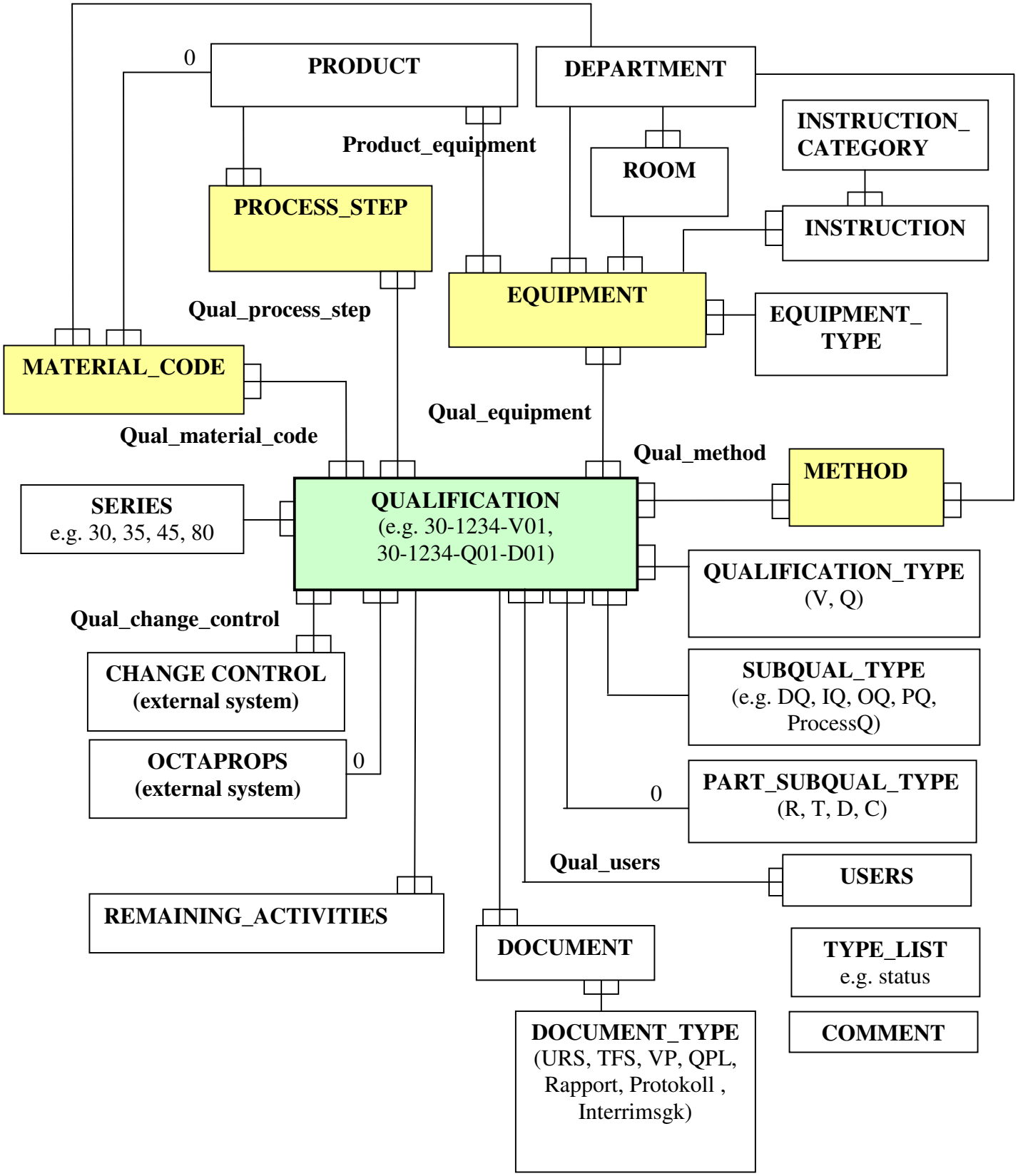
DATA MODEL



2007-03-19, 09:27

TYPE_LIST

Class	Value	COMMENT_ID	Timestamp	Userstamp
VARCHAR(40)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

Status – CURRENT, MODIFIED, ...
Qual_status – APPROVED, CANCELED,...
Requal_interval – 1 MONTH, 3 MONTHS, 1 YEAR, ...
Condition – OBSOLETE, REQUAL, APPROVED
Other config values (e.g. Root - Electronic documents root directory)

USERS

USER_ID	User_name	First_name	Last_name	Access_level	Last_logon
INT	VARCHAR(20)	VARCHAR(100)	VARCHAR(100)	VARCHAR(40)	DATETIME

COMMENT_ID	Timestamp	Userstamp
INT	DATETIME	VARCHAR(20)

Access_level: admin, create, write, read, locked

grey	primary or composite key
-----	foreign key

QUAL_USERS

QUALIFICATION_ID	USER_ID	COMMENT_ID
INT	INT	INT

COMMENT

COMMENT_ID	COMMENT_VERSION	Status	Text	Timestamp	Userstamp
INT	INT	VARCHAR(40)	VARCHAR(1000)	DATETIME	VARCHAR(20)

PRODUCT

PRODUCT_ID	Product_name	Requal_interval	Condition	COMMENT_ID	Timestamp	Userstamp
INT	VARCHAR(100)	VARCHAR(100)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

PRODUCT_EQUIPMENT

PRODUCT_ID	EQUIPMENT_ID	EQUIP_VERSION	COMMENT_ID	Timestamp	Userstamp
INT	INT	INT	INT	DATETIME	VARCHAR(20)

grey	primary or composite key
-----	foreign key

EQUIPMENT

EQUIPMENT_ID	EQUIP_VERSION	EQUIP_TYPE_ID	ROOM_ID	Equipment_name	Requal_interval
INT	INT	INT	INT	VARCHAR(100)	VARCHAR(100)

