A 5. Übung

A.1 Überblick

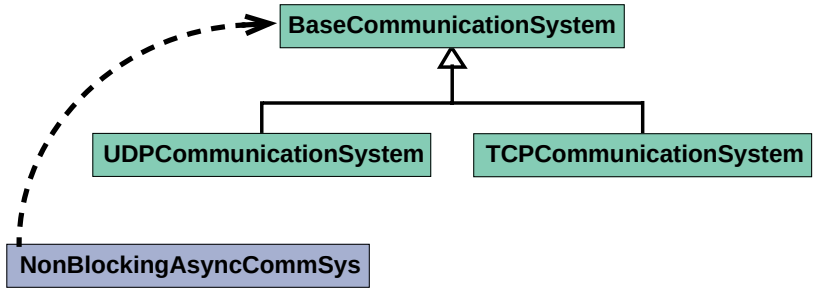
- Aufgabe 3
  - Ableitungshierarchie vs. Templates
- Aufgabe 2
  - Tipps und sonstiges
  - Thread-Klasse
  - Advanced C++

A.2 Ableitungshierarchie vs. Templates

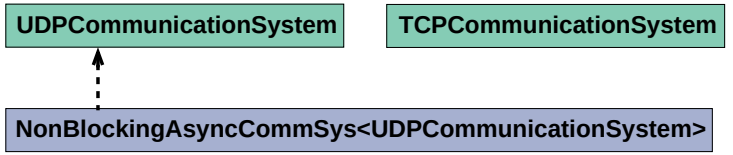
- Aufgabe:
  - Implementierung verschiedener Kommunikationsvarianten
  - basierend auf einem Basiskommunikationssystem
  - welches eine einheitlicher Schnittstelle anbietet
- Die Wahl des Basissystems soll/kann zum Erstellungszeitpunkt getroffen werden.
- Lösungsmöglichkeiten:
  - durch Ableitungshierarchie des Basissystems
  - mit Hilfe von Templates

1 Lösungsmöglichkeiten für Aufgabe 3

- Lösungsweg 1: Ableitungshierarchie



- Lösungsweg 2: mit Hilfe von Templates



2 Ableitungshierarchie

- **UDPCommunicationSystem** von **BaseCommunicationSystem** abgeleitet

```
class UDPCommunicationSystem
                    : public BaseCommunicationSystem {
 public:
   void send(Address dest, Buffer *message);
   void register_send_handler(send_signal_t fn);
   void register_receive_handler(...),
...
}
```

- **NonBlockingAsyncCommSys** verwendet **BaseCommunicationSystem**

```
class NonBlockingAsyncCommSys {
  private:
    BaseCommunicationSystem *com;
  public:
    NonBlockingAsyncCommSys( BaseCommunicationSystem *com)
        : com(com){};

    void send(Address dest, Buffer *message)
       { com->send(dest, message);}
...
}
```

3 Templates

- **UDPCommunicationSystem** hat keine Oberklasse

```
class UDPCommunicationSystem
                        : public BaseCommunicationSystem {
public:
    void send(Address dest, Buffer *message);
    void register_send_handler(send_signal_t fn);
    void register_receive_handler(...),
...
}
```

- **NonBlockingAsyncCommSys** ist durch Template parametrisiert

```
template <class BaseCommunicationSystem>
class NonBlockingAsyncCommSys {
    private:
        BaseCommunicationSystem *com;
    public:
        NonBlockingAsyncCommSys(
               BaseCommunicationSystem *com) : com(com){};

        void send(Address dest, Buffer *message)
            { com->send(dest, message);}
...
}
```


4 Unterschiede

- Vergleich der Grösse (an diesem Beispiel!):
  - Ableitungshierarchie

```
-rw-r----- 1 felser i4staff 8136 May 12 16:53 comm.o
-rw-r----- 1 felser i4staff 6936 May 12 16:53 main.o
```

  - Templates

```
-rw-rw-r-- 1 felser i4staff 5640 May 12 16:53 comm.o
-rw-rw-r-- 1 felser i4staff 7200 May 12 16:53 main.o
```

- Vergleich der Ausführungsgeschwindigkeit
  ($1*10^9$ Aufrufe von **set_receive_signaler**)
  - Ableitungshierarchie: 7.274 s
  - Templates: 5.807 s

A.3 Aufgabe2

1 Thread Klasse: möglich Lösung

- Schnittstelle

```
// Interface

class Runnable{
 public: virtual void run() = 0;
};

class Thread {
    private:
        THREAD_DESC t;
        static void *run_fn (void *);
    public:
        Thread(Runnable *run);
        void join();
};
```


1 Klasse Thread: möglich Lösung

- Implementierung:

```
class Thread {
    private:
        THREAD_DESC t;
        static void *run_fn (void *);
    public:
        Thread(Runnable *run);
        void join();
};
```

```
void* Thread::run_fn(void *run){
  Runnable *r =(Runnable *)run;
  r->run();
  return NULL;
}

Thread::Thread(Runnable *run){
    if (pthread_create(&t, NULL, run_fn, run) != 0){
        perror("Thread::start: pthread_create failed");
        t = 0;
  }
}


void Thread::join(){
    if (pthread_join(t, NULL) != 0)
        perror("thread::join: pthread_join failed");
}
```