OOP F16 ordinær eksamen – løsningsforslag

OPGAVE 1

```
// Milestone 1 (ComplexNumber.cpp)
ComplexNumber::ComplexNumber(double realPart, double imaginaryPart)
{
     realPart_ = realPart;
     imaginaryPart_ = imaginaryPart;
}

double ComplexNumber::getRealPart() const
{
     return realPart_;
}

double ComplexNumber::getImaginaryPart() const
{
     return imaginaryPart_;
}


// Milestone 2 (ComplexNumber.h)
ostream& operator<<(ostream& out, const ComplexNumber& number);

ComplexNumber operator+(const ComplexNumber& left, const ComplexNumber& right);


// Milestone 2 (ComplexNumber.cpp)
ostream& operator<<(ostream& out, const ComplexNumber& number)
{
     number.print();

     return out;
}

ComplexNumber operator+(const ComplexNumber& left, const ComplexNumber& right)
{
     return ComplexNumber(left.getRealPart() + right.getRealPart(),
                          left.getImaginaryPart() + right.getImaginaryPart());
}


// Milestone 1, 2 og 3 (main)
int main()
{
     // Milestone 1
     {
          ComplexNumber z1(4, 2);

          z1.print();
     }
```

```
    // Milestone 2
    {
        ComplexNumber z1(4, 2);
        ComplexNumber z2(-6, 3);

        cout << z1 + z2 << "\n\n";
    }

    // Milestone 3
    {
        list<ComplexNumber> myList;
        ostream_iterator<ComplexNumber> myOutput(cout, "\n");

        myList.push_back(ComplexNumber(-2, 4));
        myList.push_back(ComplexNumber(6, -8));
        myList.push_back(ComplexNumber(10, 12));

        copy(myList.begin(), myList.end(), myOutput);
    }

    return 0;
}
```

OPGAVE 2

```
// Milestone 4
Ejer::Ejer(char *navn)
{
    setNavn(navn);
}

Ejer::~Ejer()
{
    delete [] navnPtr_;
}

void Ejer::setNavn(char *nytNavn)
{
    delete[] navnPtr_;
    navnPtr_ = new char[strlen(nytNavn) + 1];
    strcpy(navnPtr_, nytNavn);
}

// Klassen Ejer mangler en copy constructor og en assignment operator

int main()
{
    Ejer ejer("Kurt Hansen");

    ejer.printEjer();
    ejer.setNavn("Kurt Pedersen");
    ejer.printEjer();

    return 0;
}
```

```
// Milestone 5
class Ejerbolig
{
public:
     Ejerbolig(int areal);
     virtual void printBolig() const;
private:
     int areal_=0;                          // areal_ >= 0
};

Ejerbolig::Ejerbolig(int areal)
{
     areal_ = (areal > 0 ? areal : 0);
}

// Ændringer/tilføjelser til klassen Ejer:
Ejer(char *navn, Ejerbolig *bolig);         // ekstra parameter i Ejer ctor
Ejerbolig *boligPtr_ = nullptr;             // ekstra attribut i klassen Ejer
boligPtr_ = bolig;                          // ekstra statement i Ejer ctor
boligPtr_->printBolig();                    // ekstra statement i printEjer()

int main()
{
     Ejerbolig minBolig(120);
     Ejer boligEjer("Kurt Hansen", &minBolig);

     boligEjer.printEjer();

     return 0;
}

// Milestone 6
class Villa : public Ejerbolig
{
public:
     Villa(int areal, int antalPlan);
     virtual void printBolig() const;
private:
     int antalPlan_ = 0;         // [0;3]
};

Villa::Villa(int areal, int antalPlan) : Ejerbolig(areal)
{
     antalPlan_ = (1 <= antalPlan && antalPlan <= 3 ? antalPlan : 0);
}

int main()
{
     Villa minVilla(143, 2);
     Ejer villaEjer("Anne Poulsen", &minVilla);

     villaEjer.printEjer();

     return 0;
}
```