**2.2.2 Đối tượng đọc giả**

1. **ObjReader**

Lớp ObjReader chứa thông tin về một độc giả.

Các thuộc tính bao gồm ReaderID, ReaderName, ReaderSex, ReaderStatus, và một vector BaRBooks lưu trữ sách đã mượn.

Các phương thức bao gồm getter và setter, cũng như các phương thức để thêm sách và lấy danh sách sách đã mượn.

**2. ListReader**

Lớp ListReader quản lý danh sách các độc giả.

- Thuộc Tính: vector<ObjReader> readers: Lưu trữ danh sách các độc giả. Đây là một vector chứa các đối tượng của lớp ObjReader. Vector cho phép lưu trữ nhiều độc giả và dễ dàng thực hiện các thao tác thêm, xóa, và truy xuất.

- Phương Thức:

* **ObjReader getReaderByID(int readerID):**   
  - Mục tiêu: Lấy thông tin của độc giả dựa trên readerID.

- Cách hoạt động: Duyệt qua danh sách các độc giả và so sánh readerID với reader.getReaderID(). Nếu tìm thấy, trả về đối tượng ObjReader tương ứng, ngược lại trả về một đối tượng không hợp lệ.

ObjReader ListReader::getReaderbyID(int readerID) {

for (const ObjReader& reader : readers) {

if (reader.getReaderID() == readerID) {

return reader;

}

}

}

* **void LoadBaRBookstoReader(ListBaRBook& barbooks):**

- Mục tiêu: Tải danh sách sách đã mượn (barbooks) vào danh sách các độc giả.

- Cách hoạt động: Duyệt qua danh sách các độc giả và danh sách sách đã mượn. Nếu có sự kết hợp giữa reader.getReaderID() và bb.getReaderID(), thêm sách vào danh sách của độc giả.

void ListReader::LoadBaRBookstoReader(ListBaRBook& barbooks) {

for (ObjReader& reader : readers) {

for (const ObjBaRBook& bb : barbooks.getBaRBooks()) {

if (bb.getReaderID() == reader.getReaderID()) {

reader.addBaRBooks(bb);

}

}

}

}

* **bool isReaderIDExists(int readerID**) :
* Mục tiêu: Kiểm tra xem mã độc giả đã tồn tại trong danh sách chưa.
* Cách hoạt động: Duyệt qua danh sách các độc giả và so sánh readerID với reader.getReaderID(). Nếu tìm thấy, trả về true, ngược lại trả về false.

bool ListReader::isReaderIDExists(int readerID) {

for (const ObjReader& reader : readers) {

if (reader.getReaderID() == readerID) {

return true;

}

}

return false;

}

* **bool isReaderActive(int readerID)** :
* Mục tiêu: Kiểm tra xem độc giả có đang hoạt động không.
* Cách hoạt động: Duyệt qua danh sách các độc giả, tìm độc giả theo readerID và kiểm tra reader.getReaderStatus(). Nếu trạng thái là 1, độc giả đang hoạt động, trả về true, ngược lại trả về false.

bool ListReader::isReaderActive(int readerID){

for (const ObjReader& reader : readers) {

if (reader.getReaderID() == readerID) {

if(reader.getReaderStatus()==1)

{

return true;

}

}

}

return false;

}

* **bool isBaRBookFull(int readerID)** :
* Mục tiêu: Kiểm tra xem độc giả đã mượn quá số lượng sách giới hạn chưa.
* Cách hoạt động: Duyệt qua danh sách các độc giả, tìm độc giả theo readerID và kiểm tra kích thước danh sách sách đã mượn. Nếu đã mượn đủ 3 cuốn sách, trả về true, ngược lại trả về false.

bool ListReader::isBaRBookFull(int readerID)

{

for (const ObjReader& reader : readers) {

if (reader.getReaderID() == readerID) {

if (reader.getBaRBooks().size() == 3) {

return true;

}

}

}

return false;