Status	Qual_condition	DEPARTMENT_ID	COMMENT_ID	Timestamp	Userstamp
VARCHAR(40)	VARCHAR(40)	INT	INT	DATETIME	VARCHAR(20)

Equipment name – M-nr

EQUIPMENT_TYPE

EQUIP_TYPE_ID	Name	Status	Timestamp	Userstamp
INT	VARCHAR(100)	VARCHAR(40)	DATETIME	VARCHAR(20)

PROCESS_STEP

PROCESS_STEP_ID	PRODUCT_ID	Step_name	Requal_interval	Qual_condition	COMMENT_ID
INT	INT	VARCHAR(100)	VARCHAR(100)	VARCHAR(40)	INT

grey	primary or composite key
-----	foreign key

Timestamp	Userstamp
DATETIME	VARCHAR(20)

Sterile prod has same process step for several products, protein purification has unique steps per product, i.e. step name not unique

ROOM

ROOM_ID	DEPARTMENT_ID	Room_name	Status	Timestamp	Userstamp
INT	INT	VARCHAR(100)	VARCHAR(40)	DATETIME	VARCHAR(20)

DEPARTMENT

DEPARTMENT_ID	Department_name	Timestamp	Userstamp
INT	VARCHAR(100)	DATETIME	VARCHAR(20)

INSTRUCTION

INSTRUCTION_ID	CATEGORY_ID	EQUIPMENT_ID	EQUIP_VERSION	Instruction_number	COMMENT_ID	Status
INT	INT	INT	INT	VARCHAR(100)	INT	VARCHAR(40)

grey	primary or composite key
-----	foreign key

Timestamp	Userstamp
DATETIME	VARCHAR(20)

INSTRUCTION_CATEGORY

CATEGORY_ID	Category_name	Timestamp	Userstamp
INT	VARCHAR(100)	DATETIME	VARCHAR(20)

e.g. operating instruction, preventive maintenance instruction, re-qualification instruction

MATERIAL_CODE

MATERIAL_CODE_ID	MTRLCODE_VERSION	PRODUCT_ID	DEPARTMENT_ID	Material_code	Name
INT	INT	INT	INT	VARCHAR(60)	VARCHAR(100)

Requal_interval	Qual_condition	COMMENT_ID	Timestamp	Userstamp
VARCHAR(100)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

Department e.g. for starting materials, media, products
Product may be 0 for e.g. starting material

grey	primary or composite key
-----	foreign key

METHOD

METHOD_ID	Method_number	Edition_number	Method_name	Requal_interval	COMMENT_ID	DEPARTMENT_ID
INT	VARCHAR(100)	VARCHAR(40)	VARCHAR(255)	VARCHAR(100)	INT	INT

Condition	Timestamp	Userstamp
VARCHAR(40)	DATETIME	VARCHAR(20)

SERIES

SERIES_ID	Series_number	Status	COMMENT_ID	Timestamp	Userstamp
INT	INT	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

30, 35, 45, 80...

QUALIFICATION_TYPE

QUAL_TYPE_ID	Qualification_type	Status	COMMENT_ID	Timestamp	Userstamp
INT	VARCHAR(100)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

V - validering, K - kvalificering

grey	primary or composite key
-----	foreign key

SUBQUAL_TYPE

SUBQUAL_TYPE_ID	Subqual_type	Status	COMMENT_ID	Timestamp	Userstamp
INT	VARCHAR(100)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

DQ, IQ, OQ, PQ, ProcessQ

PARTIAL_QUAL_TYPE

PART_QUAL_TYPE_ID	Partial_qualification_type	Status	COMMENT_ID	Timestamp	Userstamp
INT	VARCHAR(100)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

D – delkvalificering; C – “commissioning”, “qualification light”, R - rekvalificering, T – tillägg; H, P, B. F, A, C, D, V, W, S, K “dokbokstav”

QUALIFICATION

QUALIFICATION_ID	SERIES_ID	Qual_seq	QUAL_TYPE_ID	Subqual_seq	SUBQUAL_TYPE_ID
INT	INT	INT	INT	INT	INT

PART_QUAL_TYPE_ID	Part_qual_seq	Qualification_title	Qualification_leader	PROJECT_ID
INT	INT	VARCHAR(255)	VARCHAR(255)	VARCHAR(100)

grey	primary or composite key
-----	foreign key

Start_date	Approval_date	End_date	Qual_condition	Status	COMMENT_ID	Timestamp	Userstamp
DATETIME	DATETIME	DATETIME	VARCHAR(40)	VARCHAR(40)	INT	DATETIME	CHAR(20)

QUAL_CHANGE_CONTROL

QUALIFICATION_ID	CHANGE_CTRL_ID	COMMENT_ID	Timestamp	Userstamp
INT	VARCHAR(100)	INT	DATETIME	VARCHAR(20)

CC eg- “2006-137”

REMAINING_ACTIVITIES

ACTIVITY_ID	QUALIFICATION_ID	Activity	Responsible	Status	Date_completed	Timestamp	Userstamp
INT	INT	VARCHAR(500)	VARCHAR(200)	VARCHAR(40)	DATETIME	DATETIME	VARCHAR(20)

QUAL_PROCESS_STEP

QUALIFICATION_ID	PROCESS_STEP_ID	COMMENT_ID	Timestamp	Userstamp
INT	INT	INT	DATETIME	VARCHAR(20)

grey	primary or composite key
-----	foreign key

QUAL_MATERIAL_CODE

QUALIFICATION_ID	MATERIAL_CODE_ID	MTRLCODE_VERSION	COMMENT_ID	Timestamp	Userstamp
INT	INT	INT	INT	DATETIME	VARCHAR(20)

QUAL_EQUIPMENT

QUALIFICATION_ID	EQUIPMENT_ID	EQUIP_VERSION	COMMENT_ID	Timestamp	Userstamp
INT	INT	INT	INT	DATETIME	VARCHAR(20)

