AAMS

(Automatic Assignment Management System)

Group 21

Ajanth Thangavelu

Roberto Castañeda Lozano

Tony Karlsson

Love Jädergård

Design document

Contents

5. Design details

5.5. Detailed Design

Global data model

The organization of the data managed by the system is logically described with the next Entity-Relationship (ER) diagrams and attribute descriptions for every class:

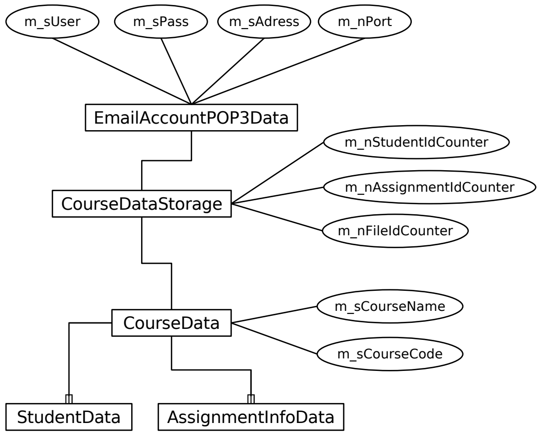

Figure 1: ER diagram for all the information that must be stored for a course

class CourseDataStorage

- *CourseData m_courseData*: all the information that belongs to a course (see below).
- *int m_nStudentIdCounter*: number of students in a course.
- *int m_nAssignmentIdCounter*: number of defined assignments for a course.
- *int m_nFileIdCounter*; number of files for a course.
- *EmailAccountPOP3Data m_POP3*: information about the selected POP3 account for a course (see below).

class EmailAccountPOP3Data

- *std::string m_sUser*: user name for the POP3 account.
- *std::string m_sPass*: password for the POP3 account.
- *std::string m_sAddress*: address of the POP3 server.
- *int m_nPort*: port for the POP3 protocol in the target server.

class CourseData

- *std::wstring m_sCourseName*: name of the course.
- *std::wstring m_sCourseCode*: code of the course.
- *std::list<StudentData> m_StudentData*: list of students and their assignments (see below).
- *std::list<AssignmentInfoData> m_assignmentInfoData*: list of defined assignments (see below).

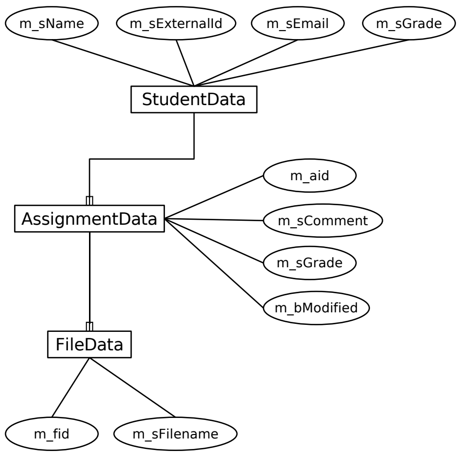

Figure 2: ER diagram showing the data model for students and assignments

class StudentData

- *int m_sid*: internal identifier of the student.
- *std::wstring m_sName*: name of the student.
- *std::wstring m_sExternalId*: external (institutional) identifier of the student.
- *std::wstring m_sEmail*: e-mail address of the student.
- *std::wstring m_sGrade*: global grade of the student.
- *std::list<AssignmentData> m_assignmentData*: list of assignments of the student (see below).

class AssignmentData

- *int m_aid*: internal identifier of the assignment.
- *std::wstring m_sComment*: teacher's comment on the assignment.
- *std::wstring m_sGrade*: grade of the assignment.
- *bool m_bModified*: specifies if the comments or the grade have been modified since the last time it was notified to the student.

- *std::list<FileData> m_fileData*: list of files that the assignment contains (see below).

class FileData

- *int m_fid*: internal identifier of the file.
- *std::wstring m_sFilename*: path of the file.

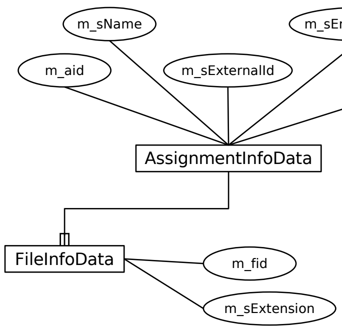

Figure 3: ER diagram showing the data model for the definition of assignments

class AssignmentInfoData

- *int m_aid*: internal identifier of the assignment definition.
- *std::wstring m_sName*: name of the assignment.
- *std::wstring m_sExternalId*: external (institutional) identifier of the assignment.
- *std::wstring m_sEmailHeader*: expected e-mail subject for sending the assignment.
- *int m_nFileCount*: expected number of files attached to the assignment.
- *std::list<FileInfoData> m_fileInfoData*: list of expected file formats of the assignment (see below).

class FileInfoData

- *int m_fid*: internal identifier of the file format.
- *std::wstring m_sExtension*: file extension.

class UnsentEmailsStorage

- *int m_nEmailIdCounter*: number of unsent e-mails.
- *std::list< UnsentEmailsData> m_unsentEmailsData*: list of unsent e-mails (see below).

class UnsentEmailsData

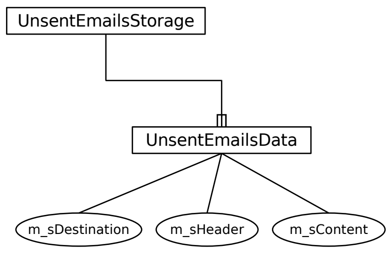

Figure 4: ER diagram showing the data model for the unsent e-mails

- *string m_sDestination*: destination address of the unsent e-mail.
- *string m_sHeader*: header of the unsent e-mail.
- *string m_sContent*: content of the unsent e-mail.

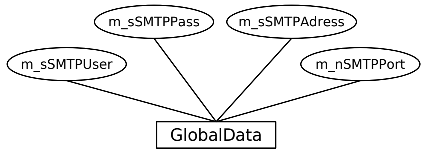

Figure 5: ER diagram showing the data model for the global settings of the system

class GlobalData

- *std::string m_sSMTPUser*: user name for the SMTP account.
- *std::string m_sSMTPPass*: password for the SMTP account.
- *std::string m_sSMTPAdress*: address of the SMTP server.
- *int m_nSMTPPort*: port for the SMTP protocol in the target server.

