TDDI14 Objektorienterad programmering, 8 hp

- **Påbyggnadskurs till TDIU01 och TDIU04**
  - *objektorienterad programmering* – klasser, arv och polymorfi
  - mer om *operatoröverlagring*
  - mer om *mallar* (**template**) – också en objektorienterad konstruktion
- **Föreläsningar** (6 stycken)
  - *klasser* – speciellt initiering, destruering, kopiering och operatoröverlagring
  - *härledda klasser* – arv, polymorfi, dynamisk typkontroll och dynamisk typomvandling
  - *uttryckskträd* – konstruktion av ett sådant är huvuduppgiften i laboration Kalkylatorn
  - *klassmallar* och *funktionsmallar*
  - *iteratorer*
- **Lektioner** (4 stycken)
  - genomgång inför laborationerna
  - övningar
- **Laborationer** (48 timmar)
  - *Listan* – ska göras under kursens första halva
  - *Kalkylatorn* – ska göras under kursens andra halva
- **Examination**
  - laborationer (LAB1), 4 hp
  - datortentamen (DAT1), 4 hp

Objektorienterad programmering

Tre fundamentala begrepp

- **objekt –** modelleras med *klasser* (**class** eller **struct**)

```
class Base { … };
```

- **arv –** nya klasser kan *härledas* från redan befintliga

```
class Derived : public Base { … };
```

  – *basklass – subklass*
  – datamedlemmar och medlemsfunktioner *ärvs*
  – vanligtvis en *specialisering* – subklassen erhåller samma egenskaper som basklassen men ofta görs tillägg

- **polymorfi** – bygger på *dynamisk bindning* av objekt och funktionsanrop
  – *pekare* och *referenser* kan vid olika tillfällen referera till olika typer av objekt inom en klasshierarki

```
Base* p = new Derived;
```

  – en *virtuell funktion* (**virtual**) i en basklass kan *överskuggas* i en subklass – samma signatur men annan definition

```
virtual void print() const;        // deklareras både i Base och Derived
```

  – vad som sker vid anrop av en virtuell funktion beror av typen på objektet ifråga – kan alltså variera

```
p->print(cout);                    // olika definitioner för Base och Derived
```

Snabbrepetition av vad vi redan kan om klasser från TDIU04

- *klassdefinition* – klassen Clock, exception- och funktionsobjektsklasser
- *klassmedlemmar – datamedlemmar* och *medlemsfunktioner*
- *åtkomstspecifikation* för klassmedlemmar – **public** och **private**
- *speciella medlemsfunktioner*
  - *konstruktorer* – någon konstruktor utförs alltid då ett nytt objekt skapas

```
Clock();                                                    Clock c1;
Clock(int h, int m, int s, const string& tz = "");          Clock c2{10, 15, 30};
```

  - *destruktor* – utförs alltid då ett objekt försvinner

```
~Clock();
```

  - *kopieringskonstruktor* – används då ett nytt objekt ska skapas som en kopia av ett redan existerande objekt

```
Clock(const Clock& other);                                  Clock c2(c1);
```

  - *kopieringstilldelningsoperator* – används då ett objekt ska tilldelas ett nytt värde genom att kopiera värdet för ett annat objekt

```
Clock& operator=(const Clock& rhs);                         c1 = c2;
```

  - *flyttkonstruktor* och *flyttilldelningsoperator* som flyttar innehållet från ett objekt till ett annat i stället för att kopiera

```
Clock(Clock&& other);                 Clock& operator=(Clock&& rhs);
```