Aufgabenstellung für die Gruppen B und C

1 Durchführung von Zweigüberdeckungstests

Mit dem *Zweigüberdeckungstest* wird überprüft, ob alle Zweige in einem Programm erreicht werden können. Zweige ergeben sich bei Fallunterscheidungen, Schleifen und Unterprogrammaufrufen.

Der *Zweigüberdeckungstest* ist eine *Whitebox*-Test, d.h. zur Vorbereitung und Durchführung des Tests muss der Quellcode des Testobjekts vorliegen. Anhand der aus dem Quellcode abgeleiteten Kontrollflussgraphen spezifiziert man die Testfälle. Jeder Testfall ist eine Kombination von Eingangsdaten, der zu einer Ausführung eines Teils der Zweige führt. Nach Ausführung aller Testfälle sollen alle Zweige erreicht worden sein.

Ihre Aufgaben sind:

- spezifizieren Sie anhand des Quellcodes Kontrollflussgraphen für die zu analysierenden Methoden
- leiten Sie daraus die Testfälle ab
- instrumentieren Sie den Quellcode um festzustellen und zu protokollieren, welche Zweige bei einem Testfall erreicht werden
- erstellen Sie einen Testrahmen, mit dem die Testfälle für den zu testenden (und instrumentierten) Quellcode ausgeführt, die Ergebnisse erfasst und übersichtlich zusammengestellt werden
- erstellen Sie eine Dokumentation, die
  - die Kontrollflussgraphen enthält
  - die Testfälle beschreibt
  - den Testrahmen beschreibt
  - die Testergebnisse zeigt.

2 Anforderungen an die Umsetzung

2.1 Testobjekt, Testfälle und Testrahmen

Das Testobjekt und der Testrahmen verwenden **Python** als Programmiersprache. Die Testfälle sind im JSON-Format zu beschreiben und in Textdateien abzulegen.

Die Testergebnisse sollen in Textdateien mit der Auszeichnungssprache Markdown ausgegeben werden, um sie einfacher in die Dokumentation einbeziehen zu können.

Sie erhalten den Quellcode des Testobjekts zu Beginn des Praktikumstermins.

Das Testobjekt ist ein Python-Modul mit einer Klasse und öffentlichen und privaten Methoden. Die privaten Methoden sind an den führenden Unterstrichen (mindestens einer) erkennbar.

Der Testrahmen instanziiert ein Objekt der im Testobjekt vorhandenen Klasse, liest die Testfalldaten ein und ruft die öffentlichen Methoden in geeigneter Weise auf. Dabei erfolgt die Protokollierung der Tests.

2.2 Instrumentierung

Das Testobjekt muss so instrumentiert werden, dass

- der Ein- und Austritt in Methoden / Prozeduren protokolliert wird
- das Durchlaufen eines Zweigs einer Alternative protokolliert wird
- das Durchlaufen einer Schleife protokolliert wird.

Instrumentieren bedeutet, den Quellcode des Testobjekts um entsprechend Protokollierungsmöglichkeiten zu erweitern.

2.3 Kontrollflussgraphen

2.3.1 Erstellung

Verwenden Sie zur Erstellung der Kontrollflussgraphen das Werkzeug **UMLet**. 'Missbrauchen' Sie die Darstellungsmöglichkeiten, die in der Diagrammform *UML State Machine* angeboten werden, und verwenden Sie Textblöcke, die Sie unter *Generic Text and Alignment* finden:

- geben Sie die Entscheidungspunkte mit dem Darstellungselement *state* an (Verzweigung / Decision enthält keine Texte!)
- geben Sie die Zweige als Kanten zu weiteren Entscheidungspunkten oder Zusammenführungen an
  - notieren Sie in Textblöcken an den Kanten den Quellcode der durchlaufenen Anweisungen
- geben Sie die Zusammenführungen mit dem Darstellungselement *Endzustand* (ausgefüllter Kreis / Punkt) an.