Student communication component

- **Method Name:** `mailbox::mailbox(GlobalStorage& globalData, UnsentEmailsStorage& unsentData, CourseDataStorage& courseData)`
- **Parameters:**
  - globalData. A reference to a globalStorage object.
  - unsentData. A reference to a UnsentEmailsStorage object.
  - courseData. A reference to a CourseDataStorage object.
- **Description:** Initializes a new mailbox object.
- **Data structure and tables it accesses:** This method does not access any data from the storage references.
- **Pre-conditions:** None
- **Post-conditions:** The mailbox object is initialized.
- **Called by:** Main application.
- **Calls:** None

- **Method Name:** `returnCode mailbox::downloadNew()`
- **Return Value:** One of the following:
  - NOERROR The method succeeded but no assignments was imported
  - MAIL_NEWASSIGNMENT The method succeeded and assignments was imported
  - MAIL_SMTPUNAVAIL The SMTP server was unavailable
  - MAIL_SMTPREFUSE The SMTP sever refused connection
  - MAIL_POPUNAVAIL The POP3 server was unavailable
  - MAIL_POPREFUSE The POP3 sever refused connection
  - MAIL_STORAGEERROR An error occurred while storing assignments, possible loss of data
- **Description:** Imports assignments from e-mail and sends confirmation e-mails. This method is the implementation of user requirement: 2.1.1, 2.1.2, 2.2.2, 4.1.1 and 4.1.2
- **Data structure and tables it accesses:**
  - SMTP account information from globalData.
  - POP3 account information from courseData
  - AssignmentInfoData from courseData
  - StudentData from courseData
  - May store data in unsentData
- **Pre-conditions:** Accurate data loaded into storage objects
- **Post-conditions:** New assignments are downloaded and stored into courseData. Confirmation e-mails are sent or stored in unsentData

- **Called by:** Main application.
- **Calls:** Functions in the storage objects.

- **Method Name:** `returnCode mailbox::sendStored()`
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - MAIL_SMTPUNAVAIL The SMTP server was unavailable
  - MAIL_SMTPREFUSE The SMTP sever refused connection
  - MAIL_STORAGEERROR An error occurred while accessing unsentData
- **Description:** Sends stored e-mails. This method is required for implementation of user requirements: 4.1.1, 4.1.2 and 4.2.1.
- **Data structure and tables it accesses:**
  - SMTP account information from globalData.
  - E-mail data in unsentData
- **Pre-conditions:** Accurate data loaded into storage objects
- **Post-conditions:** Unsent e-mails are sent or stored in unsentData
- **Called by:** Main application.
- **Calls:** Functions in the storage objects.

- **Method Name:** `returnCode mailbox::sendAll()`
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - MAIL_SMTPUNAVAIL The SMTP server was unavailable
  - MAIL_SMTPREFUSE The SMTP sever refused connection
  - MAIL_STORAGEERROR An error occurred while accessing courseData or unsentData
- **Description:** Sends feedback, composed of a comment and a grade, to all students on modified assignments. This method is part of the implementation of user requirement. 4.2.1.
- **Data structure and tables it accesses:**
  - SMTP account information from globalData
  - StudentData from courseData
  - May store data in unsentData
- **Pre-conditions:** Accurate data loaded into storage objects.
- **Post-conditions:** Feedback e-mails are composed and sent or stored in unsentData
- **Called by:** Main application.
- **Calls:** mailbox::sendFeedback(int sID, int aID) Functions in the storage objects.

- **Method Name:** `returnCode mailbox::sendFeedback(int sID, int aID)`
- **Parameters:**
  - sID. Integer to identify a student in courseData.
  - aID Integer to identify the students assignment.
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - MAIL_SMTPUNAVAIL The SMTP server was unavailable
  - MAIL_SMTPREFUSE The SMTP sever refused connection
  - MAIL_STORAGEERROR An error occurred while accessing courseData or unsentData
- **Description:** Sends feedback, composed of a comment and a grade, to a student on a specified assignment. This method is the implementation of user requirements: 4.2.1 and 4.2.2
- **Data structure and tables it accesses:**
  - SMTP account information from globalData.
  - StudentData from courseData
  - May store data in unsentData
- **Pre-conditions:** Accurate data loaded into storage objects.
- **Post-conditions:** A feedback e-mail is composed and sent or stored in unsentData
- **Called by:** Main application and mailbox
- **Calls:** Functions in the storage objects.

- **Method Name:** `void mail::setAdress(wstring address)`
- **Parameters:** address. A string containing an e-mail address.
- **Description:** Sets the address in a mail object.
- **Pre-conditions:** None
- **Post-conditions:** The mail objects address is changed.
- **Called by:**

- **Method Name:** `void mail::setSubject(wstring subject)`
- **Parameters:** address. A string containing an e-mail subject.
- **Description:** Sets the subject in a mail object.
- **Pre-conditions:** None
- **Post-conditions:** The mail objects subject is changed.
- **Called by:**

- **Method Name:** `void mail::setMessage(wstring message)`
- **Parameters:** address. A string containing an e-mail message.
- **Description:** Sets the message in a mail object.
- **Pre-conditions:** None
- **Post-conditions:** The mail objects message is changed.
- **Called by:**

- **Method Name:** `void mail::addAttachment(wstring filename, char* data, int size)`
- **Parameters:**
  - filename. The name of the attached file
  - data. The data in the file.
  - size The size of data in bytes
- **Description:** Adds a attachment to a mail object.
- **Pre-conditions:** data points to allocated memory of the size of the parameter size
- **Post-conditions:** An attachment object is added to the mail objects attachments.
- **Called by:**

- **Method Name:** `wstring mail::getAdress()`
- **Return Value:** A string containing the address of a mail object. If the mail objects address is uninitialized the empty string is returned.
- **Description:** Retrieves the address of a mail object.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**

- **Method Name:** `wstring mail::getSubject()`
- **Return Value:** A string containing the subject of a mail object. If the mail objects subject is uninitialized the empty string is returned.
- **Description:** Retrieves the subject of a mail object.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**

- **Method Name:** `wstring mail::getMessage()`
- **Return Value:** A string containing the message of a mail object. If the mail objects message is uninitialized the empty string is returned.
- **Description:** Retrieves the message of a mail object.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**

- **Method Name:** `int mail::countAttachment()`
- **Return Value:** A integer with the number of files attached to a mail object.
- **Description:** Retrieves the number of files attached to a mail object.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**

- **Method Name:** `attachment* mail::getAttachment(int idx)`
- **Parameters:** idx. The desired attachment index in the mail object.
- **Return Value:** A pointer to the attachment pointed to by the parameter idx. If there are no attachment with the given index zero is returned.
- **Description:** Retrieves attachments from a mail object.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**

- **Method Name:** `void mail::clear()`
- **Description:** Clears address, subject and message in a mail object. The attachments are deleted.
- **Pre-conditions:** None
- **Post-conditions:** address, subject and message are set to the empty string. Attachments are deleted.
- **Called by:**

