

## MILESTONE 1

```
// Lokale.h
class Lokale
{
public:
    Lokale(int areal);
    void setAreal(int areal);
    int getAreal() const;
    void setType(char *type);
    virtual void print() const;
    virtual ~Lokale();
protected:
    char *type_;
public:
    int areal_;
};

// Lokale.cpp
Lokale::Lokale(int areal)
{
    areal_ = (areal > 0 ? areal : 0);
    type_ = nullptr;
}

void Lokale::setAreal(int areal)
{
    areal_ = (areal > 0 ? areal : areal_);
}

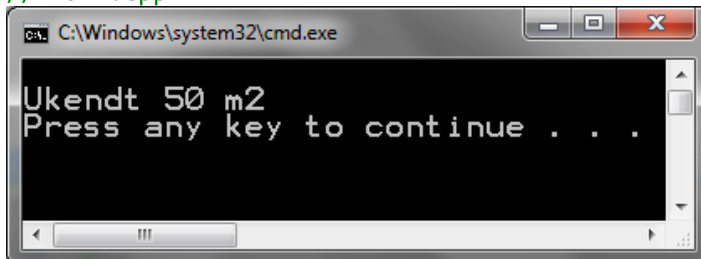
int Lokale::getAreal() const
{
    return areal_;
}

void Lokale::setType(char *type)
{
    if (type_ != nullptr)
        delete[] type_;

    type_ = new char[strlen(type) + 1];
    strcpy(type_, type);
}

Lokale::~~Lokale()
{
    delete [] type_;
}

// main.cpp
```



## MILESTONE 2

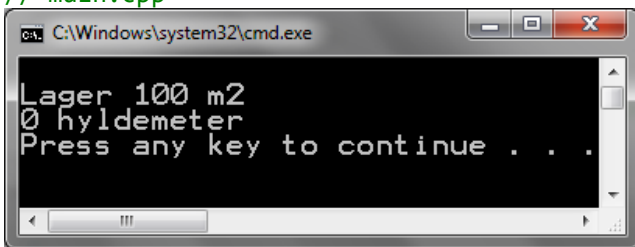
```
// Lager.cpp
Lager::Lager(int areal, int hyldemeter) : Lokale(areal)
{
    setType("Lager");
    hyldemeter_ = (hyldemeter > 0 ? hyldemeter : 0);
}

void Lager::setHyldemeter(int hyldemeter)
{
    hyldemeter_ = (hyldemeter > 0 ? hyldemeter : hyldemeter_);
}

int Lager::getHyldemeter() const
{
    return hyldemeter_;
}

void Lager::print() const
{
    Lokale::print();
    cout << hyldemeter_ << " hyldemeter" << endl;
}

// main.cpp
```



## MILESTONE 3

```
// MILESTONE 3 (Lokale.h)
Lokale(const Lokale &copyMe);
const Lokale & operator=(const Lokale &copyMe);

// MILESTONE 3 (Lokale.cpp)
Lokale::Lokale(const Lokale & copyMe)
{
    areal_ = copyMe.areal_;
    type_ = new char[strlen(copyMe.type_) + 1];
    strcpy(type_, copyMe.type_);
}

const Lokale & Lokale::operator=(const Lokale & copyMe)
{
    if (this != &copyMe)
    {
        areal_ = copyMe.areal_;

        if (strlen(type_) != strlen(copyMe.type_))
        {
            delete[] type_;
            type_ = new char[strlen(copyMe.type_) + 1];
        }
        strcpy(type_, copyMe.type_);
    }
    return *this;
}
```

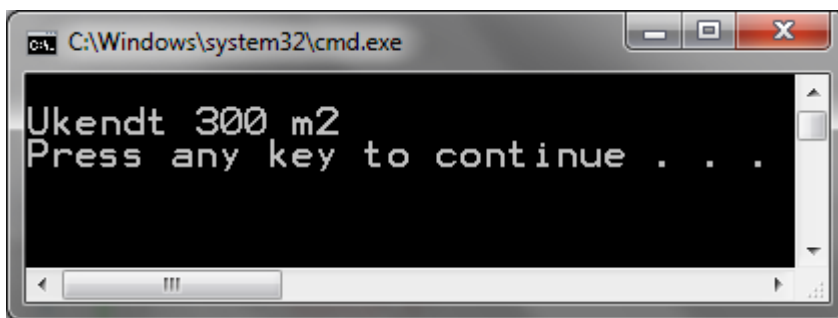
## MILESTONE 4

```
// MILESTONE 4 (Lokale.h)
const Lokale operator+(const Lokale &, const Lokale &);
```

```
// MILESTONE 4 (main.cpp)
int main()
{
    Lager l1(100), l2(200);
    Lokale sum = l1 + l2;

    sum.print();

    return 0;
}
```



