Приложение 3

ПРОГРАММНЫЙ МОДУЛЬ ПРОВЕДЕНИЯ ФИНАНСОВЫХ ОПЕРАЦИЙ НА POS-ТЕРМИНАЛЕ

ФРАГМЕНТЫ ИСХОДНОГО КОДА

Москва 2018

**TransactionGUI.java**

|  |
| --- |
| package application**.**activity**;**  **import** android**.**content**.**Context**;**  **import** android**.**os**.**Handler**;**  **import** helpers**.**Amount**;**  public interface TransactionGUI  **{**  Amount requestAmount**(**String title**,** String hint**);**  String requestUserInput**(**InputSpecs input\_specs**);**  String requestDecimalUserInput**(**String title**,** String hint**,** int min\_length**,** int max\_length**);**  int askUser**(**String question**,** String**[]** buttons\_strings**);**  int askUser**(**String question**,** String**[]** buttons\_strings**,** int**[]** buttons\_drawables**);**  PinPadActivity**.**PinData requestPIN**(**boolean online**,** final byte**[]** PAN**,** final byte**[]** keyMod**,** final byte**[]** keyExp**,** int count**);**  void showPopupInfo**(**String message**);**  void showPopupWarning**(**String message**);**  void showPopupError**(**String message**);**  void showInfo**(**String msg**);**  void showError**(**String msg**);**  void showProgressBar**();**  void hideProgressBar**();**  void enqueueTransitionAnimation**();**  void setOperationTitle**(**String title**);**  void setOperationInfoText**(**String text**);**  void setStatusImage**(**String name**);**  void setStatusImage**(**int drawable\_id**);**  void finishActivity**();**  Context getContext**();**  Handler getHandler**();**  String getLocalizedString**(**int id**);**  **}** |

**TransactionTask.java**

|  |
| --- |
| package application**.**activity**;**  **import** android**.**util**.**Log**;**  **import** com**.**szzt**.**sdk**.**device**.**DeviceManager**;**  **import** java**.**util**.**Date**;**  **import** application**.**Animations**;**  **import** application**.**Beepers**;**  **import** application**.**MainApplication**;**  **import** application**.**QuickDialogs**;**  **import** config**.**base**.**Currency**;**  **import** config**.**base**.**Readers**;**  **import** config**.**base**.**Terminal**;**  **import** database**.**journal**.**Journal**;**  **import** hardware**.**audio**.**Beeper**;**  **import** hardware**.**readers**.**AbstractReader**;**  **import** hardware**.**readers**.**Contactless**;**  **import** hardware**.**readers**.**MSR**;**  **import** hardware**.**readers**.**SCR**;**  **import** helpers**.**Amount**;**  **import** helpers**.**CryptoException**;**  **import** helpers**.**LogChannel**;**  **import** helpers**.**PopupMessage**;**  **import** helpers**.**Waiter**;**  **import** hosts**.**HostHolder**;**  **import** hosts**.**HostUtils**;**  **import** licence**.**LicenceManager**;**  **import** operation\_processors**.**ContactlessOperationProcessor**;**  **import** operation\_processors**.**DummyOperationProcessor**;**  **import** operation\_processors**.**EMVOperationProcessor**;**  **import** operation\_processors**.**MSROperationProcessor**;**  **import** operation\_processors**.**OperationProcessor**;**  **import** operations**.**Operation**;**  **import** operations**.**OperationContext**;**  **import** operations**.**OperationResult**;**  **import** receipts**.**OperationReceiptPrinter**;**  **import** termt**.**R**;**  **import** utils**.**ThreadUtils**;**  public class TransactionTask **extends** Thread  **{**  private Operation operation\_**;**  private TransactionGUI gui\_**;**  private final int kContactSCRFailuresToFallback **=** 3**;**  private final int kMaxRetriesCount **=** 5**;**  private LogChannel log\_ **=** **new** LogChannel**(**"trantask"**);**  private AbstractReader**[]** readers\_ **=** **{** SCR**.**SCRReader**.**getInstance**(),**  MSR**.**MSRReader**.**getInstance**(),**  Contactless**.**ContactlessReader**.**getInstance**()};**  //==========================================================================================  public TransactionTask**(**Operation operation**,** TransactionGUI gui**)**  **{**  operation\_ **=** operation**;**  gui\_ **=** gui**;**  log\_**.**logInfo**(**"task operation " **+** System**.**identityHashCode**(**operation\_**));**  **}**  //==========================================================================================  @Override public void run**()**  **{**  **try**  **{**  log\_**.**logInfo**(**"starting transaction"**);**  doTransaction**();**  **}**  **catch** **(**Exception e**)**  **{**  Log**.**e**(**"transtask"**,** "do transaction exception"**);**  e**.**printStackTrace**();**  **}**  **finally**  **{**  **try**  **{**  log\_**.**logInfo**(**"transaction post exec"**);**  log\_**.**logInfo**(**"finally operation " **+** System**.**identityHashCode**(**operation\_**));**  **if** **(**operation\_ **!=** **null)**  **{**  operation\_**.**context**().