- **Method Name:** `attachment::attachment(wstring filename, int size, char* data)`

- **Parameters:**
  - filename. The name of the attachment.
  - size. The size of the attachment in bytes.
  - data The attachment data.
- **Description:** Initializes an attachment object. The memory pointed to by data will be deallocated when the attachment object is destroyed.
- **Pre-conditions:** data points to allocated memory of the size of the parameter size
- **Post-conditions:** The attachment object is initialized
- **Called by:**
- **Calls:**

- **Method Name:** `wstring attachment::getFilename()`
- **Return Value:** A string containing the filename of an attachment.
- **Description:** Retrieves the filename of an attachment.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**
- **Calls:**

- **Method Name:** `int attachment::getSize()`
- **Return Value:** The size of the attachment data in bytes.
- **Description:** Retrieves the size of an attachment.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**
- **Calls:**

- **Method Name:** `char* attachment::getData()`
- **Return Value:** A pointer to attachment data.
- **Description:** Retrieves the data from an attachment.
- **Pre-conditions:** None
- **Post-conditions:** None
- **Called by:**

- **Calls:**

- **Method Name:** `validator::validator(CourseDataStorage& courseData)`
- **Parameters:** courseData. A reference to a CourseDataStorage object, used to validate e-mails.
- **Description:** Initializes a validator object.
- **Data structure and tables it accesses:** This method does not access any data in the courseData object.
- **Pre-conditions:** None.
- **Post-conditions:** The validator object is initialized.
- **Called by:** mailbox
- **Calls:**

- **Method Name:** `bool validator::validate(wstring subject)`
- **Parameters:** subject. A string containing a e-mail subject line, used to validate the e-mail.
- **Return Value:** true if the e-mail can be related to a student and an assignment in courseData. Otherwise false.
- **Description:** Validates e-mail headers. This method is the implementation of user requirement: 2.1.2, 2.1.3.
- **Data structure and tables it accesses:** courseData
- **Pre-conditions:** Accurate data loaded into storage objects.
- **Post-conditions:** None
- **Called by:** mailbox
- **Calls:** Functions in courseData.

- **Method Name:** `smtpTransport::smtpTransport(GlobalStorage& globalData)`
- **Parameters:** globalData. A reference to a GlobalStorage object, used to get account information.
- **Description:** Initializes a smtpTransport object.
- **Data structure and tables it accesses:** SMTP account information in globalData.
- **Pre-conditions:** Accurate data loaded into storage objects.
- **Post-conditions:** The smtpTransport object is initialized.
- **Called by:** mailbox
- **Calls:**

- **Method Name:** `returnCode smtpTransport::sendMessage(mail& message)`
- **Parameters:** message. A reference to a mail object.
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - MAIL_SMTPUNAVAIL The SMTP server was unavailable
  - MAIL_SMTPREFUSE The SMTP sever refused connection
- **Description:** Sends a e-mail message stored in a mail object.
- **Pre-conditions:** The mail object holds a e-mail.
- **Post-conditions:** If no error occurred the e-mail is sent.
- **Called by:** mailbox
- **Calls:** Functions in the e-mail library

- **Method Name:** `pop3Transport::pop3Transport(EmailAccountPOP3Data& pop3Data)`
- **Parameters:** pop3Data. A reference to a EmailAccountPOP3Data object, used to get account information.
- **Description:** Initializes a pop3Transport object.
- **Data structure and tables it accesses:** pop3 account information in pop3Data.
- **Pre-conditions:** Accurate data loaded into pop3Data.
- **Post-conditions:** The pop3Transport object is initialized.
- **Called by:** mailbox
- **Calls:**

- **Method Name:** `returnCode pop3Transport::readHeaders(vector<wstring>& headers)`
- **Parameters:** headers. A reference to a vector<wstring> object, used to store e-mail headers.
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - MAIL_POPUNAVAIL The pop3 server was unavailable
  - MAIL_POPREFUSE The pop3 sever refused connection
- **Description:** Reads headers on a pop3 server.
- **Pre-conditions:** The parameter is a empty vector.
- **Post-conditions:** Headers are downloaded from the server and added to the vector.
- **Called by:** mailbox
- **Calls:** Functions in the e-mail library.

- **Method Name:** `returnCode pop3Transport::download(pop3messageIdentifier messageId, mail& message)`
- **Parameters:**
  - message id. An identifier to identify a message on a pop3 server. The exact format for this is not decided yet
  - message. A reference to a mail object.
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - MAIL_POPUNAVAIL The pop3 server was unavailable
  - MAIL_POPREFUSE The pop3 sever refused connection
- **Description:** Downloads a specified message from a pop3 server
- **Pre-conditions:** None
- **Post-conditions:** The e-mail is downloaded and stored in message
- **Called by:** mailbox
- **Calls:** Functions in the e-mail library

External communication and RES communication components

- **Method Name:** `returnCode externalCom::exportGrades(int sysID)`
- **Parameters:**
  - sysID. The internal identifier for a defined external system.
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - EXTSYSCOMP_NOFOUND Impossible to find the specified external communication component
  - EXTSYS_ERR Impossible to communicate with the external system
- **Description:** Exports the grades of the assignments of the currently opened course to the specified external system. This method implements the user requirement 6.2.1
- **Data structure and tables it accesses:**
  - ExternalSystemsInfoData from GlobalData (to define)
- **Pre-conditions:** Accurate data loaded into storage objects
- **Post-conditions:** The grades for the existent assignments in the course are exported to the specified external system
- **Called by:** Main application.
- **Calls:** Functions in the storage objects, externalCom::createGradesExcFile(), execution of RESCom::importGrades()

- **Method Name:** `returnCode externalCom::importStudents(int sysID)`
- **Parameters:**
  - sysID. The internal identifier for a defined external system.
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - EXTSYSCOMP_NOFOUND Impossible to find the specified external communication component
  - EXTSYS_ERR Impossible to communicate with the external system
- **Description:** Imports the students from the specified external system into the currently opened course. This method implements the user requirement 6.1.1
- **Data structure and tables it accesses:**
  - ExternalSystemsInfoData from GlobalData (to define)
- **Pre-conditions:** Accurate data loaded into storage objects
- **Post-conditions:** The students from the specified external system are loaded into the currently opened course.

