Klasa kao realizacija apstraktnog tipa podataka

- Kako bi to izgledalo na jeziku C++ korišćenjem samo onoga što smo do sada videli:

```
class complex {
public:
  complex (double real, double imag);
  complex (complex* other);  // For copy-initialization

  void copy (complex* other);  // For copy-assignment

  static complex* add (complex* c1, complex* c2);
  static complex* sub (complex* c1, complex* c2);

private:
  double re, im;
};

complex::complex (double real, double imag) {
  this->re = real; this->im = imag;
}

complex* complex::add (complex* c1, complex* c2) {
  complex* result = new complex(c1->re+c2->re,c1->im+c2->im);
  return result;
}

...
```

- *Statičke funkcije članice* (*static member functions*) su deklarisane uz ključnu reč *static* ispred:
  - jesu članice klase, pripadaju oblasti važenja klase i mogu da pristupaju privatnim članovima te klase (bilo kog objekta)
  - nemaju u sebi implicitno deklarisan pokazivač *this*
  - stoga ne mogu da pristupaju članovima neposredno, bez navođenja objekta kome član pripada
  - zato konceptualno "ne pripadaju" pojedinačnom objektu, već celoj klasi: pružaju uslugu koja se traži od klase, a ne pojedinačnog objekta

Klasa kao realizacija apstraktnog tipa podataka

- Nove operacije koje smo uveli:

```
complex::complex (complex* other) {
  this->copy(other);
}

void complex::copy (complex* other) {
  if (other!=nullptr && other!=this) {
    this->re = other->re;
    this->im = other->im;
  }
}
```

- I onda to koristimo:

```
complex* c1 = new complex(-2.5,6.8);
complex* c2 = new complex(46.5,-34.45);
complex* c3 = complex::add(c1,c2);
complex* c4 = complex::sub(c1,c2);
complex* c5 = new complex(c3);
c3->copy(c4);
…
delete c1; delete c2; delete c3;
delete c4; delete c5;
```

- Problemi:
  - zamorno i nezgrapno (nečitko) manipulisanje objektima i pokazivačima
  - neophodno uništavati objekte (operator *delete*)