**setTrack1Data**(new** String**());**  operation\_**.**context**().**setTrack2Data**(new** String**());**  operation\_**.**context**().**setTrack3Data**(new** String**());**  operation\_**.**context**().**resetEncryptedPin**();**  **}**  **}**  **finally**  **{**  postExecute**();**  **}**  log\_**.**logInfo**(**"post exec finished"**);**  **}**  **}**  //==========================================================================================  private void doTransaction**()**  **{**  **if** **(**MainApplication**.**getLicenceNotFound**())**  **{**  log\_**.**logWarn**(**"licence not found"**);**  setNotProcessed**();**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**licence\_check\_on\_operation\_failed**));**  **return;**  **}**  **if** **(**operation\_**.**code**()** **==** Operation**.**OperationCode**.**OpSettlement**)**  **{**  **if** **(!**checkAndUpdateLicence**(true))**  **{**  setNotProcessed**();**  **return;**  **}**  **}**  **if** **(!**Journal**.**getInstance**().**inited**())**  **{**  setNotProcessed**();**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**journal\_not\_inited**));**  **return;**  **}**  gui\_**.**setOperationTitle**(**operation\_**.**name**());**  gui\_**.**setStatusImage**(**Animations**.**empty**());**  gui\_**.**showInfo**(**""**);**  **if** **(!**preProcessTransaction**())**  **{**  log\_**.**logInfo**(**"pre process failed"**);**  **return;**  **}**  updateTransactionInfo**();**  **try**  **{**  log\_**.**logInfo**(**"call main transaction processing"**);**  mainTransactionProcessing**();**  log\_**.**logInfo**(**"main transaction processing finished"**);**  **}**  **catch** **(**Exception e**)**  **{**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_exception\_toast**));**  e**.**printStackTrace**();**  **}**  **finally**  **{**  log\_**.**logInfo**(**"final operatin set finish"**);**  closeReaders**();**  **}**  logOperationIDs**();**  **}**  //==========================================================================================  private boolean preProcessTransaction**()**  **{**  **if** **(**Journal**.**inSettlement**()** **&&** operation\_**.**context**().**getOpCode**().**forbiddenDuringSettlement**())**  **{**  log\_**.**logInfo**(**"need settlement"**);**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_settlement\_required**));**  **return** **false;**  **}**  **if** **(!**checkTransactionAmounts**())**  **{**  log\_**.**logInfo**(**"amounts check failed"**);**  setNotProcessed**();**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**common\_invalid\_amount**));**  **return** **false;**  **}**  **if** **(!**HostHolder**.**host**().**operationSupported**(**operation\_**.**code**()))**  **{**  log\_**.**logInfo**(**"operation not supported"**);**  setNotProcessed**();**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_unsupported\_operation\_toast**));**  **return** **false;**  **}**  //invoice will be requested only if no ERN in context.  //there are 2 cases:  // - gui operation: no ern -> need invoice for void  // - ttk operation: has ern -> get invoice after from journal  **if** **(!**requestAmountIfNeeded**()** **||**  **!**requestRRNIfNeeded**()** **||**  **!**requestInvoiceNumberIfNeeded**())**  **{**  log\_**.**logInfo**(**"input request canceled"**);**  setCancelled**();**  **return** **false;**  **}**  **if** **(!**isAmountValid**(**operation\_**.**getInitialConditions**().**getTransactionAmount**()))**  **{**  log\_**.**logInfo**(**"invalid transaction amount"**);**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**Declined**);**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**common\_invalid\_amount**));**  **return** **false;**  **}**  **if** **(!**invoiceNumberPresents**())**  setInvoiceNumberToInitialCondition**(**HostUtils**.**getNextInvoiceNumber**());**  **if** **((**operation\_**.**code**()** **==** Operation**.**OperationCode**.**OpDirectVoid **||** operation\_**.**code**()** **==** Operation**.**OperationCode**.**OpAdjustment**)** **&&**  **!**initVoidTransaction**())**  **{**  log\_**.**logInfo**(**"original operation not found for void"**);**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**OperationNotFound**);**  **return** **false;**  **}**  **if** **(**operation\_**.**code**()** **==** Operation**.**OperationCode**.**OpAdjustment**)**  **{**  **if** **(**isAdjustmentAmountExceed**())**  **{**  log\_**.**logInfo**(**"adjustment amount exceed"**);**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_amount\_exceed**));**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**OperationCanceled**);**  **return** **false;**  **}**  **if** **(**isAdjustmentFullAmount**())**  **{**  log\_**.**logInfo**(**"adjustment is full-amount, processing as VOID"**);**  operation\_**.**changeOperationCode**(**Operation**.**OperationCode**.