BERGISCHE UNIVERSITÄT
WUPPERTAL
GAUSS-STRASSE 20
42119 WUPPERTAL
TELEFAX (0202) 439-2901
TELEFON (0202) 439-0
WWW www.uni-wuppertal.de

Fachbereich C

MATHEMATIK UND NATURWISSENSCHAFTEN
Fachgruppe Mathematik und Informatik
Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik / Numerik

E-Mail: Hans-Juergen.Buhl@math.uni-wuppertal.de
WWW: http://www.math.uni-wuppertal.de/∼buhl

Programming by Contract

WS 2005/2006 – Übungsblatt 13 (optional)

Ausgabe: 2. Februar 2006

Aufgabe 1. *Contracts bei Vererbung*

Ergänzen Sie den folgenden „Vertrag“ um Klasseninvarianten.

Benutzen Sie dabei immer den expliziten Aufruf der Invariante der Vaterklasse (warum) und schränken Sie alle Attribute in den Invarianten möglichst genau ein.

```
...
#include <iostream>
#include <string>
...
class Fahrzeug
{
protected:
    int AnzahlRaeder;
public:
    virtual bool invariant();
    Fahrzeug(int r=0);
    virtual void info();
};

bool Fahrzeug::invariant()
{
    ...
};

Fahrzeug::Fahrzeug(int r)
    : AnzahlRaeder(r)
{
};
```

```
void Fahrzeug::info()
{
};

...

class Kfz : public Fahrzeug
{
protected:
    double KW;
public:
    virtual bool invariant();
    Kfz(double k=0.0, int r=0);
    virtual void info();
};

bool Kfz::invariant()
{
    ...
};

Kfz::Kfz(double k, int r)
                          : Fahrzeug(r), KW(k)
{
};

void Kfz::info()
{
      ...
};

...

class PKW : public Kfz
{
private:
  int Sitzplaetze;
public:
  virtual bool invariant();
  PKW(int s=0, double k=0.0, int r=0);
  virtual void info();
};

bool PKW::invariant()
{
   ...
};
```

```
PKW::PKW(int s, double k, int r)
                                    : Kfz(k,r), Sitzplaetze(s)
{
};

void PKW::info()
{
      ...
};

...

class LKW : public Kfz
{
private:
  double Ladung;
public:
  virtual bool invariant();
  LKW(double l=0.0, double k=0.0, int r=0);
  virtual void info();
};

bool LKW::invariant()
{
   ...
};

LKW::LKW(double l, double k, int r)
                                        : Kfz(k,r), Ladung(l)
{
};

void LKW::info()
{
      ...
};

...

class Zweirad : public Fahrzeug
{
private:
  string Name;
public:
  virtual bool invariant();
  Zweirad(string n="");
  virtual void info();
};
```

```
bool Zweirad::invariant()
{
   ...
};

Zweirad::Zweirad(string n)
                              : Fahrzeug(2), Name(n)
{
};

void Zweirad::info()
{
      ...
};

int main(void){ // ...
}
```

Aufgabe 2. *Contract Klasse Restklasse*

Schreiben Sie einen Contract für eine generische Klasse `Restklasse` (template-Parameter vom Typ `unsigned int`) mit mindestens den folgenden Methoden:

Default- und Kopierkonstruktor, (abgeschalteter) Wertzuweisungsoperator, grundlegende Observatoren, Operator `==`, Operator `+`.

Aufgabe 3. *rationalNumber*

Klassifizieren Sie die Klasse `rationalNumber` in

http://www.math.uni-wuppertal.de/~buhl/Inf1/ratnum3.cc

gemäß der Einteilung *basic queries, invariant, derived queries, constructors, modificators* und ergänzen Sie sie um geeignete Contracts. Testen Sie.

Aufgabe 4. *guarded postconditions*

Verdeutlichen Sie an einem geschickt gewählten Beispiel, warum die Nachbedingung eines in einer Kindklasse redefinierten Modifikators von der Form

```
...
   ENSURE( ! VorbedingungDerElternklasse || NachbedingungDerElternklasse);
   ENSURE(   VorbedingungDerElternklasse || neueNachbedingung);
...
```

sein sollte. Wie sollte die redefinierte Vorbedingung einer Kindklassenmethode mit derjenigen ihrer Vaterklasse zusammenhängen?

Aufgabe 5. *Contract find()*

Schreiben Sie einen Contract für die Methode `find()` von

http://www.math.uni-wuppertal.de/~buhl/teach/exercises/OOP-WS0102/info3-Erg.pdf (Seite 52f.)