=")) {

contextStr = arg.substring(10);

isDefaultContext = **false**;

} **else** **if** (arg.startsWith("--father\_pid=")) {

fatherPid = arg.substring(13);

} **else** **if** (arg.startsWith("--root\_pid=")) {

rootPid = arg.substring(11);

} **else** **if** (arg.startsWith("--father\_node=")) {

fatherNode = arg.substring(14);

} **else** **if** (arg.startsWith("--pid=")) {

pid = arg.substring(6);

} **else** **if** (arg.startsWith("--context\_type")) {

String keyValue = arg.substring(15);

**int** index = -1;

**if** (keyValue != **null** && (index = keyValue.indexOf('=')) > -1) {

**if** (fatherPid == **null**) {

context\_param.setContextType(keyValue.substring(0, index),

replaceEscapeChars(keyValue.substring(index + 1)));

} **else** { // the subjob won't escape the especial chars

context\_param.setContextType(keyValue.substring(0, index),

keyValue.substring(index + 1));

}

}

} **else** **if** (arg.startsWith("--context\_param")) {

String keyValue = arg.substring(16);

**int** index = -1;

**if** (keyValue != **null** && (index = keyValue.indexOf('=')) > -1) {

**if** (fatherPid == **null**) {

context\_param.put(keyValue.substring(0, index),

replaceEscapeChars(keyValue.substring(index + 1)));

} **else** { // the subjob won't escape the especial chars

context\_param.put(keyValue.substring(0, index),

keyValue.substring(index + 1));

}

}

} **else** **if** (arg.startsWith("--log4jLevel=")) {

log4jLevel = arg.substring(13);

}

}

**private** **static** **final** String NULL\_VALUE\_EXPRESSION\_IN\_COMMAND\_STRING\_FOR\_CHILD\_JOB\_ONLY = "<TALEND\_NULL>";

**private** **final** String[][] escapeChars = { { "\\\\", "\\" }, { "\\n", "\n" },

{ "\\'", "\'" }, { "\\r", "\r" }, { "\\f", "\f" }, { "\\b", "\b" },

{ "\\t", "\t" } };

**private** String replaceEscapeChars(String keyValue) {

**if** (keyValue == **null** || ("").equals(keyValue.trim())) {

**return** keyValue;

}

StringBuilder result = **new** StringBuilder();

**int** currIndex = 0;

**while** (currIndex < keyValue.length()) {

**int** index = -1;

// judege if the left string includes escape chars

**for** (String[] strArray : escapeChars) {

index = keyValue.indexOf(strArray[0], currIndex);

**if** (index >= 0) {

result.append(keyValue.substring(currIndex,

index + strArray[0].length()).replace(strArray[0],

strArray[1]));

currIndex = index + strArray[0].length();

**break**;

}

}

// if the left string doesn't include escape chars, append the left

// into the result

**if** (index < 0) {

result.append(keyValue.substring(currIndex));

currIndex = currIndex + keyValue.length();

}

}

**return** result.toString();

}

**public** Integer getErrorCode() {

**return** errorCode;

}

**public** String getStatus() {

**return** status;

}

ResumeUtil resumeUtil = **null**;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 315288 characters generated by Talend Open Studio for Data Integration on the

\* October 4, 2019 12:17:26 AM CDT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/