**OpDirectVoid**);**  **}**  **}**  **return** **true;**  **}**  //==========================================================================================  boolean initVoidTransaction**()**  **{**  OperationContext ctx**;**  **try**  **{**  ctx **=** findOperationInJournalByOneOfID**(**operation\_**.**getInitialConditions**().**getERN**(),**  operation\_**.**getInitialConditions**().**getRRN**(),**  operation\_**.**getInitialConditions**().**getInvoiceNumber**());**  **}**  **catch** **(**Journal**.**JournalException e**)**  **{**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**journal\_not\_inited**));**  **return** **false;**  **}**  **if** **(**ctx **==** **null)**  **{**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_cant\_find\_toast**));**  **return** **false;**  **}**  fillVoidOriginalData**(**ctx**);**  operation\_**.**setContext**(**ctx**);**  **return** **true;**  **}**  //==========================================================================================  static public void fillVoidOriginalData**(**OperationContext void\_context\_from\_orig**)**  **{**  void\_context\_from\_orig**.**setOrigOpCode**(**void\_context\_from\_orig**.**getOpCode**());**  void\_context\_from\_orig**.**setOrigSTAN**(**void\_context\_from\_orig**.**getSTAN**());**  void\_context\_from\_orig**.**setOrigAcqInstCode**(**void\_context\_from\_orig**.**getAcqInstCode**());**  void\_context\_from\_orig**.**setOrigTrDateTime**(**void\_context\_from\_orig**.**getTrDateTime**());**  void\_context\_from\_orig**.**setIsVoidOp**(true);**  **}**  //==========================================================================================  // return false means no result message (canceled operation)  private boolean mainTransactionProcessing**()**  **{**  int retries\_count **=** 0**;**  int contact\_scr\_card\_failures **=** 0**;**  AbstractReader reader **=** **null;**  OperationProcessor**.**OperationStatus status **=** **null;**  doSCRReaderConflictWorkaround**();**  **while** **(true)**  **{**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**NotProcessed**);**  gui\_**.**hideProgressBar**();**  **if** **(**operation\_**.**cancelRequested**())**  **{**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**OperationCanceled**);**  **break;**  **}**  OperationProcessor card\_processor**;**  **if** **(!**operation\_**.**needCardProcessign**())**  **{**  log\_**.**logInfo**(**"no card processing"**);**  card\_processor **=** **new** DummyOperationProcessor**(**reader**,** operation\_**,** gui\_**);**  operation\_**.**setCardInputed**(false);**  **}**  **else**  **{**  clearCardInputInfo**();**  log\_**.**logInfo**(**"waiting for card input"**);**  gui\_**.**setStatusImage**(**Animations**.**waitForCard**());**  gui\_**.**showInfo**(**waitForCardMessage**(**status**));**  reader **=** waitForCard**();**  operation\_**.**setCardInputed**(true);**  **if** **(**operation\_**.**cancelRequested**())**  **{**  log\_**.**logInfo**(**"operation cancel requested"**);**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**OperationCanceled**);**  //check if not first processing: no receipt need on cancel before processing start  **if** **(**status **!=** **null)**  **break;**  **else**  **return** **false;**  **}**  card\_processor **=** procForReader**(**reader**);**  **}**  **if** **(**card\_processor **==** **null)**  **{**  log\_**.**logInfo**(**"card proc is null"**);**  **return** **false;**  **}**  gui\_**.**setStatusImage**(**Animations**.**processing**());**  gui\_**.**showInfo**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_processing**));**  gui\_**.**showProgressBar**();**  log\_**.**logInfo**(**"run card proc"**);**  status **=** card\_processor**.**process**();**  operation\_**.**context**().**resetEncryptedPin**();**  log\_**.**logInfo**(**"card proc finished"**);**  **++**retries\_count**;**  **if** **(!**needRetryCardInput**(**status**,** retries\_count**))**  **{**  log\_**.**logInfo**(**"not need to reetry"**);**  **break;**  **}**  log\_**.**logInfo**(**"waiting for card remove"**);**  askAndWaitForRemove**(**reader**);**  **if** **(**status **==** OperationProcessor**.**OperationStatus**.**CDCVM**)**  **{**  log\_**.**logInfo**(**"CDCVM message"**);**  gui\_**.**setStatusImage**(**Animations**.**lookAtDevice**());**  gui\_**.**showInfo**(**gui\_**.**getLocalizedString**(**R**.**string**.**contactless\_see\_device**));**  ThreadUtils**.**sleep**(**3000**);**  **}**  closeReaders**();**  **if** **(**card\_processor**.**isContactSCRProcessor**())**  **{**  log\_**.**logInfo**(**"fallback checks"**);**  **++**contact\_scr\_card\_failures**;**  **if** **(!**operation\_**.**inFallbackEnterState**()** **&&**  **(**contact\_scr\_card\_failures **>=** kContactSCRFailuresToFallback**))**  **{**  log\_**.**logInfo**(**"going fallback"**);**  operation\_**.