QUAL_METHOD

QUALIFICATION_ID	METHOD_ID	COMMENT_ID	Timestamp	Userstamp
INT	INT	INT	DATETIME	VARCHAR(20)

DOCUMENT_TYPE

DOC_TYPE_ID	Document_type	Status	COMMENT_ID	Timestamp	Userstamp
INT	VARCHAR(100)	VARCHAR(40)	INT	DATETIME	VARCHAR(20)

URS, TS, FS, TFS, VP, QPL, Rapport, Protokoll, Interrimsgk

grey	primary or composite key
-----	foreign key

DOCUMENT

DOCUMENT_ID	DOC_VERSION	QUALIFICATION_ID	DOC_TYPE_ID	Document_title	Author
INT	INT	INT	INT	VARCHAR(200)	VARCHAR(100)

Approval_date	Archive_location	Archive_ID	Electronic_storage_path	COMMENT_ID
DATETIME	VARCHAR(300)	VARCHAR(100)	VARCHAR(300)	INT

Status	Timestamp	Userstamp
VARCHAR(40)	DATETIME	VARCHAR(20)

grey	primary or composite key
-----	foreign key

```
/*
 * storeType stores of modifies a new class – value pair used to configure the system (e.g. class “status”, value “approved”)
 * in the table type_list. Any given comment is stored in the table comment.
 * Checks for the presence of class-value pair, updates if found, inserts if not present
 * Post-condition: Class – value pair has been stored or modified
 * Called by business layer database connection object
 *
 */
void storeType ( String   class          // category
                , String   value         // new value for category
                , String   comment       // additional information
                , String   user          // user performing operation
                )


/*
 * removeType removes a given class – value pair used to configure the system from the table type_list.
 * Checks if the value has been used in database instance tables, if so, does not remove and throws exception
 * Post-condition: Class – value pair has been removed if unused in other database columns
 * Called by business layer database connection object
 *
 */
void removeType ( String   class                // category
                , String   value         // value to remove
                ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * getTypeValues returns the given values for a given class from the table type_list. If class is null, values for all classes
 * are returned. Returns null if the given class is not present.
 * Post-condition: Class/value pairs have been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getTypeValues (String   class)              // type class to return values for



/*
 * storeUser stores a user with the given username, first and last name, and access level in the table users. If the user
 * already exists, first and last name, access level and last logon are updated if not null. Any given comment is stored in the table comment.
 * Checks for presence of user name and performs insert or update accordingly
 * Pre-condition: user name is unique in database
 * Post-condition: user entry in table users has been stored or changed,
 *                      comment has been saved in table comment and linked if not null
 * Called by business layer database connection object
 *
 */
void storeUser (String   user_name             // username
                , String   first_name          // user first name
                , String   last_name           // user last name
                , String   access_level        // user access level
                , Date     last_logon          // last logon
                , String   comment             // additional information
                  )
```

Req. 2.40

```
/*
 * getUser returns the information in a user entry from the table users. If the given user_name is null, records for all users
 * are returned. Returns null if the given user is not present.
 * Post-condition: User information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getUser (String   user_name)                // user


/*
 * assignQualUser stores the given user_name and qualification_id in the table qual_users.
 * Checks for presence of user and qualification and throws an error if not present
 * Post-condition: user qualification access permission has been stored
 * Called by business layer database connection object
 *
 */
void assignQualUser (String   user_name                // username
                   , int      qual_id                  // qualification ID
                     ) throws UserNotFoundException, QualificationNotFoundException

/*
 * getQualUsers returns the usernames of the users that have access to the given qualification.
 * Returns null if the given qualification is not present.
 * Post-condition: User information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getQualUsers (int   qual_id)                // qualification ID
```

```
/*
 * removeQualUser removes the given  user_id and qualification_id from the table qual_users.
 * Post-condition: user qualification access is no longer present in the table qual_users
 * Called by business layer database connection object
 *
 */
void removeQualUser ( String   user_name          // username
                    , String   qual_id            // qualification ID
                    )

/*
 * storeProduct stores the given product information in the table product. If product ID is not null,
 * the corresponding product is updated. Otherwise a new product is saved. Returns the product ID.
 * Checks that product name is unique in database when saving new product, otherwise throws error.
 * Post-condition: given product has been stored or updated in table product,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 *
 */
int storeProduct ( int       product_id           // product ID
                 , String   product_name          // product name
                 , String   requal_interval       // re-qualification interval
                 , String   condition             // product condition (e.g. obsolete)
                 , String   comment               // additional information
                 , String   username              // user performing operation
                 )  throws NotUniqueException
```

Req. 2.22

```
/*
 * getProduct returns the information in a product entry from the table users. If the given product name is null, records for all
 * products are returned. Returns null if the given product is not present.
 * Post-condition: Product information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getProduct (String   product_name)                  // product
```

```
/*
 * removeProduct removes a given product from the table products if the product ID is not referenced by other tables.
 * Checks if the product has links to other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: Product entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeProduct ( int   product_id                    // product ID
                        ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeEquipProduct stores the product ID for the given equipment version in the table product_equipment. If the information
 * is already stored, no action is taken.
 * Checks if product ID and equipment ID and version are present in database and throws error if not
 * Post-condition: given equipment - product information has been stored in table product_equipment
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 *
 */
void storeEquipProduct ( int       equipment_id          // equipment ID
                       , int       equipment_version     // equipment version (e.g. use for different products)
                       , int       product_id            // product ID
                       , String    comment               // additional information
                       , String    username              // user performing operation
                       ) throws EquipmentNotFound, ProductNotFound
```