- **Called by:** Main application.
- **Calls:** Functions in the storage objects, externalCom::loadStudentsExcFile(), execution of RESCom::exportStudents(std::wstring m_sCourseCode)

- **Method Name:** `returnCode externalCom::createGradesExcFile()`
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - ERROR Impossible to create the grades exchange file
- **Description:** Creates the exchange grades file used by the specified external system communication component. This method will implement the user requirement 6.2.2
- **Data structure and tables it accesses:**
  - AssignmentInfoData from courseData
  - StudentData from courseData
  - AssignmentData from studentData
  - ExternalSystemsInfoData from GlobalData (to define)
- **Pre-conditions:** Accurate data loaded into storage objects
- **Post-conditions:** An exchange file with the grades for the existent assignments in the course is created.
- **Called by:** externalCom::exportGrades(int sysID)
- **Calls:** Functions in the storage objects.

- **Method Name:** `returnCode externalCom::loadStudentsExcFile()`
- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - ERROR Impossible to read the students exchange file
- **Description:** Imports the students from an exchange file into the currently opened course.
- **Data structure and tables it accesses:**
  - StudentData from courseData
  - CourseDataStorage
- **Pre-conditions:** Accurate data loaded into storage objects
- **Post-conditions:** The students from the exchange file are loaded into the currently opened course.
- **Called by:** externalCom::loadStudentsExcFile()
- **Calls:** Functions in the storage objects

NOTE: The two next methods belong to two independent single-class applications (scripts) that complement the AAMS system. In order to keep consistent with the earlier sections and the used notation, they appear as methods of an artificial class called *RESCom*.

- **Method Name:** `void RESCom::importGrades()`
- **Description:** Imports the grades from the exchange file into the RES system. This method implements the user requirement 6.2.1
- **Data structure and tables it accesses:**
  - RES system (file "res")
- **Pre-conditions:** The information in the exchange file (course code, etc.) is consistent with the information in the RES system.
- **Post-conditions:** The grades from the exchange file are imported to the RES system
- **Called by:** RES system importing script application.

- **Method Name:** `void RESCom::exportStudents(std::wstring m_sCourseCode)`
- **Parameters:**
  - m_sCourseCode. The code of the target course in the RES system
- **Description:** Exports a list with the students of the RES system course with code *m_sCourseCode*. This method implements the user requirement 6.1.1
- **Data structure and tables it accesses:**
  - RES system (file "res")
- **Pre-conditions:** The given code is a valid one in the RES system.
- **Post-conditions:** A students exchange file with the RES system students is created.
- **Called by:** RES system exporting script application.

GUI/Event handler component

- **Method Name:** `bool AAMSGUI::OnInit()`
- **Return Value:** One of the following:
  - AAMS_START The start is successful and the GUI can be displayed
  - AAMS_FAILURE Impossible to start the system
- **Description:** Initializes the main system objects and GUI elements and runs the system.
- **Data structure and tables it accesses:**
  - Defined GUI elements in the *GUIcreator* class.
- **Pre-conditions:** The operative system is ready and able to run the system
- **Post-conditions:** The system is initialized and the main window is shown
- **Called by:** The operative system ("main()" method).
- **Calls:** Initializing elements of the GUI library defined by the class *GUIcreator*.

The class *GUIcreator* contains the definition of the whole AAMS GUI. For each form, a wxWidgets class is defined. The controls inside these are defined as attributes, and each wxWidgets class contains a constructor method, where the static properties of every control are defined, and an event table in order to route the events to the *GUIcontroller* object. These methods are trivial and for clarity reasons they will not be listed here.

The class *GUIcontroller* contains a set of methods that handle all the user events, calling the proper methods of every component, displaying the corresponding forms, and managing the possible errors.

- **Method Name:** `void ErrorManager::handleUsrError(int errorCode)`
- **Parameters:**
  - errorCode. Identifier of the kind of error.
- **Description:** Handles possible errors, displaying explanations of them or finishing the system if they are critical.
- **Post-conditions:** The error is managed and an explanation of it is shown.
- **Called by:** GUIController.
- **Calls:** WindowManager::displayMsgBoxErr(wstring msg)

- **Method Name:** `returnCode FileManager::openExternalFile(wstring filePath)`
- **Parameters:**

  - filePath. Path where the file to be opened is.

- **Return Value:** One of the following:
  - NOERROR The method succeeded
  - UNEXISTENT_FILE The file could not be found
  - UNEXISTENT_APP There is no declared external application to open the file
- **Description:** Opens the given file with an external application.
- **Pre-conditions:** The operative system keeps a list of default applications for different file extensions.
- **Post-conditions:** The error is managed and an explanation of it is shown.
- **Called by:** GUIController.
- **Calls:** wxMimeTypesManager::GetFileTypeFromExtension(extension), wxFileType::GetOpenCommand(fileName)

- **Method Name:** `void WindowManager::displayMsgBoxErr(wstring msg)`
- **Parameters:**
  - msg. The message to be displayed
- **Description:** Shows a typical user error message dialog with the given text
- **Post-conditions:** The message box with the specified message is shown
- **Called by:** GUIcontroller.
- **Calls:** wxMessageDialog::wxMessageDialog(), wxMessageDialog::ShowModal()

- **Method Name:** `void WindowManager::showSecondaryWindow(int wCode)`
- **Parameters:**
  - wCode. Code that identifies the secondary window that will be opened
- **Description:** Shows a secondary window, disabling the main one at the same time
- **Pre-conditions:** There is no secondary window already opened
- **Post-conditions:** The secondary window is displayed and the main window is disabled
- **Called by:** GUIcontroller.
- **Calls:** Show the target window, disable the main one

- **Method Name:** `void WindowManager::hideSecondaryWindow()`
- **Description:** Hides the currently displayed secondary window, giving the focus back to the main one at the same time
- **Pre-conditions:** There is a secondary window already opened

- **Post-conditions:** The secondary window is hidden and the main window is enabled
- **Called by:** GUIcontroller.
- **Calls:** Show the main window, hide the secondary one

Data storage component

Class EncryptUtil

`+ void Encrypt(std::string &sData)`

`+ void Decrypt(std::string &sData)`

- **Method Name:** `void Encrypt(std::string &sData)`
- **Description:** The method encrypt the data in parameter sData.
- **Parameters:**
  - sData: The data to be encrypted.
- **Pre-condition:** The data in sData is not encrypted.
- **Post-condition:** The data in sData is encrypted.
- **Called by:** XMLUnsentEmailsDataWrite::WriteXMLUnsentEmailsData(), XMLGlobalDataWrite::WriteXMLGlobalData and XMLCourseDataWrite::WriteXMLCourseData

- **Method Name:** `void Decrypt(std::string &sData)`
- **Description:** The method decrypt the data in parameter sData.
- **Parameters:**
  - sData: The data to be decrypted.
- **Pre-condition:** The data in sData is encrypted.
- **Post-condition:** The data in sData is not encrypted.
- **Called by:** XMLUnsentEmailsDataRead::ReadXMLUnsentEmailsData(), XMLGlobalDataRead::ReadXMLGlobalData and XMLCourseDataRead::ReadXMLCourseData