**setInFallbackEnterState**(true);**  closeReaders**();**  readers\_ **=** getReadersWithoutContactSCR**();**  **}**  **}**  **}**  **if** **(**status **==** OperationProcessor**.**OperationStatus**.**TransportError**)**  **{**  log\_**.**logInfo**(**"transport error status"**);**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**Declined**);**  **}**  showResultInfo**();**  final Waiter result\_waiter **=** **new** Waiter**(**needShowResult**()** **?** Terminal**.**getTerminalUI**().**final\_screen\_timeout **:** 0**);**  **if** **(**operation\_**.**code**().**needPrintReceipt**()** **&&** **!**operation\_**.**getInitialConditions**().**getTransactionMode**().**no\_receipts**)**  **{**  processReceipt**(!**needShowResult**());**  **}**  gui\_**.**hideProgressBar**();**  log\_**.**logInfo**(**"result screen wait"**);**  result\_waiter**.**waitFinished**();**  log\_**.**logInfo**(**"waiting for card remove"**);**  askAndWaitForRemove**(**reader**);**  log\_**.**logInfo**(**"card remove done"**);**  **return** **true;**  **}**  //==========================================================================================  private void processReceipt**(**boolean draw\_gui\_info**)**  **{**  log\_**.**logInfo**(**"processing receipt"**);**  **if** **(**draw\_gui\_info**)**  **{**  gui\_**.**setStatusImage**(**Animations**.**printing**());**  gui\_**.**showInfo**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_printing**));**  **}**  gui\_**.**showProgressBar**();**  **try**  **{**  OperationReceiptPrinter**.**printOperationReceipt**(**operation\_**.**context**(),** gui\_**.**getContext**());**  **}**  **catch** **(**Exception e**)**  **{**  e**.**printStackTrace**();**  gui\_**.**showPopupError**(**e**.**getMessage**());**  **}**  **finally**  **{**  **if** **(**draw\_gui\_info**)**  **{**  gui\_**.**setStatusImage**(**Animations**.**empty**());**  gui\_**.**showInfo**(**""**);**  **}**  gui\_**.**hideProgressBar**();**  **}**  **}**  //==========================================================================================  private void postExecute**()**  **{**  **try**  **{**  **if** **(**operation\_ **!=** **null** **&&**  operation\_**.**code**()** **!=** Operation**.**OperationCode**.**OpSettlement **&&**  **(**Journal**.**getFinTransactionCounter**()** **%** 5 **==** 1**))**  **{**  checkAndUpdateLicence**(true);**  **}**  **}**  **catch** **(**Exception e**)**  **{**  log\_**.**logError**(**"exception during licence check"**);**  MainApplication**.**setLicenceNotFound**(true);**  **}**  **finally**  **{**  **if** **(**operation\_ **!=** **null)**  **{**  operation\_**.**finish**();**  log\_**.**logInfo**(**"force finishing done"**);**  **}**  gui\_**.**enqueueTransitionAnimation**();**  log\_**.**logInfo**(**"pre finish activity"**);**  gui\_**.**finishActivity**();**  log\_**.**logInfo**(**"post finish activity"**);**  **}**  **}**  //==========================================================================================  private void setNotProcessed**()**  **{**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**NotProcessed**);**  **}**  //==========================================================================================  private void setCancelled**()**  **{**  operation\_**.**requestCancel**();**  operation\_**.**setTransactionDecision**(**OperationResult**.**TransactionDecision**.**OperationCanceled**);**  **}**  //==========================================================================================  private void showResultInfo**()**  **{**  **if** **(!**needShowResult**())**  **return;**  final OperationResult**.**TransactionDecision decision **=** operation\_**.**context**().**getTransactionDecision**();**  String result\_info**;**  int drawable\_id**;**  **if** **(**operation\_**.**context**().**getOpCode**()** **==** Operation**.**OperationCode**.**OpSettlement **&&**  decision **==** OperationResult**.**TransactionDecision**.**Accepted **&&**  operation\_**.**getSettlementNotNeeded**())**  **{**  result\_info **=** gui\_**.**getLocalizedString**(**R**.**string**.**result\_settlement\_not\_needed**);**  drawable\_id **=** R**.**drawable**.**success\_icon**;**  **}**  **else**  **{**  result\_info **=** OperationResult**.**buildResultText**(**operation\_**.**context**().**getResponseCode**(),** operation\_**.**context**().**getTransactionDecision**());**  **if** **(**decision **==** OperationResult**.**TransactionDecision**.**Accepted **||**  decision **==** OperationResult**.**TransactionDecision**.**HostApproved**)**  **{**  drawable\_id **=** R**.**drawable**.**success\_icon**;**  **}**  **else**  **{**  drawable\_id **=** R**.**drawable**.**declined\_icon**;**  **}**  **}**  gui\_**.**hideProgressBar**();**  gui\_**.**setStatusImage**(**drawable\_id**);**  gui\_**.