## MILESTONE 5

```
// MILESTONE 4 (Erhvervsbygning.cpp)
Erhvervsbygning::Erhvervsbygning(const string &adresse, const string &ejer)
{
    adresse_ = adresse;
    ejer_ = ejer;
}

void Erhvervsbygning::setEjer(const string &ejer)
{
    ejer_ = ejer;
}

string Erhvervsbygning::getEjer() const
{
    return ejer_;
}

void Erhvervsbygning::tilfoejKontor(int areal, int antalBorde)
{
    lokaler_.push_back(new Kontor(areal, antalBorde));
}

void Erhvervsbygning::tilfoejFrokoststue(int areal, int antalPladser)
{
    lokaler_.push_back(new Frokoststue(areal, antalPladser));
}

void Erhvervsbygning::tilfoejLager(int areal, int hyldemeter)
{
    lokaler_.push_back(new Lager(areal, hyldemeter));
}

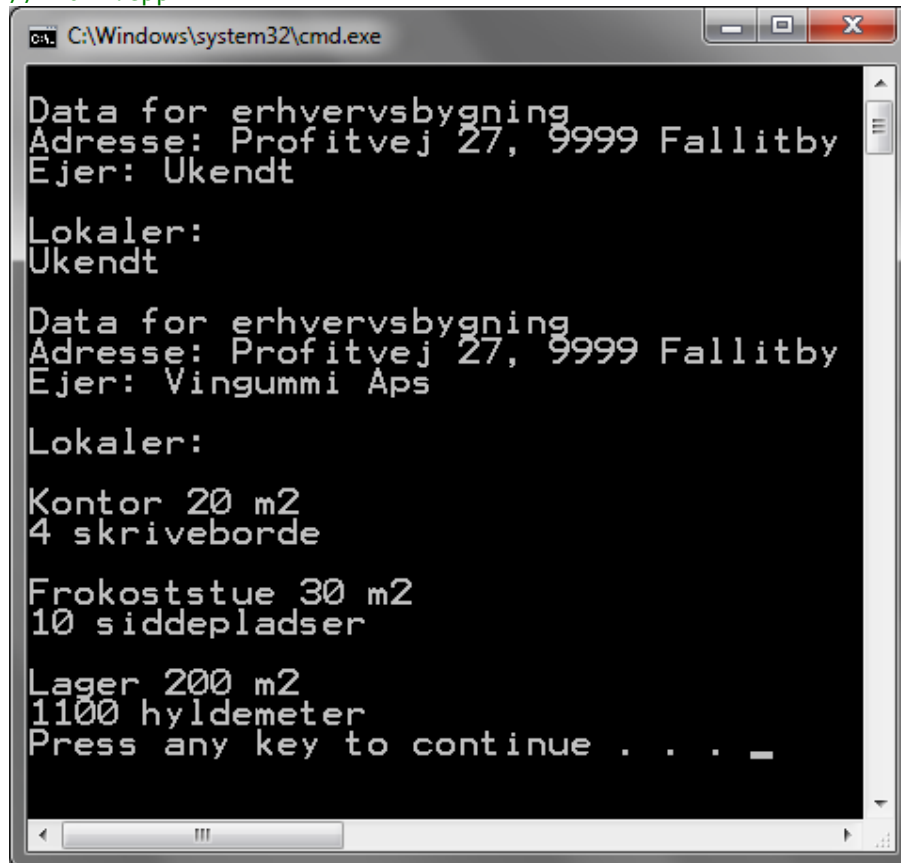
void Erhvervsbygning::print() const
{
    list<Lokale *>::const_iterator listIter;

    cout << endl << "Data for erhvervsbygning" << endl;
    cout << "Adresse: " << adresse_ << endl;
    cout << "Ejer: " << ejer_ << endl;
    cout << endl << "Lokaler:" << endl;

    if (lokaler_.size() > 0)
    {
        for (listIter = lokaler_.begin(); listIter != lokaler_.end(); ++listIter)
            (*listIter)->print();
    }
    else
        cout << "Ukendt" << endl;
}
```

## MILESTONE 6

// main.cpp



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The text displayed is the output of a C++ program, showing data for a commercial building. The output is as follows:

```
Data for erhvervsbygning
Adresse: Profitvej 27, 9999 Fallitby
Ejer: Ukendt

Lokaler:
Ukendt

Data for erhvervsbygning
Adresse: Profitvej 27, 9999 Fallitby
Ejer: Vingummi Aps

Lokaler:

Kontor 20 m2
4 skriveborde

Frokoststue 30 m2
10 siddepladser

Lager 200 m2
1100 hylde-meter
Press any key to continue . . . _
```