Req. 2.25

```
/*
 * removeEqipProduct removes the given product ID for the given equipment version in the table product_equipment.
 * Post-condition: given product and equipment linkage is no longer stored in the table product_equipment
 *                 any corresponding comment has been deleted from the table comment
 * Called by business layer database connection object
 *
 */
void removeEquipProduct ( int       equipment_id          // equipment ID
                        , int       equipment_version     // equipment version (e.g. use for different products)
                        , int       product_id            // product ID
                        , String    username )            // user performing operation
```

```
/*
 * getEquipProduct returns a collection of product ID:s for the given equipment version from the table product_equipment.
 * Returns null if no entries are found.
 * Post-condition: collection of product ID:s and comments have been returned if found
 * Called by business layer database connection object
 *
 */
Collection getEquipProduct ( int     equipment_id          // equipment ID
                           , int     equipment_version )   // equipment version (e.g. use for different products)


/*
 * storeEquipment stores the given equipment information in the table equipment. If new_version is “true” and equipment ID is not null
 * a new version of the equipment is stored. If  new_version is “false” and equipment ID and version is not null, the equipment
 * information is updated. Returns the equipment ID and version.
 * Checks if the given room, department, equipment type exist in the database and throws exception if not.
 * Post-condition: given equipment information has been stored in table equipment
 *                 if new_version true, a new equipment version has been created
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
Collection storeEquipment (int      equipment_id          // equipment ID
                           , int    equip_version         // equipment version
                           , String new_version           // true if new version should be created
                           , String equipment_name        // name of the equipment
                           , int    equip_type_id         // equipment type ID
                           , int    room_id               // room ID
                           , int    department_id         // department ID
                           , String requal                // re-qualification interval
```

```
                    , String   status                 // equipment status (e.g. current, obsolete)
                    , String   qual_condition         // qualification condition (e.g. approved, )
                    , String   comment                // additional information
                    , String   username               // user performing operation
                    ) throws RoomNotFound, DepartmentNotFound, EquipTypeNotFound
```

Req. 2.24, 2.26

```
/*
 * getEquipment returns a collection of equipment information for the given equipment version from the tables equipment and comment.
 * Returns null if no entries are found. Returns all equipments if equipment ID and version is null.
 * Post-condition: collection of equipments and comments have been returned if found
 * Called by business layer database connection object
 *
 */
Collection getEquipment ( int       equipment_id          // equipment ID
                        , int       equipment_version )   // equipment version (e.g. use for different products)


/*
 * searchEquipment returns a collection of equipment information for the given name and type from the tables equipment and comment.
 * Returns null if no entries are found.
 * Post-condition: collection of equipments and comments have been returned if found
 * Called by business layer database connection object
 *
 */
Collection searchEquipment (String  name              // equipment name
                          , int     equip_type_id )   // equipment type ID
```

```
/*
 * storeEquipType stores of modifies a given equipment type in the table equipment_type.
 * Any given comment is stored in the table comment.
 * Checks for the presence of the equipment type, updates if found, inserts if not present
 * Post-condition: Equipment type has been stored or modified
 * Called by business layer database connection object
 *
 */
void storeEquipType ( int   equip_type_id          // equipment type id
                    , String   name                // equipment type
                    , String   status              // equipment type status (e.g. current, obsolete)
                    , String   comment             // additional information
                    , String   user)               // user performing operation


/*
 * removeEquipType removes an equipment type from the table equipment_type.
 * Checks if the id is references by other database tables, if so, does not remove and throws exception
 * Post-condition: Equipment type has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeEquipType ( String   class              // category
                     , String   value // value to remove
                     ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * getEquipType returns a collection of equipment types from the table equipment_types. If equip_type is null, all equipment types
 * are returned. Returns null if the given equipment type is not present.
 * Post-condition: Collection of equipment types and corresponding comments have been returned if data found
 * Called by business layer database connection object
 *
 */
Collection getEquipType (String   equip_type)                // equipment type to return values for


/*
 * storeProcessStep stores the given process step information in the table process_step. If process step ID is null, a new process step is created.
 * If the process step is already present, the process step information is updated. Returns the process step ID.
 * Checks if the combination of step name and product is unique when creating new process step, throws error if not.
 * Post-condition: given process step information has been stored in table process_step
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
int storeProcessStep (int       process_step_id      // process step ID
                    , String    step_name            // name of the process step
                    , int       product_id           // product ID
                    , String    requal               // re-qualification interval
                    , String    qual_condition       // qualification condition (e.g. approved, )
                    , String    comment              // additional information
                    , String    username             // user performing operation
                    ) throws NotUniqueException
```

Req. 2.23

```
/*
 * getProcessStep returns a collection of process steps and product names from the tables process_step and product.
 * If step_name is null, all process steps and corresponding product names are returned.
 * Returns null if the given step name is not present.
 * Post-condition: Collection of process steps and corresponding products and comments have been returned if data found
 * Called by business layer database connection object
 *
 */
Collection getProcessStep (String   step_name)                // process step to return values for
```

```
/*
 * removeProcessStep removes a given process step from the table process_step if the process step ID is not referenced by other tables.
 * Checks if the process step is referenced by other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: Process step entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeProcessStep ( int   process_step_id                // process step ID
                        ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeRoom stores the given room information in the table room. If room ID is not null,
 * the corresponding room is updated. Otherwise a new room is saved. Returns the room ID.
 * Checks that room name is unique in database when saving new room, otherwise throws error.
 *
 * Post-condition: given room has been stored or updated in table room
 *
 * Called by business layer database connection object
 *
 */
int storeRoom ( int       room_id              // room ID
              , String    room_name            // room name
              , String    status               // room status
              , String    username             // user performing operation
              )  throws NotUniqueException
```