**showInfo**(**result\_info**);**  **}**  //==========================================================================================  private boolean needShowResult**()**  **{**  **return** **!**operation\_**.**getInitialConditions**().**getTransactionMode**().**no\_indication**;**  **}**  //==========================================================================================  private OperationContext findOperationInJournalByOneOfID**(**Long ERN**,** String RRN**,** Integer invoice**)** **throws** Journal**.**JournalException  **{**  OperationContext ctx **=** Journal**.**getInstance**().**getNormalOperationByERN**(**Journal**.**getBatchNumber**(),** ERN**,** **false);**  **if** **(**ctx **!=** **null)**  **return** ctx**;**  ctx **=** Journal**.**getInstance**().**getNormalOperationByRRN**(**Journal**.**getBatchNumber**(),** RRN**);**  **if** **(**ctx **!=** **null)**  **return** ctx**;**  ctx **=** Journal**.**getInstance**().**getNormalOperationByReceipeNumber**(**Journal**.**getBatchNumber**(),** invoice**);**  **return** ctx**;**  **}**  //==========================================================================================  private boolean isAmountValid**(**Amount transactionAmount**)**  **{**  **if** **(**Terminal**.**getTerminalConfiguration**().**allow\_zero\_amount**)**  **return** **true;**  **if** **(**transactionAmount **==** **null)**  **return** **true;**  **return** transactionAmount**.**getAmount**()** **>** 0**;**  **}**  //==========================================================================================  private boolean invoiceNumberPresents**()**  **{**  **return** operation\_**.**getInitialConditions**().**getInvoiceNumber**()** **!=** **null;**  **}**  //==========================================================================================  private boolean isSettlementNeeded**()** **{**  **return** **!**Journal**.**getInstance**().**isSettled**();**  **}**  //==========================================================================================  private boolean needRetryCardInput**(**OperationProcessor**.**OperationStatus status**,** int retries\_count**)**  **{**  **if** **(**status **==** OperationProcessor**.**OperationStatus**.**NeedRepeat**)**  **return** **true;**  **if** **(**status **==** OperationProcessor**.**OperationStatus**.**TransportError **&&** retries\_count **<** kMaxRetriesCount**)**  **return** **true;**  **if** **(**status **==** OperationProcessor**.**OperationStatus**.**CDCVM**)**  **return** **true;**  **if** **(**status **==** OperationProcessor**.**OperationStatus**.**NeedUseAnotherInterface**)**  **return** **true;**  **return** **false;**  **}**  //==========================================================================================  private OperationProcessor procForReader**(**AbstractReader reader**)**  **{**  **if** **(**reader **==** **null)**  **return** **null;**  **if** **(**reader**.**getClass**()** **==** SCR**.**SCRReader**.**class**)**  **{**  **return** **new** EMVOperationProcessor**(**reader**,** operation\_**,** gui\_**);**  **}**  **if** **(**reader**.**getClass**()** **==** Contactless**.**ContactlessReader**.**class**)**  **{**  **return** **new** ContactlessOperationProcessor**(**reader**,** operation\_**,** gui\_**);**  **}**  **if** **(**reader**.**getClass**()** **==** MSR**.**MSRReader**.**class**)**  **{**  **return** **new** MSROperationProcessor**(**reader**,** operation\_**,** gui\_**);**  **}**  **return** **null;**  **}**  //==========================================================================================  private boolean requestAmountIfNeeded**()**  **{**  **if** **(!**operation\_**.**needAmount**()** **||** **(**operation\_**.**getInitialConditions**().**getTransactionAmount**()** **!=** **null))**  **return** **true;**  final Amount amount **=** gui\_**.**requestAmount**(**operation\_**.**name**(),** gui\_**.**getLocalizedString**(**R**.**string**.**amount\_enter**));**  **if** **(**amount **==** **null** **||** **!**checkAmountLimit**(**amount**))**  **return** **false;**  setAmountToInitialCondition**(**amount**);**  **return** **true;**  **}**  //==========================================================================================  private boolean checkTransactionAmounts**()**  **{**  **return** checkAmountLimit**(**operation\_**.**getInitialConditions**().**getTransactionAmount**())** **&&**  checkAmountLimit**(**operation\_**.**context**().**getTransactionAmount**());**  **}**  //==========================================================================================  private boolean checkAmountLimit**(**Amount value**)**  **{**  **return** **(**value **==** **null** **||** **!**value**.**greater**(**Terminal**.**MAX\_AMOUNT\_VALUE**));**  **}**  //==========================================================================================  private boolean requestRRNIfNeeded**()**  **{**  **if** **(!