```cpp
// Frank Bodholdt Jacobsen
// Studienr. 197600001

// Opgave 1A

// Document.h
#include <string>
using namespace std;

class Document
{
public:
    // Constructor decucted from usage in Jpg class
    Document(string, bool);
    // Alle basisklasser bør have virtual destructor
    virtual ~Document();

    void setName(std::string);
    string getName() const;
    void setPriority(bool);
    bool getPriority() const;
    virtual void print() const = 0;

private:
    std::string name_;
    bool priority_;
};

// Document.cpp
#include "Document.h"

Document::Document(string name, bool prio)
    : name_(name), priority_(prio)
{

}

Document::~Document()
{

}

void Document::setName(string name)
{
    name_ = name;
}

string Document::getName() const
{
    return name_;
}

void Document::setPriority(bool prio)
{
    priority_ = prio;
}

bool Document::getPriority() const
{
    return priority_;
}

// Opgave 1B

// Txt.h
#pragma once
#include "Document.h"
#include <iostream>

class Txt : public Document
{
public:
    Txt(string, bool = false);
    ~Txt();
    void setText(char *);
    const char* getText() const;
    void print() const;

    // Ekstra metoder opgave 1F
    const Txt & operator=(const Txt &);
    Txt(const Txt &);

private:
    char * text_;

};

// Txt.cpp
#include "Txt.h"
#include <iostream>

Txt::Txt(string name, bool priority)
    : Document(name + ".txt", priority)
{
    text_ = new char[1]{ '\0' };
    *text_ = '\0';
    text_[0] = '\0';
}

Txt::~Txt()
{
    delete[] text_;
    cout << "Memory for text_ is deallocated" << endl;
}

void Txt::print() const
{
    cout << endl << (getPriority() == 1 ? "High: " : "Low:  ");
    cout << getName() << endl << endl;
    cout << text_ << endl << endl;
}

const char * Txt::getText() const
{
    return text_;
}

// Opgave 1C - del af txt.cpp
void Txt::setText(char * newText)
{
    if (strlen(text_) != strlen(newText))
    {
        delete[] text_;
        text_ = new char[strlen(newText) + 1];
    }

    strcpy(text_, newText);
}
```

```

// Opgave 1D
#include "Txt.h"
#include <iostream>

using namespace std;

int main()
{
    Document *docPtr;
    docPtr = new Txt("Eksempel");

    delete docPtr;
    docPtr = nullptr;

    Txt myText("MyTxt");
    myText.setText("Invitation til pyjamasparty");

    //myText.print();

    cout << myText;

    return 0;
}

// Opgave 1E
// Del af txt.h
ostream & operator<<(ostream &os, const Txt &);

// Del af txt.cpp
ostream & operator<<(ostream & os, const Txt &t)
{
    os << endl << (t.getPriority() == 1 ? "High: " : "Low:  ");
    os << t.getName() << endl << endl;
    os << t.getText() << endl << endl;

    return os;
}

// Opgave 1F
// To ekstra metoder i class Txt
class Txt : public Document
{
public:

    // Ekstra metoder opgave 1F
    const Txt & operator=(const Txt &);
    Txt(const Txt &);


};



// Opgave 2A

// Printer.h
#pragma once

#include "Document.h"
#include <deque>
using namespace std;

class Printer
```

```cpp
{
public:
    void addToQueue(Document *);
    void printNextDocument();

    void showQueue() const;

private:
    deque<Document *> printerQueue_;
};

// Printer.cpp
#include "Printer.h"

#include <iostream>
using namespace std;

void Printer::addToQueue(Document * d)
{
    if (d->getPriority())
    {
        // High priority, put in front
        printerQueue_.push_front(d);
    }
    else
    {
        printerQueue_.push_back(d);
    }
}

void Printer::printNextDocument()
{
    if (printerQueue_.empty())
    {
        cout << "Print queue is empty\n\n";
    }
    else
    {
        printerQueue_.front()->print();
        printerQueue_.pop_front();
    }
}

// Opgave 2B
// Tilføjet til main.cpp
// tilføj kode her jvf. opgave 2B
Printer printer;

printer.addToQueue(&text1);
printer.addToQueue(&pic1);
printer.addToQueue(&text2);
printer.addToQueue(&text3);
printer.addToQueue(&pic2);

for (int i = 0; i < 6; i++)
{
    printer.printNextDocument();
}

// Opgave 2C
// Del af printer.cpp
void Printer::showQueue() const
{
    deque<Document *>::const_iterator it;
    for (it = printerQueue_.begin(); it != printerQueue_.end(); it++)
    {
        const Document * dPtr = *it;

        if (dPtr->getPriority())
        {
            cout << "High: ";
        }
        else
        {
            cout << "Low:  ";
        }

        cout << dPtr->getName() << endl;
    }
}


```