1 Anmerkungen

- Für die Mittwochsgruppe: Am 21.01. treffen wir uns um 18:00Uhr
- Für die Freitagsgruppe: Am 23.01. findet das Tutorium in Raum 220 statt

2 Vererbung: Begriffe

- Abgeleitete Klasse (*inherited class*):
  - Subklasse (*subclass*)
  - Kindklasse (*child class*)
  - abgeleitete Klasse (*derived class*)
  - Erbe (*heir class*)
- Ableitende Klasse (*inheriting class*):
  - Superklasse (*superclass*)
  - Elternklasse(*parent class*)
  - Basisklasse (*base class*)

3 Vererbung: virtual, pure virtual

virtual_inheritance_pub.cpp

```
/*
 * Licencing information
 *
 * (C) 2015 Michael F. Herbst <info@michael-herbst.com>
 *
 * This program is free software: you can redistribute it and/or
    modify
 * it under the terms of the GNU General Public License as
    published by
 * the Free Software Foundation, either version 3 of the License,
    or
 * (at your option) any later version.
 *
 * It is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * A copy of the GNU General Public License can be found
 * at <http://www.gnu.org/licenses/>.
 *
 */

#include <string>
#include <iostream>

class animal {
public:
  virtual std::string greet() const = 0;
  virtual ~animal() = default;
};

class dog : public animal {
public:
  virtual std::string greet() const {
    return "Wuff!";
        }
  virtual ~dog() = default;
};

class cat : public animal {
public:
  virtual std::string greet() const {
    return "Miao!";
        }
  virtual ~cat() = default;
```

```
};

class petwrapper : public animal {
private:
  const std::string m_name;
  const animal* m_inner;

public:
  petwrapper(const animal& inner, const std::string& name) :
     m_name(name), m_inner(&inner) { }

  virtual std::string greet() const {
    return m_inner->greet() + "(from " + m_name + ")";
  }

  const std::string& get_name() const {
    return m_name;
  }

};

class human : public animal {
private:
  const std::string m_name;

public:
  human(const std::string& name) : m_name(name) {}

  virtual std::string greet() const {
    return "Hello, my name is " + m_name + ".";
  }

  const std::string& get_name() const {
    return m_name;
  }
};

void call_greet(const animal* a) {
  std::cout << a->greet() << std::endl;
}

int main() {
  dog bello_dog, random_dog;
  cat mieze_cat, random_cat;
  human james("James"), mark("Mark");
  petwrapper bello(bello_dog,"Bello"), mieze(mieze_cat,"Mieze");

  call_greet(&bello_dog);
  call_greet(&bello);
```

```
    call_greet(&mieze);
    std::cout << bello.get_name() << std::endl;

    std::cout << std::endl;

    call_greet(&random_dog);
    call_greet(&random_cat);

    std::cout << std::endl;

    call_greet(&james);
    call_greet(&mark);

    return 0;
}
```