**operation\_**.**needExternalRRN**()** **||** **(**operation\_**.**getInitialConditions**().**getRRN**()** **!=** **null** **&&** **!**operation\_**.**getInitialConditions**().**getRRN**().**isEmpty**()))**  **return** **true;**  final String res **=** gui\_**.**requestDecimalUserInput**(**operation\_**.**name**(),** gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_input\_rrn**),** 1**,** 12**);**  **if** **(**res **==** **null)**  **return** **false;**  setRRNToInitialCondition**(**res**);**  **return** **true;**  **}**  //==========================================================================================  private boolean requestInvoiceNumberIfNeeded**()**  **{**  **if** **(!**operation\_**.**needExternalInvoiceNumber**()** **||** invoiceNumberPresents**()** **||** ernPresents**())**  **return** **true;**  final String res **=** gui\_**.**requestDecimalUserInput**(**operation\_**.**name**(),** gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_input\_invoice**),** 1**,** 6**);**  **if** **(**res **==** **null)**  **return** **false;**  int res\_as\_int **=** 0**;**  **try**  **{**  res\_as\_int **=** Integer**.**parseInt**(**res**);**  **}**  **catch** **(**NumberFormatException e**)**  **{**  gui\_**.**showPopupError**(**"Invoice must be INTEGER"**);**  **return** **false;**  **}**  setInvoiceNumberToInitialCondition**(**res\_as\_int**);**  **return** **true;**  **}**  //==========================================================================================  private boolean ernPresents**()** **{**  **return** operation\_**.**getInitialConditions**().**getERN**()** **!=** **null;**  **}**  //==========================================================================================  private void updateTransactionInfo**()**  **{**  setTransactionInfoText**(**operation\_**.**code**().**localizedName**(),** buildTransactionInfoText**());**  **}**  //==========================================================================================  private String buildTransactionInfoText**()**  **{**  final Operation**.**OperationCode current\_code **=** operation\_**.**code**();**  **if** **(**current\_code **==** Operation**.**OperationCode**.**OpSettlement**)**  **{**  **return** gui\_**.**getLocalizedString**(**R**.**string**.**main\_journal**)** **+** " #" **+** Integer**.**toString**(**Journal**.**getBatchNumber**());**  **}**  final Amount current\_amount **=** getShowableAmount**();**  **if** **(**current\_amount **!=** **null)**  **{**  **return** current\_amount**.**getDottedStr**()** **+** " " **+** Currency**.**getCurrency**().**name**;**  **}**  **if** **(**operation\_**.**needAmount**())**  **{**  **return** gui\_**.**getLocalizedString**(**R**.**string**.**amount\_error**);**  **}**  **if** **(**current\_code **==** Operation**.**OperationCode**.**OpDirectVoid**)**  **{**  final Integer current\_invoice\_number **=** getShowableInvoiceNumber**();**  **if** **(**current\_invoice\_number **!=** **null)**  **return** gui\_**.**getLocalizedString**(**R**.**string**.**invoice\_number\_info**)** **+** " " **+** Integer**.**toString**(**current\_invoice\_number**);**  **}**  **else** **if** **(**current\_code **==** Operation**.**OperationCode**.**OpRefund**)**  **{**  final String current\_rrn **=** getShowableRRN**();**  **if** **(**current\_rrn **!=** **null)**  **return** gui\_**.**getLocalizedString**(**R**.**string**.**rrn\_info**)** **+** " " **+** current\_rrn**;**  **}**  **return** ""**;**  **}**  //==========================================================================================  private Amount getShowableAmount**()**  **{**  Amount showable\_amount **=** operation\_**.**getInitialConditions**().**getTransactionAmount**();**  **if** **(**showable\_amount **==** **null)**  showable\_amount **=** operation\_**.**context**().**getTransactionAmount**();**  **return** showable\_amount**;**  **}**  //==========================================================================================  private Integer getShowableInvoiceNumber**()**  **{**  Integer invoice\_number **=** operation\_**.**getInitialConditions**().**getInvoiceNumber**();**  **if** **(**invoice\_number **==** **null)**  invoice\_number **=** operation\_**.**context**().**getInvoiceNumber**();**  **return** invoice\_number**;**  **}**  //==========================================================================================  private String getShowableRRN**()**  **{**  String rrn **=** operation\_**.**getInitialConditions**().**getRRN**();**  **if** **(**rrn **==** **null)**  rrn **=** operation\_**.**context**().**getRRN**();**  **return** rrn**;**  **}**  //==========================================================================================  private void setTransactionInfoText**(**String opname**,** String opparams**)**  **{**  gui\_**.