2.3.2 Berücksichtigung von Ausnahmebehandlungen

Mit `try/except` wird die Überwachung von Bereichen beschrieben. Ausnahmen könnten in jeder in einem Bereich enthaltenen Anweisung auftreten. Dann müsste man bei nach jeder Anweisung einen Verzweigung vorsehen. Zur Vereinfachung wird die Darstellung der Diagrammform *UML State Machine* zur Schachtelung von Automaten verwendet:

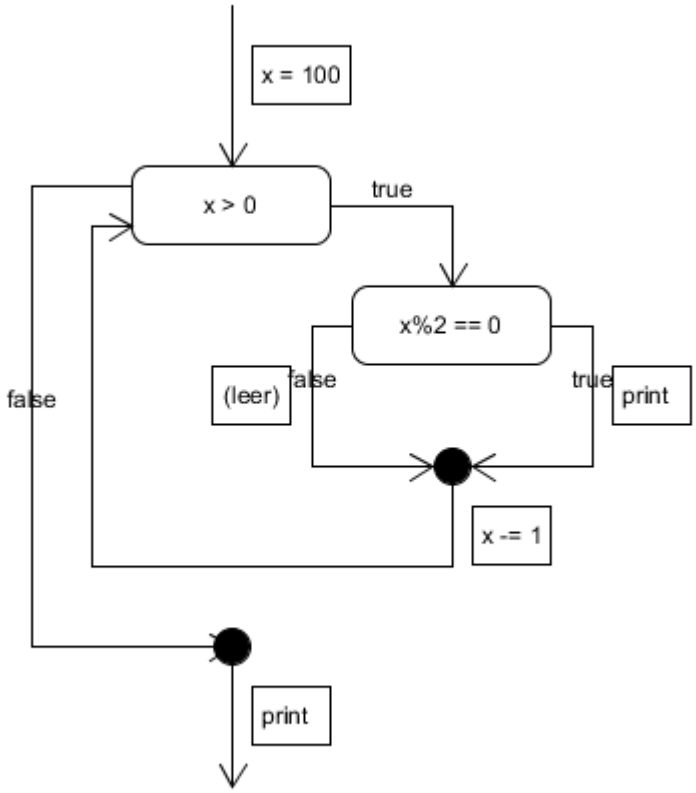
- ein geschützter Bereich wird mit dem Darstellungselement *state with substates* dargestellt
- Kanten von diesem Element aus repräsentieren *alle* Zweige, die durch das Auftreten einer Ausnahme bei einer der Anweisungen im geschützten Bereich möglich sind
- der Inhalt des geschützten Bereichs wird als Verfeinerung dargestellt.

Das ist eine Vereinfachung nur im Rahmen der Praktikumsaufgabe, mit der die Zahl der Zweige reduziert wird!

2.3.3 Beispiel 1

```
x = 100
while x > 0:
   if x%2 == 0:
      print("klappt doch")
   x -= 1
print("das war's")
```

Der Kontrollflussgraph könnte mit Hilfe von UMLet etwa so aussehen:

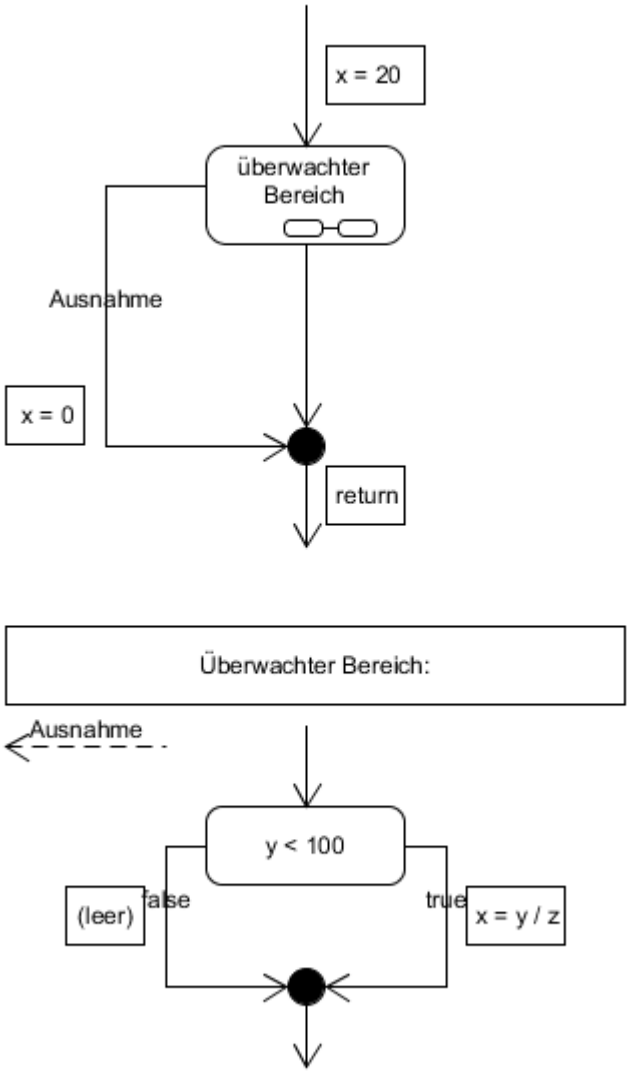


Kontrollflussgraph Beispiel 1

2.3.4 Beispiel 2

```
def calc (y, z):
   x = 20
   try:
      if y < 100:
         x = y / z
   except:
      x = 0
  return x
```