Class UnsentEmailsStorage

`- int m_nEmailIdCounter`

`- std::list<UnsentEmailsData> m_unsentEmailsData`

`+ bool Load()`

`+ bool Save()`

`+ void AddEmail(const std::wstring sDestination, const std::wstring sHeader, const std::wstring sContent)`

+ `void RemoveEmail(int eid)`

+ `const std::list<UnsentEmailsData> & GetUnsentEmails()`

+ `int GetEmailCount()`

- **Method Name:** `bool Load()`
- **Description:** The method loads the unsent emails from a file.
- **Return value:** true if the load from file succeeded, otherwise false.
- **Post-condition:** The data in the file is now in loaded into std::list<UnsentEmailsData>
- **Called by:** GUIController
- **Calls:** XMLUnsentEmailsDataRead::ReadXMLUnsentEmailsData(), XMLUnsentEmailsDataRead::Open() and XMLUnsentEmailsDataRead::Close().

- **Method Name:** `bool Save()`
- **Description:** The method saves the unsent emails to a file.
- **Return value:** true if the save to file succeeded, otherwise false.
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:**
- **Called by:** GUIController
- **Calls:** XMLUnsentEmailsDataWrite::WriteXMLUnsentEmailsData(), XMLUnsentEmails-DataWrite::Open(), XMLUnsentEmailsDataWrite::Close()

- **Method Name:** `void AddEmail(const std::wstring sDestination, const std::wstring sHeader, const std::wstring sContent)`
- **Description:** The method add the email with the data from the parameters into the list of unsent emails
- **Parameters:**
  - sDestionation: Contain the destination e-mail.
  - sHeader: Contain the subject line of the e-mail.
  - sContent: Contain the content of the e-mail.
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The e-mail specified by the parameters exists in the list of unsent emails m_unsentEmailsData
- **Called by:**

- **Calls:** None

- **Method Name:** `void RemoveEmail(int eid)`
- **Description:** The method remove the unsent e-mail specified by the email identifier eid
- **Parameters:**
  - eid: An identifier for an unsent e-mail in the list of unsent e-mails m_unsentEmailsData.
- **Pre-condition:** None
- **Validity checks:**
- **Post-condition:** The e-mail with the identifier eid is no longer in the list of unsent e-mails m_unsentEmailsData
- **Called by:**
- **Calls:** None

- **Method Name:** `const std::list<UnsentEmailsData> & GetUnsentEmails()`
- **Description:** The method return the list of unsent e-mails, m_unsentEmailsData.
- **Return value:** The list of unsent e-mails, m_unsentEmailsData.
- **Pre-condition:** None
- **Validity checks:**
- **Post-condition:** The list of unsent e-mails m_unsentEmailsData remains intact.
- **Called by:**
- **Calls:** None

- **Method Name:** `int GetEmailCount()`
- **Description:** The method returns the number of unsent e-emails from the list of unsent e-mails m_unsentEmailsData.
- **Return value:** The number of unsent e-mails in the list of unsent e-mails.
- **Pre-condition:** None
- **Validity checks:**
- **Post-condition:** The list of unsent e-mails m_unsentEmailsData remains intact.
- **Called by:**
- **Calls:** None

Class UnsentEmailsData

- std::wstring m_sDestination

- std::wstring m_sHeader

- std::wstring m_sContent

Class XMLUnsentEmailsDataRead

+ bool Open()

+ void ReadXMLUnsentEmailsData(std::list<UnsentEmailsData> &unsentEmailsData, int &emailIdCounter)

+ void Close()

- TiXmlDocument m_doc

- **Method Name:** `bool Open()`
- **Description:** The method opens the XML file containing the unsent emails for reading.
- **Return value:** true if opening of file succeeded, otherwise false.
- **Pre-condition:** None
- **Validity checks:**
- **Post-condition:** Variable m_doc contain the XML file.
- **Called by:** UnsentEmailsStorage::Load()
- **Calls:** None

- **Method Name:** `void ReadXMLUnsentEmailsData(std::list<UnsentEmailsData> &unsentEmailsData, int &emailIdCounter)`
- **Description:** The method read the XML value
- **Parameters:**
  - unsentEmailsData: The list of unsent emails.
  - emailIdCounter: The identification counter.
- **Return value:** None
- **Pre-condition:** The XML file have been successfully opened.
- **Validity checks:** If the file is opened.
- **Post-condition:** the parameters contain the data that was in the XML file.
- **Called by:** UnsentEmailsStorage::Load()
- **Calls:** None

- **Method Name:** `void Close()`
- **Description:** The method will close the file if opened.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if file is opened.
- **Post-condition:** The file is closed.
- **Called by:** UnsentEmailsStorage::Load()
- **Calls:** None

Class XMLUnsentEmailsDataWrite

```
+ bool Open()

+ void WriteXMLUnsentEmailsData(const std::list<UnsentEmailsData> &unsentEmailsData,
  int emailIdCounter)

+ void Close()

- TiXmlDocument m_doc
```

- **Method Name:** `bool Open()`
- **Description:** The method will open the XML file for writing.
- **Return value:** true if the file is successfully opened, otherwise false.
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** None
- **Called by:** UnsentEmailsStorage::Save()
- **Calls:** None

- **Method Name:** `void WriteXMLUnsentEmailsData(const std::list<UnsentEmailsData> &unsentEmailsData, int emailIdCounter)`
- **Description:** The method will write the content of the parameters unsentEmailsData and emailIdCounter into the XML file that hold the unsent emails.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.