**setOperationTitle**(**opname**);**  gui\_**.**setOperationInfoText**(**opparams**);**  **}**  //==========================================================================================  private void setAmountToInitialCondition**(**Amount amount**)**  **{**  operation\_**.**getInitialConditions**().**setTransactionAmount**(**amount**);**  operation\_**.**getInitialConditions**().**setCurrencyName**(**Currency**.**getCurrency**().**name**);**  operation\_**.**getInitialConditions**().**setCurrencyCode**(**Currency**.**getCurrency**().**code**);**  **}**  //==========================================================================================  private void setInvoiceNumberToInitialCondition**(**Integer rid**)**  **{**  operation\_**.**getInitialConditions**().**setInvoiceNumber**(**rid**);**  **}**  //==========================================================================================  private void setRRNToInitialCondition**(**String rrn**)**  **{**  operation\_**.**getInitialConditions**().**setRRN**(**rrn**);**  **}**  //==========================================================================================  private void clearCardInputInfo**()**  **{**  log\_**.**logInfo**(**"clearing card input..."**);**  operation\_**.**context**().**setTagProxy**(null);**  operation\_**.**context**().**setCardInputInfo**(null);**  operation\_**.**context**().**setTrack1Data**(new** String**());**  operation\_**.**context**().**setTrack2Data**(new** String**());**  operation\_**.**context**().**setTrack3Data**(new** String**());**  operation\_**.**context**().**resetEncryptedPin**();**  operation\_**.**context**().**setAIDHEX**(null);**  operation\_**.**context**().**setAIDLable**(null);**  operation\_**.**context**().**setCardIssuerName**(null);**  operation\_**.**context**().**setTVR**(null);**  operation\_**.**context**().**setEmvResponseData**(null);**  **try**  **{**  operation\_**.**context**().**setPAN**(null);**  **}**  **catch** **(**CryptoException e**)**  **{**  log\_**.**logError**(**"crypto problem while clearing card input PAN"**);**  e**.**printStackTrace**();**  **}**  **try**  **{**  operation\_**.**context**().**setExpDate**(null);**  **}**  **catch** **(**CryptoException e**)**  **{**  log\_**.**logError**(**"crypto problem while clearing card input expdate"**);**  e**.**printStackTrace**();**  **}**  **try**  **{**  operation\_**.**context**().**setServiceCode**(null);**  **}**  **catch** **(**CryptoException e**)**  **{**  log\_**.**logError**(**"crypto problem while clearing card input service code"**);**  e**.**printStackTrace**();**  **}**  log\_**.**logInfo**(**"clearing card input done"**);**  **}**  //==========================================================================================  boolean checkAndUpdateLicence**(**boolean update\_animation**)**  **{**  log\_**.**logInfo**(**"licence check"**);**  **if** **(**MainApplication**.**getLicenceNotFound**())**  **{**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**licence\_check\_on\_operation\_failed**));**  **return** **false;**  **}**  **if** **(**update\_animation**)**  gui\_**.**setStatusImage**(**Animations**.**processing**());**  **if** **(!**LicenceManager**.**getInstance**().**checkAppLicence**())**  **{**  MainApplication**.**setLicenceNotFound**(true);**  **;**  gui\_**.**showPopupError**(**gui\_**.**getLocalizedString**(**R**.**string**.**licence\_check\_on\_operation\_failed**));**  **return** **false;**  **}**  **return** **true;**  **}**  //==========================================================================================  private AbstractReader waitForCard**()**  **{**  Readers**.**CardReaderSettings settings **=** Readers**.**getCardReaderSettings**();**  **return** waitForCardFromAnyReader**(**settings**.**wait\_timeout\_ms**);**  **}**  //==========================================================================================  private AbstractReader waitForCardFromAnyReader**(**int timeout**)**  **{**  final int reader\_pol\_timeout\_ms **=** 100**;**  **try**  **{**  **for** **(**AbstractReader r **:** readers\_**)**  **{**  **if** **(!**r**.**openCardReader**(**DeviceManager**.**getInstance**()))**  **{**  log\_**.**logError**(**"failed to open reader"**);**  **}**  **else**  **{**  r**.**startWaiting**(**timeout**);**  **}**  **}**  final long start\_time **=** **new** Date**().**getTime**();**  **while** **((new** Date**().**getTime**())** **-** start\_time **<=** timeout**)**  **{**  **for** **(**AbstractReader curr\_reader **:** readers\_**)**  **{**  **if** **(**curr\_reader**.**cardInside**())**  **return** curr\_reader**;**  Thread**.**sleep**(**reader\_pol\_timeout\_ms**);**  **if** **(**operation\_**.**cancelRequested**())**  **return** **null;**  **}**  **}**  **}**  **catch** **(**Exception e**)**  **{**  e**.