}

* bool isReaderHasBaRLimited(int readerID) :
* Mục tiêu: Kiểm tra xem độc giả có mượn sách giới hạn không.
* Cách hoạt động: Duyệt qua danh sách các độc giả, tìm độc giả theo readerID và kiểm tra danh sách sách đã mượn của độc giả. Nếu có cuốn sách nào đó là sách giới hạn (isBaRBookLimited()), trả về true, ngược lại trả về false.

bool ListReader::isReaderHasBaRLimited(int readerID) {

for (const ObjReader& reader : readers) {

if (reader.getReaderID() == readerID) {

for (const ObjBaRBook& bb : reader.getBaRBooks()) {

if (bb.isBaRBookLimited()) {

return true;

}

}

}

}

return false;

}

* **bool isReaderBorrowingBook(int readerID, int bookID)** :
* Mục tiêu: Kiểm tra xem độc giả có đang mượn cuốn sách có bookID không.
* Cách hoạt động: Duyệt qua danh sách các độc giả, tìm độc giả theo readerID, sau đó duyệt qua danh sách sách đã mượn của độc giả và so sánh bb.getBookID() với bookID. Nếu tìm thấy, trả về true, ngược lại trả về false.

bool ListReader::isReaderBorrowingBook(int readerID, int bookID)

{

for (const ObjReader& reader : readers) {

if (reader.getReaderID() == readerID) {

for (const ObjBaRBook& bb : reader.getBaRBooks()) {

if (bb.getBookID()==bookID) {

return true;

}

}

}

}

}

* **bool isReadrhasBaRBook(int readerID)** :
* Mục tiêu: Kiểm tra xem độc giả có sách mượn không.
* Cách hoạt động: Duyệt qua danh sách các độc giả, tìm độc giả theo readerID, sau đó lấy danh sách sách đã mượn của độc giả. Nếu danh sách có ít nhất một cuốn sách, trả về true, ngược lại trả về false.

bool ListReader::isReadrhasBaRBook(int readerID){

for (const ObjReader& reader : readers) {

if(reader.getReaderID() == readerID)

{

vector<ObjBaRBook> list = reader.getBaRBooks();

if(list.size()>0)

{

return true;

}

}

}

return false;

}

* **ObjReader addReader(const QString& readerName, const QString& readerSex):**
* Mục tiêu: Thêm một độc giả mới vào danh sách.
* Cách hoạt động: Tạo một số ngẫu nhiên readerID từ 100000 đến 999999. Sau đó, kiểm tra xem số này đã tồn tại chưa. Nếu chưa, tạo một đối tượng ObjReader mới với thông tin được cung cấp và thêm vào danh sách.

ObjReader ListReader::addReader(const QString& readerName, const QString& readerSex) {

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<int> dist(100000, 999999);

int readerID;

do {

readerID = dist(gen);

} while (isReaderIDExists(readerID)==true);

ObjReader reader(readerID, readerName, readerSex, 1);

readers.push\_back(reader);

saveToTextFile("C:/Users/84976/Documents/LibraryManager/txtfile/reader.txt");

return reader;

}

* **const vector<ObjReader> getReaders()** :
* Mục tiêu: Lấy danh sách các độc giả.
* Cách hoạt động: Trả về danh sách các độc giả.

const vector<ObjReader> ListReader::getReaders()

{

return readers;

}

* **void BlockReader(int readerID)** :
* Mục tiêu: Khóa một độc giả.
* Cách hoạt động: Duyệt qua danh sách các độc giả và tìm độc giả theo readerID. Nếu tìm thấy, đặt trạng thái của độc giả thành không hoạt động (0).

void ListReader::BlockReader(int readerID)

{

for(ObjReader& rd: readers)

{

if(rd.getReaderID() == readerID)

{

rd.setReaderStatus(0);

}

}

saveToTextFile("C:/Users/84976/Documents/LibraryManager/txtfile/reader.txt");

}

* **void unBlockReader(int readerID)** :
* Mục tiêu: Mở khóa một độc giả.
* Cách hoạt động: Duyệt qua danh sách các độc giả và tìm độc giả theo readerID. Nếu tìm thấy, đặt trạng thái của độ

void ListReader::unBlockReader(int readerID)

{

for(ObjReader& rd: readers)

{

if(rd.getReaderID() == readerID)