Req. 2.30

```
/*
 * getRoom returns the information in a room entry from the table room. If the given room name is null, records for all
 * rooms are returned. Returns null if the given room is not present.
 * Post-condition: Room information has been returned in a collection if data found
 *
 * Called by business layer database connection object
 *
 */
Collection getRoom (String   room_name)            // room
```

```
/*
 * removeRoom removes a given room from the table room if the room ID is not referenced by other tables.
 * Checks if the room is referenced by other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: Room entry has been removed if not referenced by other database columns
 *
 * Called by business layer database connection object
 *
 */
void removeRoom ( int   room_id                // room ID
                        ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeDepartment stores the given department information in the table department. If departmemt ID is not null,
 * the corresponding department is updated. Otherwise a new department is saved. Returns the department ID.
 * Checks that department name is unique in database when saving new department, otherwise throws error.
 * Post-condition: given department has been stored or updated in table department
 *
 * Called by business layer database connection object
 *
 */
int storeDepartment ( int       department_id          // department ID
                        , String   department_name     // department name
                        , String   username            // user performing operation
                          )  throws NotUniqueException
```

Req. 2.30

```
/*
 * getDepartment returns the information in a department entry from the table room. If the given department name is null, records for all
 * departments are returned. Returns null if the given department is not found.
 * Post-condition: Department information has been returned in a collection if data found
 *
 * Called by business layer database connection object
 *
 */
Collection getDepartment (String   department_name)                  // department
```

```
/*
 * removeDepartment removes a given department from the table department if the department ID is not referenced by other tables.
 * Checks if the department is referenced by other database tables, and if so, does not remove the entry and throws an exception.
 * Post-condition: Department entry has been removed if not referenced by other database columns
 *
 * Called by business layer database connection object
 *
 */
void removeDepartment ( int   department_id                    // department ID
                        ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeInstruction stores the given instruction information in the table instruction. If instruction ID is null, a new instruction is created.
 * If the instruction is already present, the information is updated. Returns the instruction ID.
 * Checks if the category and equipment version exist in the database, throws error if not.
 * Post-condition: given instruction information has been stored in table instruction,
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
int storeInstruction (int    instruction_id    // instruction ID
                    , String instruction       // instruction number
                    , int    category_id       // instruction category ID
                    , int    equip_id          // equipment ID
                    , int    equip_version     // equipment version
                    , String status            // status of instruction (e.g. approved)
                    , String comment           // additional information
                    , String username          // user performing operation
                      ) throws CategoryNotFound, EquipNotFound


/*
 * getEquipInstruction returns a collection of instruction information from the tables instruction and comment.
 * Returns null if the given equipment ID is not present.
 * Post-condition: Collection of instructions and comments have been returned if data found
 *
 * Called by business layer database connection object
 *
 */
Collection getEquipInstruction (int   equipment_id)          // equipment to return instruction for
```

```
/*
 * storeCategory stores the given instruction category information in the table instruction category. If category ID is not null,
 * the corresponding category is updated. Otherwise a new category is saved. Returns the category ID.
 * Checks that category name is unique in database when saving new category, otherwise throws error.
 * Post-condition: given category has been stored or updated in table instruction_category
 *
 * Called by business layer database connection object
 *
 */
int storeCategory ( int       category_id        // category ID
                  , String   category_name      // category name
                  , String   username           // user performing operation
                  )  throws NotUniqueException



/*
 * getCategory returns the information in a category entry from the table instruction_category. If the given category name is null,
 * records for all category are returned. Returns null if the given category is not found.
 * Post-condition: category information has been returned in a collection if data found
 *
 * Called by business layer database connection object
 *
 */
Collection getCategory (String   category_name)              // instruction category
```

```
/*
 * removeCategory removes a given category from the table instruction_category if the category ID is not referenced by other tables.
 * Checks if the category is referenced by other database tables, and if so, does not remove the entry and throws an exception.
 * Post-condition: Category entry has been removed if not referenced by other database columns
 *
 * Called by business layer database connection object
 *
 */
void removeCategory ( int   category_id                 // instruction category ID
                    ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeMtrlCode stores the given material code information in the table material_code. If material code ID is null,
 * a new material code is created. If the material code is already present, the information is updated.
 * If new_version is set to true, a new version of the material code is created. Returns the material code ID.
 *
 * Checks if the material code is unique when creating new material code, throws exception if not.
 * Checks if product ID and department ID exist in the database, throws exception if not.
 *
 * Post-condition: given material code information has been stored in table material_code,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 */
int storeMtrlCode ( int       mtrl_code_id              // material code ID
                  , int         mtrl_code_version     // material code version
                  , String  new_version               // true if new version should be created
```

```
, int       department_id           // department ID
, int       product_id              // product ID
, String    name                    // material name
, String    requal_interval         // re-qualification interval
, String    qual_condition          // qualification condition (e.g. approved, )
, String    comment          // additional information
, String    username                // user performing operation
) throws NotUniqueException, ProdNotFoundException, DeptNotFoundException
```

```
/*
 * getMtrlCode returns a collection of material codes and product names from the tables material_code, product and comment.
 * If step_name is null, all material codes and the corresponding product names are returned.
 * Returns null if the given material code is not present.
 * Post-condition: Collection of material codes and corresponding products and comments have been returned if data found
 * Called by business layer database connection object
 *
 */
Collection getMtrlCode (String   code_name)                  // material code to return records for
```

```
/*
 * removeMtrlCode removes a given material code from the table material_code if the material code ID is not referenced by other tables.
 * Checks if the material code is referenced by other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: Material code entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeMtrlCode ( int   mtrl_code_id              // material code ID
                    ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeMethod stores the given method information in the table method. If method ID is null,
 * a new method is created. If the method is already present, the information is updated.
 * Returns the method ID.
 *
 * Checks if the method is unique when creating new method, throws exception if not.
 * Checks if department ID exist in the database, throws exception if not.
 *
 * Post-condition: given method information has been stored in table method,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 */
int storeMethod ( int       method_id          // method ID
                , int       department_id      // department ID
                , String    method_name        // method name
                , String    requal_interval    // re-qualification interval
```

```
, String   qual_condition      // qualification condition (e.g. approved, )
, String   comment             // additional information
, String   username            // user performing operation
) throws NotUniqueException, DeptNotFoundException
```