- **Post-condition:** The data is written to the file.
- **Called by:** UnsentEmailsStorage::Save()
- **Calls:** None

- **Method Name:** `void Close()`
- **Description:** The method close the file if it is opened.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** The file is closed
- **Called by:** UnsentEmailsStorage::Save()
- **Calls:** None

Class GlobalStorage

```
+ bool Load()

+ bool Save()

+ void SetSMTPUser(const std::string sUser)

+ void SetSMTPPass(const std::string sPass)

+ void SetSMTPAdress(const std::string sAdress)

+ void SetSMTPPort(const int nPort)

+ const GlobalData &GetGlobalData()

- GlobalData m_globalData
```

- **Method Name:** `bool Load()`
- **Description:** The method loads the global data from file
- **Return value:** true if the load from file succeeded otherwise false
- **Pre-condition:**
- **Validity checks:**
- **Post-condition:** The data from the file is loaded into m_globalData
- **Called by:** GUIController
- **Calls:** XMLGlobalDataRead::Open() , XMLGlobalDataRead::Close() and XMLGlobalDataRead::ReadXMLGlobalData()

- **Method Name:** `bool Save()`
- **Description:** The method saves the data in m_globalData to a file.
- **Return value:** None
- **Pre-condition:**
- **Validity checks:**
- **Post-condition:** The data in m_globalData is saved to a file.
- **Called by:** XMLGlobalDataWrite::Open() , XMLGlobalDataWrite::WriteXMLGlobalData() and XMLGlobalDataWrite::Close()
- **Calls:** None

- **Method Name:** `void SetSMTPUser(const std::string sUser)`
- **Description:** The method sets the user name for the SMTP server.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:**
- **Post-condition:** The m_sSMTPUser of m_globalData has the user name specified by parameter sUser.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetSMTPPass(const std::string sPass)`
- **Description:** The method set the password for the SMTP server.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The m_sSMTPPass field of m_globalData have the password specified by parameter sPass.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetSMTPAdress(const std::string sAdress)`
- **Description:** The method set the address for the SMTP server.
- **Return value:** None

- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The m_sSMTPAdress field of m_globalData have the address specified by parameter sAdress.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetSMTPPort(const int nPort)`
- **Description:** The method set the port for the SMTP server.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The m_nSMTPPort field of m_globalData have the port specified by parameter nPort.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `const GlobalData &GetGlobalData()`
- **Description:** The method return the m_globalData for reading.
- **Return value:** The data structure m_globalData
- **Pre-condition:** None
- **Validity checks:**
- **Post-condition:** The data of m_globalData remains intact.
- **Called by:** Different components
- **Calls:** None

Class GlobalData

```
- std::string m_sSMTPUser

- std::string m_sSMTPPass

- std::string m_sSMTPAdress

- int m_nSMTPPort
```

Class XMLGlobalDataRead

`+ bool Open()`

`+ void ReadXMLGlobalData(GlobalData &globalData)`

`+ void Close()`

`+ TiXmlDocument m_doc`

- **Method Name:** `bool Open()`
- **Description:** The method opens the file for reading.
- **Return value:** true if file was successfully opened.
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** None
- **Called by:** GlobalStorage::Load()
- **Calls:** None

- **Method Name:** `void ReadXMLGlobalData(GlobalData &globalData)`
- **Description:** The method reads the file and stores the information from the file in globalData parameter.
- **Parameters:**
  - globalData: holds the global data.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** The data from the file is in the parameter globalData.
- **Called by:** GlobalStorage::Load()
- **Calls:** None

- **Method Name:** `void Close()`
- **Description:** The method closes the file if it is opened.
- **Return value:** None
- **Pre-condition:** None

- **Validity checks:** Will check if the file is opened
- **Post-condition:** The file is closed.
- **Called by:** GlobalStorage::Load()
- **Calls:** None

Class XMLGlobalDataWrite

```
+ bool Open()
+ void WriteXMLGlobalData(const GlobalData &globalData)
+ void Close()
- TiXmlDocument m_doc
```

- **Method Name:** `bool Open()`
- **Description:** The method will open the file for writing.
- **Return value:** true if the file is opened successfully, otherwise false
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** None
- **Called by:** GlobalStorage::Save()
- **Calls:** None

- **Method Name:** `void WriteXMLGlobalData(const GlobalData &globalData)`
- **Description:** The method will save the data in globalData to a file.
- **Parameters:**
  - globalData: contains the data to be written to file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** None
- **Called by:** GlobalStorage::Save()
- **Calls:** None

- **Method Name:** `void Close()`
- **Description:** The method will close the file if it is opened.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** The file is closed.
- **Called by:** GlobalStorage::Save()
- **Calls:** None

Class CourseDataStorage

```
+ bool LoadCourse(const string sName)

+ bool SaveCourse()

+ void NewCourse(const std::string sName)

+ void SetCourseName(const std::wstring sCourseName)

+ void SetCourseCode(const std::wstring sCourseCode)

+ void SetStudentName(const int sid, const std::wstring sName)

+ void SetStudentExternalId(const int sid, const std::wstring sExternalId)

+ void SetStudentEmail(const int sid, const std::wstring sEmail)

+ void SetStudentGrade(const int sid, const std::wstring sGrade)

+ void SetAssignmentName(const int aid, const std::wstring sName)

+ void SetAssignmentExternalId(const int aid, const std::string sName)

+ void SetAssignmentEmailHeader(const int aid, const std::wstring sEmailHeader)

+ void SetAssignmentGrade(const int sid, const int aid, const std::wstring sGrade)

+ void SetAssignmentComment(const int sid, const int aid, const std::wstring sComment)

+ void SetAssignmentModified(const int sid, const int aid)

+ void SetFileAssignment(const std::string sStudentEmail, const std::wstring sEmailHeader,
  const std::wstring sFilename)

+ void SetFileComment(const int sid, const int aid, const int fid, const std::wstring
  sComment)

+ void SetFileFilename(const int sid, const int aid, const int fid, const std::wstring
  sFileName)

+ void RemoveStudent(const int sid)

+ void RemoveAssignment(const int aid)
+ void RemoveFile(const int fid, const int aid)
+ void AddNewStudent(const std::wstring sName, const std::wstring sExternalId, const
  std::wstring sEmail)
+ void AddNewAssignment(const std::wstring sAssignmentName, const std::wstring sExternalId,
  const std::wstring sEmailHeader)
+ void AddNewFile(const int aid, const std::wstring &sExtension)
+ const CourseData &GetCourseData()
+ const EmailAccountPOP3Data &GetEmailPOP3()
- CourseData m_courseData
- int m_nStudentIdCounter
- int m_nAssignmentIdCounter
- int m_nFileIdCounter
- EmailAccountPOP3Data m_POP3
```

- **Method Name:** `bool LoadCourse(const string sName)`
- **Description:** The method will load a course with the name specified by the parameter.
- **Return value:** true if the loading of the course was successful.
- **Pre-condition:** None
- **Validity checks:** Will check if the load was successful.
- **Post-condition:** None
- **Called by:** GUIController
- **Calls:** XMLCourseDataRead::Open() , XMLCourseDataRead::ReadXMLCourseData() and XMLCourseDataRead::Close()

- **Method Name:** `bool SaveCourse()`
- **Description:** The method will save the course data to file.
- **Return value:** true if saving was successful, otherwise false.
- **Pre-condition:** None
- **Validity checks:** Will check if the saving of the course was successful.
- **Post-condition:** None
- **Called by:** GUIController
- **Calls:** XMLCourseDataWrite::Open() , XMLCourseDataWrite::WriteXMLCourseData() and XMLCourseDataWrite::Close()