Der Kontrollflussgraph könnte mit Hilfe von UMLet etwa so aussehen:



Kontrollflussgraph Beispiel 2

Der überwachte Bereich wird im unteren Teil als Verfeinerung dargestellt, der gestrichelte Pfeil soll das Auftreten einer Ausnahme im Bereich symbolisieren.

2.4 Markdown-Dokument

Die Dokumentation muss als UTF-8 kodiertes (**ohne** BOM) Plain-Text-Dokument erstellt werden. Als Auszeichnungssprache wird Markdown mit den beim Konverter pandoc zur Verfügung stehenden Erweiterungen, insbesondere Tabellen, verwendet.

Inhalte dürfen nur dann als Abbildungen referenziert werden, wenn es sich um Diagramme handelt.

2.4.1 Erstellung

Zur Vereinfachung der Erzeugung der Dokumentation können Sie die Python-Template-Engine **mako** nutzen:

- die Testergebnisse werden als Datenstruktur abgespeichert (transient oder persistent, z.B. im JSON-Format)
- es wird ein mako-Template benutzt

- um die manuell erstellten Bestandteile der Dokumentation einzubinden
- um die Testergebnisse einzubinden.

2.4.2 Meta-Angaben

Sehen Sie einleitend im Dokument einen Abschnitt mit Meta-Angaben vor, der im YAML-Format angegeben wird:

```
---
author:
   name1: IhrName1
   name2: IhrName2
   name3: IhrName3
title:
   main: Hauptüberschrift
   sub1: Untertitel1
   sub2: Untertitel2
revision:
   doc: Dokumentname
   level: Änderungszähler
   date: Gültigkeitdatum
lang: de
---
```

Es sind folgende Angaben vorgesehen:

- Geben Sie bei `IhrName1` etc. Ihre Namen und Vornamen an; wenn Sie weniger als 3 Namen angeben wollen, lassen Sie den entsprechenden Wert leer
- Geben Sie die Hauptüberschrift (z.B. "Testdokumentation") sowie weitere Untertitel (z.B. "SWE-Praktikum WS 2014/2015") an
- Geben Sie den (Datei-)Namen des Dokuments an sowie einen Änderungszähler (bei 0 beginnend), der einzelne Versionen kennzeichnet
- Geben Sie das Gültigkeitsdatum des Dokuments an
- Spitze Klammern < oder > dürfen in den Werten nicht auftreten
- mit `cssextra: CSSDateiName` können Sie eine eigene CSS-Datei berücksichtigen; geben Sie diesen Angabe z.B. unterhalb der Zeile `lang: de` an.

2.4.3 Namenskonvention für die Dokumentation

Bezeichnen Sie die Dokumention nach folgendem Schema: `swe_p3_testdoc.<revision>.md`. Inkrementieren Sie die Revision-Nummer beim Dateinamen wie auch in den Meta-Angaben konsistent, wenn Sie das Dokument überarbeiten.

2.4.4 Erstellung von PDF-Varianten

Erstellen Sie mit den Werkzeugen `pandoc` und `prince` (siehe http://www.princexml.com) für die Dokumention eine PDF-Variante.

Verwenden Sie dazu die Vorlage-Datei und die Style-Sheet-Dateien, die Ihnen mit der Aufgabenstellung 2 zur Verfügung gestellt wurden (`hdoc_pdf.css.inc` und `hdoc_syntax.css.inc`).

Schritt 1: Umwandlung in HTML-Datei

```
pandoc -f markdown -t html -o %1.pdf.html -s -N --number-offset=%2 --template=hdoc.tpl -H hdoc_pdf.css.inc
       -H  hdoc_syntax.css.inc --highlight-style pygments %1.md
```

Schritt 2: Erzeugung PDF-Datei

```
<Pfad-zu-Prince-Executable>\prince %1.pdf.html -o %1.pdf
```