Req. 2.27

```
/*
 * getMethod returns a collection of methods and departments from the tables method, department and comment.
 * If method_name is null, all method and the corresponding departments are returned.
 * Returns null if the given method is not present.
 * Post-condition: Collection of method and corresponding departments and comments have been returned if data found
 * Called by business layer database connection object
 *
 */
Collection getMethod (String method _name)                    // method to return records for


/*
 * removeMethod removes a given method from the table method if the method ID is not referenced by other tables.
 * Checks if the method is referenced by other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: method entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeMethod ( int   method _id                    // method ID
                    ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeSeries stores the given series information in the table series. If series ID is not null,
 * the corresponding series is updated. Otherwise a new series is saved. Returns the series ID.
 * Checks that series is unique in database when saving new series, otherwise throws error.
 * Post-condition: given series has been stored or updated in table series,
 *                      comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 *
 */
int storeSeries ( int       series_id           // series ID
                , String    series_number       // product name
                , String    status              // series status (e.g. current)
                , String    comment             // additional information
                , String    username            // user performing operation
                  )  throws NotUniqueException
```

Req. 2.31

```
/*
 * getSeries returns the information in a series entry from the tables series and comment. If the given series name is null, records for all
 * series are returned. Returns null if the given series is not present.
 * Post-condition: series information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getSeries (String   series_number)                  // number of series
```

```
/*
 * removeSeries removes a given series from the table series if the series ID is not referenced by other tables.
 * Checks if the series has links to other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: series entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeSeries ( int   series_id              // series ID
                  ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeQualType stores the given qualification type information in the table qualification_type.
 * If qualification type ID is not null, the corresponding qualification type is updated.
 * Otherwise a new qualification type is saved. Returns the qualification type ID.
 * Checks that qualification type is unique in database when saving new qualification type, otherwise throws error.
 * Post-condition: given qualification type has been stored or updated in table qualification_type,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 *
 */
int storeQualType ( int      qual_type_id     // qualification type ID
                  , String   qual_type        // qualification type
                  , String   status           // qualification type status (e.g. current)
                  , String   comment          // additional information
                  , String   username         // user performing operation
                  )  throws NotUniqueException
```

```
/*
 * getQualType returns the information in a qualification type entry from the tables qualification_type and comment.
 * If the given qualification type is null, records for all qualification types are returned.
 * Returns null if the given qualification type is not present.
 * Post-condition: qualification type information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getQualType (String qual_type)              // qualification type
```

```
/*
 * removeQualType removes a given qualification type from the table qualification_type if the qualification type ID is not referenced by other tables.
 * Checks if the qualification type has links to other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: qualification type entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeQualType ( int   qual_type_id               // qualification type ID
                    ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeSubQualType stores the given sub-qualification type information in the table subqual_type. If subqual_type ID is not null,
 * the corresponding sub-qualification type is updated. Otherwise a new sub-qualification type is saved.
 * Returns the sub-qualification type ID.
 * Checks that sub-qualification type is unique in database when saving new series, otherwise throws error.
 * Post-condition: given sub-qualification type has been stored or updated in table subqual_type,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 *
 */
int storeSubQualType ( int      subqual_type_id     // sub-qualification type ID
                     , String   sub_qual_type       // sub-qualification type
                     , String   status              // sub-qualification type status (e.g. current)
                     , String   comment             // additional information
                     , String   username            // user performing operation
                     )  throws NotUniqueException


/*
 * getSubQualType returns the information in a sub-qualification type entry from the tables subqual_type and comment.
 * If the given sub-qualification type name is null, records for all sub-qualification type are returned.
 * Returns null if the given sub-qualification type is not present.
 * Post-condition: sub-qualification type information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getSubQualType (String   subqual_type)          // sub-qualification type
```

```
/*
 * removeSubQualType removes a given sub-qualification type from the table subqual_type if the sub-qualification type ID
 * is not referenced by other tables.
 * Checks if the sub-qualification type has links to other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: sub-qualification type entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 */
void removeSubQualType ( int   subqual_type_id              // sub-qualification type ID
                                ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storePartQualType stores the given partial qualification type information in the table partial_qual_type.
 * If partial qualification type ID is not null, the information is updated. Otherwise a new partial qualification type is saved.
 * Returns the partial qualification type ID.
 * Checks that partial qualification type is unique in database when saving new partial qualification type, otherwise throws error.
 * Post-condition: given partial qualification type has been stored or updated in table series,
 *                      comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 *
 */
int storePartQualType ( int       part_qual_type_id          // partial qualification type ID
                        , String  part_qual_type             // partial qualification type
                        , String  status                     // partial qualification type status (e.g. current)
                        , String  comment                    // additional information
                        , String  username                   // user performing operation
                        )  throws NotUniqueException
```

```
/*
 * getPartQualType returns the information in a partial qualification type entry from the tables partial_qual_type and comment.
 * If the given partial qualification type is null, records for all partial qualification types are returned.
 * Returns null if the given partial qualification type is not found.
 * Post-condition: partial qualification type information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getPartQualType (String   part_qual_type)                    // partial qualification type
```

```
/*
 * removePartQualType removes a given partial qualification type from the table partial_qual_type if
 * the partial qualification type ID is not referenced by other tables.
 * Checks if the partial qualification type has links to other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: partial qualification type entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removePartQualType ( int   part_qual_type_id              // partial qualification type ID
                                    ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeQualification stores the given qualification information in the table qualification. If qualification ID is not null
 * the qualification information is updated. Returns the qualification ID.
 * Checks if the given series, qualification type, sub-qualification type, and partial qualification type
 * exist in the database and throws exception if not.
 * Checks that qualification number to be stored is unique and throws exception if not.
 * Post-condition: given qualification information has been stored in table qualification
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 */
int storeQualification ( int       qualification _id      // qualification ID
                       , String  qual_title             // qualification title
                       , int       series_id            // series ID
                       , int       qual_type_id         // qualification type ID
                       , int       subqual_type_id      // sub-qualification type ID
                       , int       part_qual_type_id    // partial qualification type ID
                       , String  leader               // qualification leader
                       , String   project             // project number
                       , Date     approval_date       // date of approval of qualification
                       , String  qual_condition       // qualification condition (e.g. online, approved)
                       , String  status               // qualification status
                       , String   comment             // additional information
                       , String   username            // user performing operation
                       ) throws RoomNotFound, DepartmentNotFound, EquipTypeNotFound, NotUniqueException
```