{

rd.setReaderStatus(1);

}

}

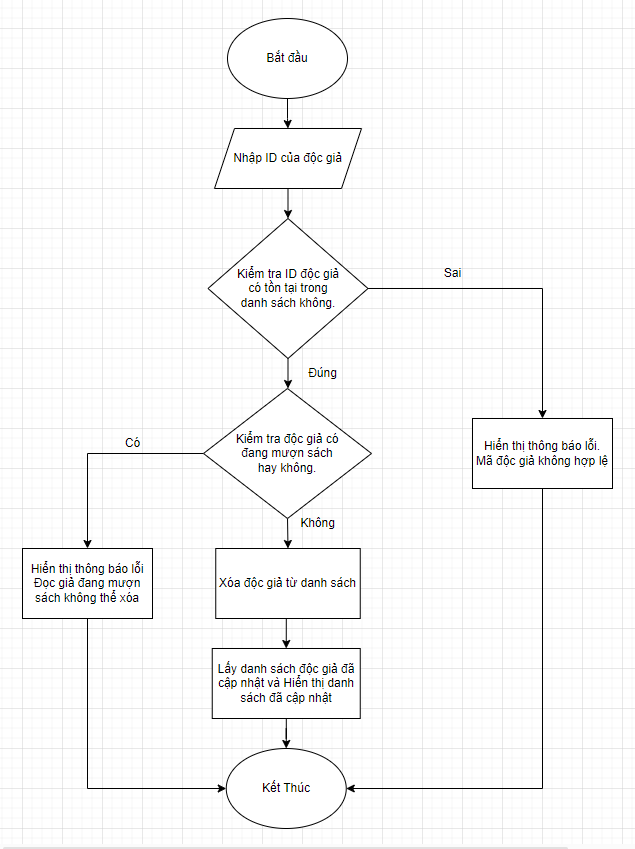
saveToTextFile("C:/Users/84976/Documents/LibraryManager/txtfile/reader.txt");

}

**3.2 Giao diện quản lí đọc giả**

**3.2.1 Bảng hiển thị**

**3.2.2 Cách thức hoạt động của nút nhấn**

* btnRemoveReader:

void reader::on\_btnRemoveReader\_clicked()

{

bool conversionSuccessful = false;

int ReaderID = ui->lbReaderID->text().toInt(&conversionSuccessful);

ListReader rd;

rd.loadFromTextFile(fileReader);

if (!conversionSuccessful) {

QMessageBox::critical(this, "Lỗi", "Mã độc giả không hợp lệ");

return;

}

else

{

ListBaRBook lbb;

lbb.loadFromTextFile(filebb);

rd.LoadBaRBookstoReader(lbb);

if(rd.isReaderIDExists(ReaderID)==false)

{

QMessageBox::critical(this, "Lỗi", "Mã độc giả không có trong thư viện");

return;

}

if(rd.isReadrhasBaRBook(ReaderID)==true)

{

QMessageBox::critical(this, "Lỗi", "Độc giả này đang mượn sách trong thư viện không thể xóa");

return;

}

ObjReader newReader = rd.getReaderbyID(ReaderID);

int newReaderID = newReader.getReaderID();

QString message = "Đã xóa độc giả có mã " + QString::number(newReaderID);

QMessageBox::information(this, "Thành công", message);

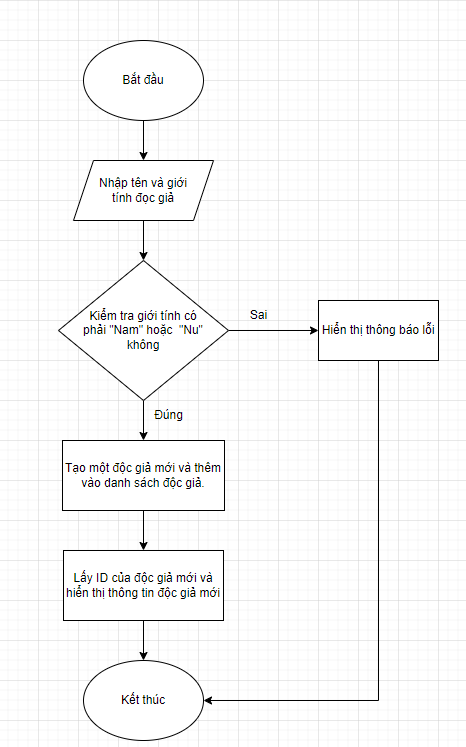
rd.removeReader(ReaderID);

