Vererbung und RTTI

Aufgabe 6.1 Führen Sie folgendes Programmfragment in Ihrem Kopf aus. Welche Ausgabe würden Sie auf den untersten vier Programmzeilen erwarten?

```
struct Parent {
  virtual void f() { std::cout << "Parent::f" << std::endl; }
  void         g() { std::cout << "Parent::g" << std::endl; }
};
struct Self : public Parent {
  void f() override { std::cout << "Self::f" << std::endl; }
  void g()          { std::cout << "Self::g" << std::endl; }
};
struct Child : public Self {
  void f() override { std::cout << "Child::f" << std::endl; }
  void g()          { std::cout << "Child::g" << std::endl; }
};

Child   c;      c.f();  c.g();
Self&   s =  c; s.f();  s.g();
Parent  p =  c; p.f();  p.g();
Parent* q = &c; q->f(); q->g();
```

Aufgabe 6.2 Seien die drei Typen `Parent`, `Self` und `Child` gleich definiert wie in der vorhergehenden Aufgabe. Welche der folgenden bedingten Anweisungen werden dann ausgeführt und welche nicht? Erklären Sie, welche dieser ausgeführten bedingten Anweisungen problematisch sind.

```
Self ss; Parent* p = &ss;
if(p != nullptr) { std::cout << "A" << std::endl; }

Self* s = static_cast<Self*>(p);
if(s != nullptr) { std::cout << "B" << std::endl; }

Child* c = static_cast<Child*>(p);
if(c != nullptr) { std::cout << "C" << std::endl; }

s = dynamic_cast<Self*>(p);
if(s != nullptr) { std::cout << "D" << std::endl; }

c = dynamic_cast<Child*>(p);
if(c != nullptr) { std::cout << "E" << std::endl; }
```

Aufgabe 6.3 Implementieren Sie die Klassen `Animal`, `Cat`, `Dog` und `Bird`, so dass der weiter unten stehende Code kompiliert. Achten Sie darauf, dass `Animal` eine abstrakte Klasse ist. Stellen Sie desweiteren sicher, dass alle Destruktoren korrekt aufgerufen werden. Folgende Ausgabe soll vom Programm gemacht werden: `Milo says Meow! Wilma says Woof! Wiggles says Woof! Charlie says Chirp!`

```
std::array<std::unique_ptr<Animal>, 4> animals = {
  std::make_unique<Cat>("Milo"),     std::make_unique<Dog>("Wilma"),
  std::make_unique<Dog>("Wiggles"), std::make_unique<Bird>("Charlie")
};

for(auto& animal : animals) {
  animal->say();
}
```