1 Fehler-Signalisierung in Konstruktor

Konstruktoren haben keinen *Return-Wert*. Eine Möglichkeit um das Fehlschlagen von Initialisierungs-Schritten zu kommunizieren ist, diese Schritte aus dem Konstruktor in eine `bool Init(...)` Methode auszulagern, welche über ein `return false;` den Fehler kommunizieren kann. Eine weitere Möglichkeit ist, eine Exception aus dem Konstruktor zu werfen.

1.1 Aufgabe

Implementieren Sie eine Klasse `ThrowingData`, welche sie steuern können, ob Sie im Konstruktor eine Exception wirft. Ein zweite Klasse `ThrowingClass` soll drei Attribute von `ThrowingData` enthalten und diese in ihrem eigenen Konstruktor initialisieren. Initialisieren Sie diese so, dass:

- das erste Attribut `a` seinen Namen auf die Konsole printed
- das zweite Attribut `b` seinen Namen printed und eine Exception wirft
- das dritte Attribut `c` gar nie zum Konstruktor kommt.

Der Output eines solchen Programms sollte etwa folgendermassen aussehen:

```
1 ThrowingData ctor: a
2 ThrowingData ctor: b throws
3 ThrowingData dtor: a
4 catch std::exception
```

Beachten Sie, dass das Attribut `a` bereits vollständig initialisiert wurde und deshalb der Destruktor aufgerufen wird. Was fällt ihnen sonst noch auf?

1.2 Lösung

Achten Sie auf folgende Punkte:

- `try`/`catch`-Block ist nötig, damit überhaupt versucht wird aufzuräumen und entsprechend der Destruktor des ersten Attributs aufgerufen wird.
- Verwenden Sie *const-Ref* im `catch`-Statement
- Der Destruktor von `ThrowingClass` wird nie aufgerufen, da die Instanz nicht korrekt konstruiert wurde. Nur die bereits initialisierten Attribute werden korrekt zerstört.
- Zudem wird auch der Body vom `ThrowingClass`-Konstruktor nie aufgerufen, da die Exception bereits in der Initialisierungs-Liste geworfen wird.

throwing_class.h

```cpp
#pragma once

#include <string>

class ThrowingData {
 public:
  explicit ThrowingData(const std::string& name, bool throws = false);
  ~ThrowingData();
 private:
  std::string name_;
};

class ThrowingClass {
 public:
  ThrowingClass();
  ~ThrowingClass();

 private:
  ThrowingData a;
  ThrowingData b;
  ThrowingData c;
};
```

throwing_class.cpp

```cpp
#include "throwing_class.h"

#include <exception>
#include <iostream>

ThrowingData::ThrowingData(const std::string& name, bool throws) : name_(name)
    {
  if (throws) {
    std::cout << "ThrowingData ctor: " << name_ << " throws" << std::endl;
    throw std::exception();
  } else {
    std::cout << "ThrowingData ctor: " << name_ << std::endl;
  }
}
ThrowingData::~ThrowingData() {
  std::cout << "ThrowingData dtor: " << name_ << std::endl;
}

ThrowingClass::ThrowingClass() : a("a"), b("b", true), c("c") {
  std::cout << "ThrowingClass ctor" << std::endl;
}
ThrowingClass::~ThrowingClass() {
  std::cout << "ThrowingClass dtor" << std::endl;
```

```
23 }
```

main.cpp

```cpp
#include <iostream>

#include "throwing_class.h"

int main() {
  try {
    ThrowingClass x;
  } catch (const std::exception&) {
    std::cout << "catch std::exception" << std::endl;
  }
}
```