Req. 2.1, 2.3, 2.4, 2.6, 2.7, 2.9, 2.10, 2.15, 2.16, 2.19
2.18 – absence of remove method

```
/*
 * getQualification returns a collection of qualification information for the given qualification from the tables qualification and comment.
 * Returns null if no entries are found.
 * Post-condition: collection of qualification and comment has been returned if found
 * Called by business layer database connection object
 *
 */
Collection getQualification ( String       qualification_number )       // qualification number


/*
 * searchQualification returns a collection of qualifications for the given search parameters. Returns null if no entries are found.
 * Post-condition: collection of qualifications has been returned if found
 * Called by business layer database connection object
 *
 */
Collection searchQualification ( String       qualification_number          // qualification number
                               , String       subqual_type                  // sub-qualification type
                               , String       title                         // qualification title
                               , String       leader                        // qualification leader
                               , String       product                       // qualified product
                               , String       process_step                  // qualified process step
                               , String       equipment                     // qualified equipment
                               , String       material_code                 // qualified material code
                               , String       method                        // qualified method
                               , String       room                          // room concerned
                               , String       qual_condition                // qualification condition
                               , String       status )                      // qualification status
```

Req. 2.33, 2.34

```
/* storeQualCC stores the change control number for the given qualification in the table qual_change_control.
 * If the information is already stored, no action is taken.
 * Checks if qualification ID is present in database and throws error if not
 * Post-condition: given change control information has been stored in table qual_change_control
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
void storeQualCC ( int          qualification_id        // qualification ID
                 , String       change_ctrl             // change control number
                 , String       comment                 // additional information
                 , String       username                // user performing operation
                 )  throws QualNotFoundException
```

Req. 2.29

```
/* storeRemActivity stores the given activity information in the table remaining activities. If activity ID is null,
 * a new activity is created. If the activity is already present, the information is updated. Returns the activity ID.
 * Checks if the given qualification ID exists in the database, throws exception if not.
 * Post-condition: given activity information has been stored in table remaining_activities
 * Called by business layer database connection object
 */
int storeRemActivity ( int        activity_id           // activity ID
                     , int        qualification_id      // qualification ID
                     , String     activity              // remaining qualification activity
                     , String     responsible           // person responsible for activity
                     , String     status                // activity status
                     ,  Date      completed             // date completed
                     , String     username              // user performing operation
                     ) throws QualNotFoundException
```

Req 2.21

```
/*
 * getRemActivity returns the information in a remaining activity entry from the table remaining_activity.
 * If the given activity or qualification number is null, records for all activities or qualification numbers are returned.
 * Returns null if the given activity for the given qualification is not found.
 * Post-condition: remaining activity information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getRemActivity ( String    activity                // remaining activity
                          , String    qual_number)            // qualification number


/*
 * storeQualProcess stores the process step for the given qualification in the table qual_process_step.
 * If the information is already stored, no action is taken.
 * Checks if qualification ID and process step ID are present in database and throws exceptions if not
 * Post-condition: given process step information has been stored in table qual_process_step,
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
void storeQualProcess ( int     qualification_id        // qualification ID
                      , int     process_step_id         // process step ID
                      , String          comment         // additional information
                      , String          username        // user performing operation
                      )  throws QualNotFoundException, ProcessNotFoundException
```

Req. 2.28

```
/*
 * getQualProcess returns a collection of process steps for the given qualification from the table qual_process_step.
 * Returns null if no entries are found.
 * Post-condition: collection of process steps and comments have been returned if found
 * Called by business layer database connection object
 *
 */
Collection getQualProcess ( int     qualification_id )          // qualification ID
```

```
/*
 * removeQualProcess removes the given process step for the given qualification in the table qual_process_step.
 * Post-condition: given process step for the given qualification is no longer stored in the table qual_process_step
 *                      any corresponding comment has been deleted from the table comment
 * Called by business layer database connection object
 *
 */
void removeQualProcess ( int        qualification_id          // qualification ID
                       , int        process_step_id           // process step ID
                       , String     username                  // user performing operation
                       )
```

```
/*
 * storeQualMtrl stores the material code version for the given qualification in the table qual_material_code.
 * If the information is already stored, no action is taken.
 * Checks if qualification ID and material code ID and version are present in database and throws exceptions if not
 * Post-condition: given material code information has been stored in table qual_material_code,
 *                      comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
void storeQualMtrl ( int        qualification_id        // qualification ID
                   , int        mtrl_code_id            // material code ID
                   , int        mtrl_code_version       // material code version
                   , String     comment                 // additional information
                   , String     username                // user performing operation
                   )  throws QualNotFoundException, MtrlNotFoundException
```

Req. 2.28

```
/*
 * getQualMtrl returns a collection of material codes for the given qualification from the table qual_material_code.
 * Returns null if no entries are found.
 * Post-condition: collection of material codes and comments have been returned if found
 * Called by business layer database connection object
 *
 */
Collection getQualMtrl ( int     qualification_id )              // qualification ID
```

```
/*
 * removeQualMtrl removes the given material code version for the given qualification in the table qual_material_code.
 * Post-condition: given material code version for the given qualification is no longer stored in the table qual_material_code,
 *                 any corresponding comment has been deleted from the table comment
 * Called by business layer database connection object
 *
 */
void removeQualMtrl ( int       qualification_id          // qualification ID
                    , int       mtrl_code_id             // process steo ID
                    , int       mtrl_code_version        // material code version
                    , String    username                 // user performing operation
                    )


/*
 * storeQualEquip stores the given equipment version for the given qualification in the table qual_equipment.
 * If the information is already stored, no action is taken.
 * Checks if qualification ID and equipment ID are present in database and throws exceptions if not
 * Post-condition: given equipment information has been stored in table qual_equipment,
 *                 comment has been saved in table comment and linked
 * Called by business layer database connection object
 */
void storeQualEquip ( int      qualification_id       // qualification ID
                    , int      equipment_id           // process step ID
                    , int      equipment_version      // equipment version
                    , String          comment         // additional information
                    , String          username        // user performing operation
                    )  throws QualNotFoundException, EquipNotFoundException
```