**printStackTrace**();**  **}**  **return** **null;**  **}**  //==========================================================================================  private boolean askAndWaitForRemove**(**AbstractReader reader**)**  **{**  **if** **(**reader **==** **null** **||** **!**reader**.**cardInside**())**  **return** **true;**  gui\_**.**setStatusImage**(**Animations**.**removeCard**());**  gui\_**.**showInfo**(**gui\_**.**getLocalizedString**(**R**.**string**.**transaction\_remove\_card**));**  gui\_**.**hideProgressBar**();**  **try**  **{**  final Waiter full\_timeout\_waiter **=** **new** Waiter**(**Readers**.**getCardReaderSettings**().**remove\_timeout\_ms**);**  **while** **(!**reader**.**waitForRemove**(**3000**))**  **{**  Beepers**.**alertBeeper**().**playSound**();**  **if** **(**full\_timeout\_waiter**.**finished**())**  **{**  Log**.**i**(**"transaction"**,** "wait for remove timeout, retrying..."**);**  full\_timeout\_waiter**.**reset**();**  **}**  **}**  **}**  **catch** **(**InterruptedException e**)**  **{**  e**.**printStackTrace**();**  **}**  gui\_**.**setStatusImage**(**Animations**.**empty**());**  gui\_**.**showInfo**(**""**);**  gui\_**.**hideProgressBar**();**  **return** **true;**  **}**  //==========================================================================================  private String waitForCardMessage**(**OperationProcessor**.**OperationStatus prev\_status**)**  **{**  int id**;**  **if** **(**operation\_**.**inFallbackEnterState**())**  **{**  id **=** R**.**string**.**transaction\_card\_instruction\_fallback**;**  **}**  **else** **if** **(**prev\_status **==** OperationProcessor**.**OperationStatus**.**NeedUseAnotherInterface**)**  **{**  id **=** R**.**string**.**transaction\_card\_instruction\_use\_contact\_card**;**  **}**  **else**  **{**  id **=** R**.**string**.**transaction\_card\_instruction**;**  **}**  **return** gui\_**.**getLocalizedString**(**id**);**  **}**  //==========================================================================================  private AbstractReader**[]** getReadersWithoutContactSCR**()**  **{**  **return** **new** AbstractReader**[]** **{** MSR**.**MSRReader**.**getInstance**(),**  Contactless**.**ContactlessReader**.**getInstance**()};**  **}**  //==========================================================================================  private boolean closeReaders**()**  **{**  log\_**.**logInfo**(**"closing readers start"**);**  **for** **(**AbstractReader reader **:** readers\_**)**  **{**  reader**.**closeCardReader**();**  **}**  log\_**.**logInfo**(**"closing readers end"**);**  **return** **true;**  **}**  //==========================================================================================  private void doSCRReaderConflictWorkaround**()**  **{**  final AbstractReader scr\_reader **=** SCR**.**SCRReader**.**getInstance**();**  **if** **(**scr\_reader **!=** **null)**  **{**  scr\_reader**.**openCardReader**(**DeviceManager**.**getInstance**());**  scr\_reader**.**closeCardReader**();**  **}**  **}**  //==========================================================================================  public boolean isAdjustmentFullAmount**()**  **{**  **return** operation\_**.**getInitialConditions**().**getTransactionAmount**().**equals**(**operation\_**.**context**().**getTransactionAmount**());**  **}**  //==========================================================================================  public boolean isAdjustmentAmountExceed**()**  **{**  **return** operation\_**.**getInitialConditions**().**getTransactionAmount**().**greater**(**operation\_**.**context**().**getTransactionAmount**());**  **}**  //==========================================================================================  private void logOperationIDs**()**  **{**  Log**.**d**(**"TranID"**,** "Op "**+** operation\_**.**context**().**getOpCode**());**  final String initrrn **=** String**.**valueOf**(**operation\_**.**getInitialConditions**().**getRRN**());**  final String initinvoice **=** String**.**valueOf**(**operation\_**.**getInitialConditions**().**getInvoiceNumber**());**  final String initern **=** String**.**valueOf**(**operation\_**.**getInitialConditions**().**getERN**());**  final String ctxrrn **=** String**.**valueOf**(**operation\_**.**context**().**getRRN**());**  final String ctxinvoice **=** String**.**valueOf**(**operation\_**.**context**().**getInvoiceNumber**());**  final String ctxern **=** String**.**valueOf**(**operation\_**.**context**().**getERN**());**  Log**.**d**(**"TranID"**,** "RRN i=" **+** initrrn **+** " c=" **+** ctxrrn**);**  Log**.**d**(**"TranID"**,** "ERN i=" **+** initern **+** " c=" **+** ctxern**);**  Log**.**d**(**"TranID"**,** "INV i=" **+** initinvoice **+** " c=" **+** ctxinvoice**);**  **}**  **}** |