- **Method Name:** `void NewCourse(const std::string sName)`
- **Description:** The method will create a new course.
- **Parameters:**
  - sName: the name of the course.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the course already exists.
- **Post-condition:** The course with the name is created.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetCourseName(const std::wstring sCourseName)`
- **Parameters:**
  - sCourseName: the name of an existing course.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetCourseCode(const std::wstring sCourseCode)`
- **Parameters:**
  - sCourseCode: A code of the course.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetStudentName(const int sid, const std::wstring sName)`

- **Parameters:**
  - sid: student identifier
  - sName: the name of the student with id sid.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetStudentExternalId(const int sid, const std::wstring sExternalId)`
- **Parameters:**
  - sid: student Identifier
  - sExternalId: the external identifier of a student.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetStudentEmail(const int sid, const std::wstring sEmail)`
- **Parameters:**
  - sid: student identifier
  - sEmail: the email of the student.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetStudentGrade(const int sid, const std::wstring sGrade)`

- **Parameters:**
  - sid: student identifier
  - sGrade: the grade of the student.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetAssignmentName(const int aid, const std::wstring sName)`
- **Parameters:**
  - aid: Assignment identifier.
  - sName: The name of the assignment.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls: None**

- void SetAssignmentExternalId(const int aid, const std::string sName)
- **Parameters:**
  - aid: Assignment identifier.
  - sName: The name of the assignment.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetAssignmentEmailHeader(const int aid, const std::wstring sEmailHeader)`

- **Parameters:**
  - aid: Assignment identifier.
  - sEmailHeader: The email header of the assignment.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetAssignmentGrade(const int sid, const int aid, const std::wstring sGrade)`
- **Parameters:**
  - sid: StudentIdentifier.
  - aid: Assignment identifier.
  - sGrade: The grade of an assignment for the student.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetAssignmentComment(const int sid, const int aid, const std::wstring sComment)`
- **Parameters:**
  - sid: Student identifier.
  - aid: Assignment identifier.
  - sComment: The comment of the assignment for the student.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController

- **Calls:** None

- **Method Name:** `void SetAssignmentModified(const int sid, const int aid)`
- **Parameters:**
  - sid: Student identifier.
  - aid: Assignment identifier.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetFileAssignment(const std::string sStudentEmail, const std::wstring sEmailHeader, const std::wstring sFilename)`
- **Parameters:**
  - sStudentEmail: The email of the student
  - sEmailHeader: The header of the email.
  - sFilename: The name of the file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetFileComment(const int sid, const int aid, const int fid, const std::wstring sComment)`
- **Parameters:**
  - sid: Student identifier
  - fid: File identifier
  - sComment: The comment of the file.
- **Return value:** None
- **Pre-condition:** None

- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void SetFileFilename(const int sid, const int aid, const int fid, const std::wstring sFileName)`
- **Parameters:**
  - sid: Student identifier
  - fid: File identifier
  - sFilename: The name of the file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData is updated with the data from the parameter.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void RemoveStudent(const int sid)`
- **Parameters:**
  - sid: Student identifier.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The student with id from parameter sid is no longer inside m_courseData
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void RemoveAssignment(const int aid)`
- **Parameters:**
  - aid: Assignment identifier
- **Return value:** None
- **Pre-condition:** None

- **Validity checks:** None
- **Post-condition:** The assignment with id from parameter aid is no longer inside m_courseData
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void RemoveFile(const int fid, const int aid)`
- **Parameters:**
  - fid: File identifier
  - aid: Assignment identifier
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The file with id from parameter fid is no longer inside m_courseData.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void AddNewStudent(const std::wstring sName, const std::wstring sExternalId, const std::wstring sEmail)`
- **Parameters:**
  - sName: The name of the student.
  - sExternalId: The external identifier of the student.
  - sEmail: The email of the student.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The student with information from the parameters is now in m_courseData.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void AddNewAssignment(const std::wstring sAssignmentName, const std::wstring sExternalId, const std::wstring sEmailHeader)`
- **Parameters:**
  - sAssignmentName: Name of the assignment.
  - sExternalId: External identifier of the assignment.

  - sEmailHeader: The header (subject) of the email.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The assignment with information from parameters is now in m_courseData.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `void AddNewFile(const int aid, const std::wstring &sExtension)`
- **Parameters:**
  - aid: Assignment identifier
  - sExtension: Extension of the file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The file with information from parameters is now in m_courseData.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `const CourseData &GetCourseData()`
- **Return value:** A reference to m_courseData
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_courseData remains intact.
- **Called by:** Different components
- **Calls:** None

- **Method Name:** `const EmailAccountPOP3Data &GetEmailPOP3()`
- **Return value:** A reference to m_POP3.
- **Pre-condition:** None
- **Validity checks:** None
- **Post-condition:** The data of m_POP3 remains intact.

- **Called by:** Different components
- **Calls:** None

Class XMLCourseDataRead

+ `bool Open()`
+ `void ReadXMLCourseData(CourseData &courseData, EmailAccountPOP3Data &emailPOP3, EmailAccountSMTPData &emailSMTP, int &nStudentIdCounter, int &nAssignmentIdCounter, int &nFileIdCounter)`
+ `void Close()`

- `TiXmlDocument m_doc`

- **Method Name:** `bool Open()`
- **Description:** The method will open the file for reading.
- **Return value:** true if the opening of the file was successful
- **Pre-condition:** None
- **Validity checks:** Will check if the open of file was successful
- **Post-condition:** None
- **Called by:** CourseDataStorage::LoadCourse()
- **Calls:** None

- **Method Name:** `void ReadXMLCourseData(CourseData &courseData, EmailAccountPOP3Data &emailPOP3, EmailAccountSMTPData &emailSMTP, int &nStudentIdCounter, int &nAssignmentIdCounter, int &nFileIdCounter)`
- **Description:** The method will read the file and store the data in the parameters.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** None
- **Called by:** CourseDataStorage::LoadCourse()
- **Calls:** None

- **Method Name:** `void Close()`

- **Description:** The method will close the file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** The file is closed.
- **Called by:** CourseDataStorage::LoadCourse()
- **Calls:** None

Class XMLCourseDataWrite

```
+ bool Open()
+ void WriteXMLCourseData(const CourseData &courseData, const EmailAccountPOP3Data
  &emailPOP3, const EmailAccountSMTPData &emailSMTP, const int nStudentIdCounter,
  const int nAssignmentIdCounter, const int nFileIdCounter)
+ void Close()
```