}

vector<ObjReader> listrd = rd.getReaders();

displaytableListReader(ui->tableListReader,listrd);

}

* ****btnRegist:

void reader::on\_btnRegist\_clicked()

{

QString ipName = ui->txtinputName->toPlainText();

QString ipSex = ui->txtinputSex->toPlainText();

if(ipSex != "Nam" && ipSex != "Nu")

{

QMessageBox::critical(this, "Lỗi", "Giới tính phải là Nam hoặc Nu");

}

else{

ListReader rd;

rd.loadFromTextFile(fileReader);

ObjReader newReader = rd.addReader(ipName,ipSex);

vector<ObjReader> listrd = rd.getReaders();

displaytableListReader(ui->tableListReader,listrd);

int newReaderID = newReader.getReaderID();

QString message = "Mã độc giả vừa thêm vào là " + QString::number(newReaderID);

QMessageBox::information(this, "Thành công", message);

ui->groupProfileReader->setVisible(true);

ui->lbReaderID->setText(QString::number(newReaderID)); // Thêm dấu "->" và chuyển đổi dữ liệu thành QString

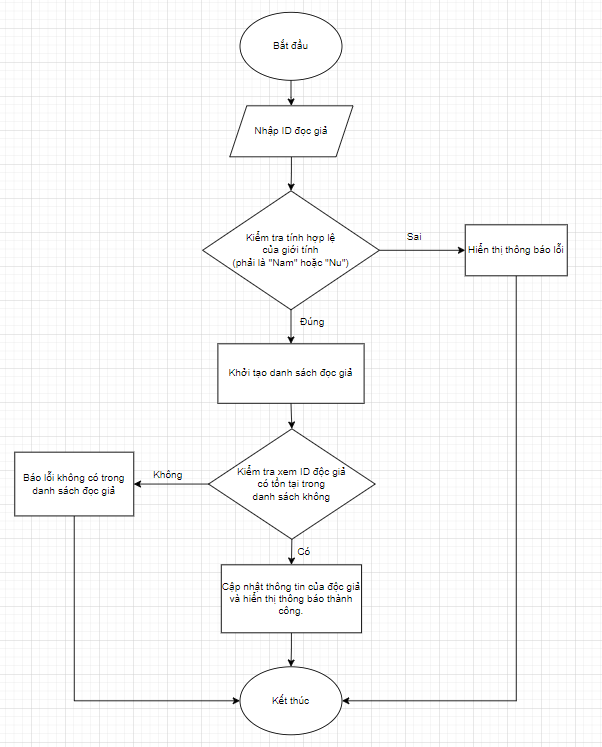
ui->txtNameReader->setPlainText(newReader.getReaderName()); // Thêm .setText()

ui->txtReaderSex->setPlainText(newReader.getReaderSex()); // Thêm .setText()

ui->lbReaderStatus->setText("Đang hoạt động"); // Thêm .setText()

}

}

* btnUpdateReader:

void reader::on\_btnUpdateReader\_clicked()

{

bool conversionSuccessful = false;

int ReaderID = ui->lbReaderID->text().toInt(&conversionSuccessful);

QString ReaderName = ui->txtNameReader->toPlainText();

QString ReaderSex = ui->txtReaderSex->toPlainText();

if (!conversionSuccessful) {

QMessageBox::critical(this, "Lỗi", "Mã độc giả không hợp lệ");

return;

}

if(ReaderSex != "Nam" && ReaderSex != "Nu")

{

QMessageBox::critical(this, "Lỗi", "Giới tính phải là Nam hoặc Nu");

return;

}

ListReader rd;

rd.loadFromTextFile(fileReader);

if(rd.isReaderIDExists(ReaderID)==false)

{

QMessageBox::critical(this, "Lỗi", "Mã độc giả không có trong thư viện");