Req. 2.28

```
/*
 * getQualEquip returns a collection of equipments for the given qualification from the table qual_equipment.
 * Returns null if no entries are found.
 *
 * Post-condition: collection of equipments and comments have been returned if found
 *
 * Called by business layer database connection object
 *
 */
Collection getQualEquip ( int     qualification_id )              // qualification ID
```

```
/*
 * removeQualEquip removes the given equipment version for the given qualification in the table qual_equipment.
 *
 * Post-condition: given equipment for the given qualification is no longer stored in the table qual_equipment,
 *                 any corresponding comment has been deleted from the table comment
 *
 * Called by business layer database connection object
 *
 */
void removeQualEquip ( int             qualification_id         // qualification ID
                     , int             equipment_id             // equipment ID
                     , int   equipment_version     // equipment version
                     , String   username             // user performing operation
                     )
```

```
/*
 * storeQualMethod stores the method for the given qualification in the table qual_method.
 * If the information is already stored, no action is taken.
 * Checks if qualification ID and method ID are present in database and throws exceptions if not
 *
 * Post-condition: given method information has been stored in table qual_method,
 *                      comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 */
void storeQualProcess ( int    qualification_id        // qualification ID
                        , int    method _id            // method ID
                        , String          comment      // additional information
                        , String          username     // user performing operation
                        )  throws QualNotFoundException, MethodNotFoundException
```

Req. 2.28

```
/*
 * getQualMethod returns a collection of methods for the given qualification from the table qual_method.
 * Returns null if no entries are found.
 *
 * Post-condition: collection of methods and comments have been returned if found
 *
 * Called by business layer database connection object
 *
 */
Collection getQualMethod ( int    qualification_id )          // qualification ID
```

```
/*
 * removeQualMethod removes the given method for the given qualification in the table qual_method.
 * Post-condition: given method for the given qualification is no longer stored in the table qual_method,
 *                 any corresponding comment has been deleted from the table comment
 * Called by business layer database connection object
 *
 */
void removeQualMethod ( int         qualification_id     // qualification ID
                      , int         method _id           // method ID
                      , String      username             // user performing operation
                      )

/*
 * storeDocType stores the given document type information in the table document_type. If document type ID is not null,
 * the corresponding document type is updated. Otherwise a new document type is saved.
 * Returns the document type ID.
 * Checks that document type is unique in database when saving new document type, otherwise throws error.
 * Post-condition: given document type has been stored or updated in table document_type,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 *
 */
int storeSubQualType ( int         doc_type_id        // document type ID
                     , String      document_type      // document type
                     , String      status             // document type status (e.g. current)
                     , String      comment            // additional information
                     , String      username           // user performing operation
                     )  throws NotUniqueException
```

```
/*
 * getDocType returns the information in a document type entry from the tables document_type and comment.
 * If the given document type name is null, records for all document types are returned.
 * Returns null if the given document type is not present.
 * Post-condition: document type information has been returned in a collection if data found
 * Called by business layer database connection object
 *
 */
Collection getDocType (String   document_type)              // document type
```

```
/*
 * removeDocType removes a given document type from the table document_type if the document type ID is not referenced by other tables.
 * Checks if the document type has links to other database tables, if so, does not remove the entry and throws an exception.
 * Post-condition: document type entry has been removed if not referenced by other database columns
 * Called by business layer database connection object
 *
 */
void removeQualType ( int   doc_type_id              // document type ID
                    ) throws ForeignReferencesException
```

Req. 2.18

```
/*
 * storeDocument stores the given document information in the table document. If document ID is not null
 * the document information is updated. Returns the document ID.
 * Checks if the given qualification ID and document type ID exist in the database and throws exception if not.
 *
 * Post-condition: given document information has been stored in table document,
 *                 comment has been saved in table comment and linked
 *
 * Called by business layer database connection object
 */
int storeDocument ( int      document _id          // document ID
                  , int      document_version      // document version
                  , String   new_version           // true if new version is to be created
                  , String   doc_title             // document title
                  , int      qualification_id      // qualification ID
                  , int      doc_type_id           // document type ID
                  , String   author                // document author
                  , Date     approval_date         // date of approval of document
                  , String   archive_location      // archive location
                  , String   archive_id            // archive ID
                  , String   store_path            // electronic storage path
                  , String   status                // document status
                  , String   comment               // additional information
                  , String   username              // user performing operation
                  ) throws QualNotFound, DocTypeNotFound
```

Req. 2.11, 2.12, 2.13, 2.17, 2.19
2.18 – absence of remove method

```
/*
 * getDocument returns a collection of document information for the given document from the tables document and comment.
 * Returns null if no entries are found.
 * Post-condition: collection of document and comment information has been returned if found
 * Called by business layer database connection object
 *
 */
Collection getDocument ( int       document_id          // document ID
                       , int       document_version )   // document version


/*
 * searchDocument returns a collection of documents for the given search parameters. Returns null if no entries are found.
 *
 * Post-condition: collection of documents has been returned if found
 *
 * Called by business layer database connection object
 *
 */
Collection searchDocument ( String  doc_type        // document type
                          , String  title           // document title
                          , String  author          // document leader
                          , String  archive_loc     // archive location
                          , String  status          // document status
                          )
```