- **Method Name:** `bool Open()`
- **Description:** The method will open the file for writing.
- **Return value:** true if the opening of the file was successful.
- **Pre-condition:** None
- **Validity checks:** Will check if the file was opened.
- **Post-condition:** None
- **Called by:** CourseDataStorage::SaveCourse()
- **Calls:** None

- **Method Name:** `void WriteXMLCourseData(const CourseData &courseData, const EmailAccountPOP3Data &emailPOP3, const EmailAccountSMTPData &emailSMTP, const int nStudentIdCounter, const int nAssignmentIdCounter, const int nFileIdCounter)`
- **Description:** The method will write the content of the parameters to a file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened.
- **Post-condition:** None

- **Called by:** CourseDataStorage::SaveCourse()
- **Calls:** None

- **Method Name:** `void Close()`
- **Description:** The method will close the file.
- **Return value:** None
- **Pre-condition:** None
- **Validity checks:** Will check if the file is opened
- **Post-condition:** The file is closed.
- **Called by:** CourseDataStorage::SaveCourse()
- **Calls:** None

Class CourseData

`- std::wstring m_sCourseName`

`- std::wstring m_sCourseCode`

`- std::list<StudentData> m_StudentData`

`- std::list<AssignmentInfoData> m_assignmentInfoData`

Class StudentData

`- int m_sid`

`- std::wstring m_sName`

`- std::wstring m_sExternalId`

`- std::wstring m_sEmail`

`- std::wstring m_sGrade`

`- std::list<AssignmentData> m_assignmentData`

Class AssignmentData

`- int m_aid`

- std::wstring m_sComment
- std::wstring m_sGrade
- bool m_bModified
- std::list<FileData> m_fileData

Class FileData

- int m_fid
- std::wstring m_sFilename

Class AssignmentInfoData

- int m_aid
- std::wstring m_sName
- std::wstring m_sExternalId
- std::wstring m_sEmailHeader
- int m_nFileCount
- std::list<FileInfoData> m_fileInfoData

Class FileInfoData

- int m_fid
- std::wstring m_sExtension

Class EmailAccountPOP3Data

- std::string m_sUser
- std::string m_sPass
- std::string m_sAdress
- int m_nPort

Report generator component

- **Method Name:** `GenerateType::getCourseData(std::wstring sCourseName)`
- **Return Value:** One of the following:
  - TRUE The method successfully executed
  - FALSE The method was unsuccessfully executed
- **Description:** Fetch data for a specific course.
- **Data structure and tables it accesses:**
  - course data from courseData
  - Student data from courseData
  - Assignment data from courseData
- **Pre-conditions:** Course identifier
- **Post-conditions:** Relevant data to create a course specific report.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `GenerateType::getStudentData(std::wstring sStudentName)`
- **Return Value:** One of the following:
  - TRUE The method successfully executed
  - FALSE The method was unsuccessfully executed
- **Description:** Fetch data for a specific student.
- **Data structure and tables it accesses:**
  - course data from courseData
  - Student data from courseData
- **Pre-conditions:** Student identifier
- **Post-conditions:** Relevant data to create a student specific report.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `GenerateType::getStudentAssignmentData(std::wstring sAssName)`
- **Return Value:** One of the following:
  - TRUE The method successfully executed
  - FALSE The method was unsuccessfully executed
- **Description:** Fetch data for a specific assignment.

- **Data structure and tables it accesses:**
  - course data from courseData
  - Assignment data from courseData
- **Pre-conditions:** Assignment identifier
- **Post-conditions:** Relevant data to create a student specific report.
- **Called by:** GUIController
- **Calls:** None

- **Method Name:** `CreatePdf::mCourseReport()`
- **Return Value:** One of the following:
  - TRUE The method successfully executed
  - FALSE The method was unsuccessfully executed
- **Description:** Generate a report based on course data. Reference to use case 5.1
- **Data structure and tables it accesses:**
  - course data from courseData
  - Student data from courseData
  - Assignment data from courseData
- **Pre-conditions:** Data to be formatted
- **Post-conditions:** Relevant data to create a course specific report.
- **Called by:** GUIController
- **Calls:** Haru Free PDF Library

- **Method Name:** `CreatePdf::mStudentReport()`
- **Return Value:** One of the following:
  - TRUE The method successfully executed
  - FALSE The method was unsuccessfully executed
- **Description:** Generate a report based on student data. Reference to use case 5.2
- **Data structure and tables it accesses:**
  - course data from courseData
  - Student data from courseData
  - Assignment data from courseData
- **Pre-conditions:** Data to be formatted
- **Post-conditions:** Relevant data to create a student specific report.
- **Called by:** GUIController

- **Calls:** Haru Free PDF Library

- **Method Name:** `CreatePdf::mAssReport()`
- **Return Value:** One of the following:
  - TRUE The method successfully executed
  - FALSE The method was unsuccessfully executed
- **Description:** Generate a report based on student data. Reference to use case 5.3
- **Data structure and tables it accesses:**
  - course data from courseData
  - Student data from courseData
  - Assignment data from courseData
- **Pre-conditions:** Data to be formatted
- **Post-conditions:** Data to create an assignment specific report.
- **Called by:** GUIController
- **Calls:** Haru Free PDF Library

5.6. Package Diagram

The data storage package controls the information flow within AAMS, hence most of the other classes are dependent of this package. Student communication packages are only needed to retrieve new data and not to process it.

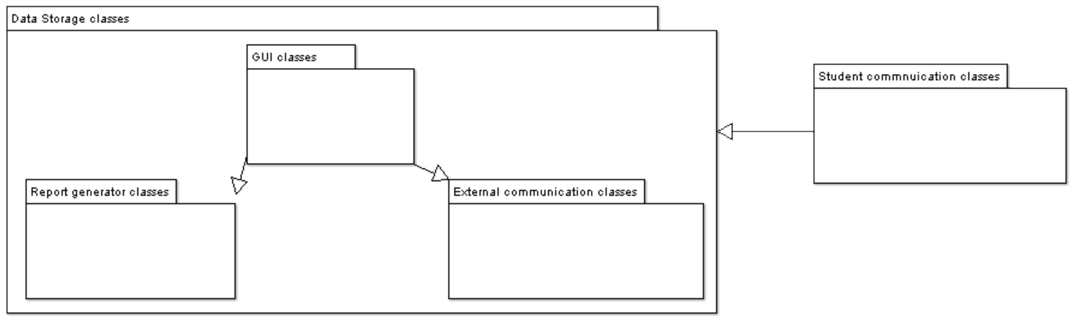

Figure 6: Package diagram of the system

References

[1] Sommerville, I., *Software Engineering*, Addison-Wesley. Eighth edition (2007)

[2] Group 21, *Requirements Document*, 2007