}

else

{

ObjReader newReader = rd.getReaderbyID(ReaderID);

int newReaderID = newReader.getReaderID();

QString message = "Đã sửa thông tin độc giả có mã " + QString::number(newReaderID);

QMessageBox::information(this, "Thành công", message);

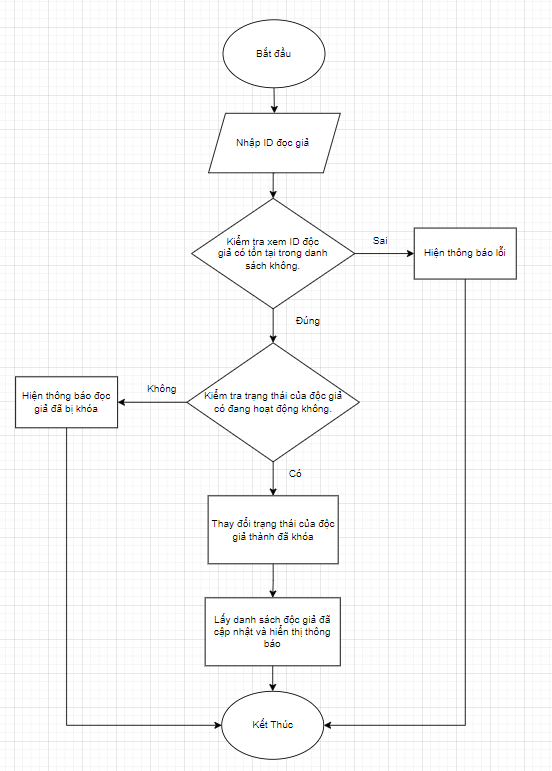
rd.updateReader(ReaderID,ReaderName,ReaderSex);

}

vector<ObjReader> listrd = rd.getReaders();

displaytableListReader(ui->tableListReader,listrd);

}

* btnBlockReader:

void reader::on\_btnBlockReader\_clicked()

{

bool conversionSuccessful = false;

int ReaderID = ui->lbReaderID->text().toInt(&conversionSuccessful);

if (!conversionSuccessful) {

QMessageBox::critical(this, "Lỗi", "Mã độc giả không hợp lệ");

return;

}

ListReader rd;

rd.loadFromTextFile(fileReader);

if (!rd.isReaderIDExists(ReaderID)) {

QMessageBox::critical(this, "Lỗi", "Mã độc giả không tồn tại trong thư viện");

return;

}

ObjReader newReader = rd.getReaderbyID(ReaderID);

if (newReader.getReaderStatus() == 1) {

rd.BlockReader(ReaderID);

QString message = "Đã khóa thẻ độc giả có mã" + QString::number(ReaderID);

QMessageBox::information(this, "Thành công", message);

} else {

QMessageBox::critical(this, "Lỗi", "Thẻ của độc giả bị khóa trước đó");

}

// Cập nhật giao diện người dùng

vector<ObjReader> listrd = rd.getReaders();

displaytableListReader(ui->tableListReader, listrd);

// Cập nhật thông tin độc giả

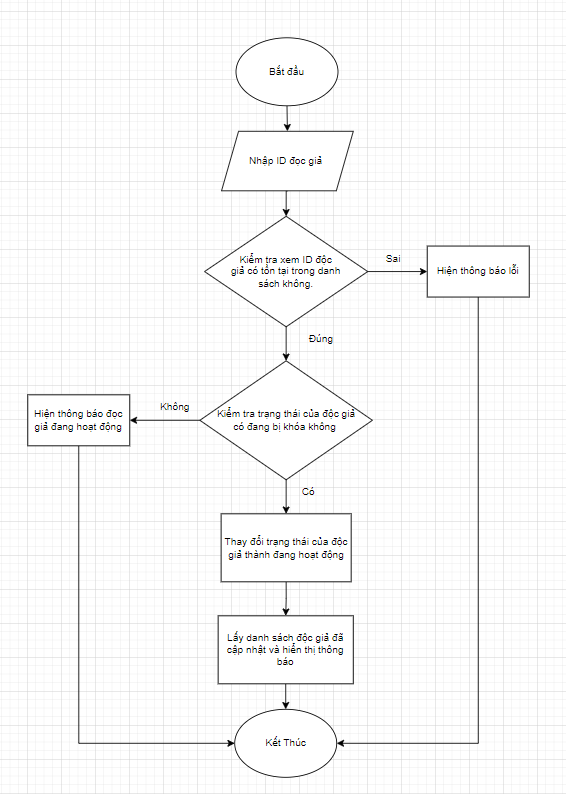
ui->lbReaderID->setText(QString::number(newReader.getReaderID()));

ui->txtNameReader->setPlainText(newReader.getReaderName());

ui->txtReaderSex->setPlainText(newReader.getReaderSex());

ui->lbReaderStatus->setText(newReader.getReaderStatus() == 0 ? "Thẻ bị khóa" : "Đang hoạt động");

}

* btnUnBlockReader:

void reader::on\_btnUnBlockReader\_clicked()

{

bool conversionSuccessful = false;

int ReaderID = ui->lbReaderID->text().toInt(&conversionSuccessful);

if (!conversionSuccessful) {

QMessageBox::critical(this, "Lỗi", "Mã độc giả không hợp lệ");

return;

}

ListReader rd;

rd.loadFromTextFile(fileReader);

if (!rd.isReaderIDExists(ReaderID)) {

QMessageBox::critical(this, "Lỗi", "Mã độc giả không tồn tại trong thư viện");

return;

}

ObjReader newReader = rd.getReaderbyID(ReaderID);

if (newReader.getReaderStatus() == 0) {

rd.unBlockReader(ReaderID);

QString message = "Đã mở lại thẻ độc giả có mã " + QString::number(ReaderID);

QMessageBox::information(this, "Thành công", message);

} else {

QMessageBox::critical(this, "Lỗi", "Thẻ của độc giả đã được mở khóa trước đó");

}

// Cập nhật giao diện người dùng

vector<ObjReader> listrd = rd.getReaders();

displaytableListReader(ui->tableListReader, listrd);

// Cập nhật thông tin độc giả

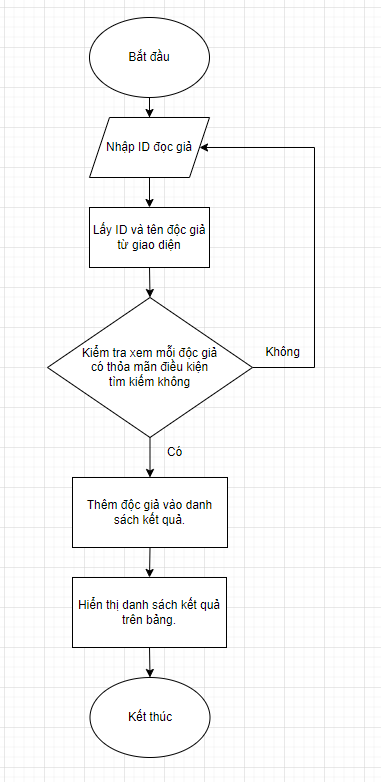
ui->lbReaderID->setText(QString::number(newReader.getReaderID()));

ui->txtNameReader->setPlainText(newReader.getReaderName());

ui->txtReaderSex->setPlainText(newReader.getReaderSex());

ui->lbReaderStatus->setText(newReader.getReaderStatus() == 0 ? "Thẻ bị khóa" : "Đang hoạt động");

}

* btnFindReader:

void reader::on\_btnFindReader\_clicked()

{

int ReaderID = ui->txtinputReaderIDFind->toPlainText().toInt();

QString ReaderName = ui->txtinputReaderNameFind->toPlainText();

vector<ObjReader> listrd;

ListReader lrd;

lrd.loadFromTextFile(fileReader);

for(const ObjReader& rd: lrd.getReaders())

{

if(rd.getReaderID()==ReaderID||rd.getReaderName()==ReaderName)

{

listrd.push\_back(rd);

}

}

displaytableListReader(ui->tableListReader,listrd);

ui->groupProfileReader->setVisible(false);

}