Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Institut für Programmstrukturen und Datenorganisation (IPD)

Prof. Dr. Walter F. Tichy

Vorlesung Softwaretechnik 1

Übungsblatt 6
Sommersemester 2009

Ausgabe: 02.07.2009

Abgabe: 16.07.2009, 11:00 Uhr

Testen und Zustandsautomaten

Aufgabe 1: Kontrollflussorientierte Testverfahren (10 Punkte)

Gegeben seien die folgenden Methoden:

```
public static int getNumberOfDays(int month, int year) {
  int days = -1;
  if (year >= 1) {
    if (month == 1 || month == 3 || month == 5 || month == 7 ||
        month == 8 || month == 10 || month == 12) {
      days = 31;
    } else if (month == 4 || month == 6 || month == 9 || month == 11) {
      days = 30;
    } else if (month == 2) {
      if (isLeapYear(year)) {
        days = 29;
      } else {
        days = 28;
      }
    }
  }
  return days;
}

public static boolean isLeapYear(int year) {
  return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
}
```

a) Setzen Sie die Methode `getNumberOfDays(…)` zunächst in die in der Vorlesung präsentierte Zwischensprache um. Erstellen Sie aus der Zwischensprache den Kontrollflussgraphen der Methode `getNumberOfDays(…)`. Wenden Sie dabei das in der Vorlesung vorgestellte Verfahren an. Betrachten Sie den Methodenaufruf `isLeapYear(…)` dabei zunächst als schwarzen Kasten, der entweder „wahr“ oder „falsch“ liefert. (6 Punkte)

b) Geben Sie für die Methode `getNumberOfDays(…)` unter Berücksichtigung der Methode `isLeapYear(…)` eine minimale Testfallmenge an, welche die Anweisungsüberdeckung erfüllt sowie eine minimale Testfallmenge, welche die Zweigüberdeckung erfüllt. (2 Punkte)

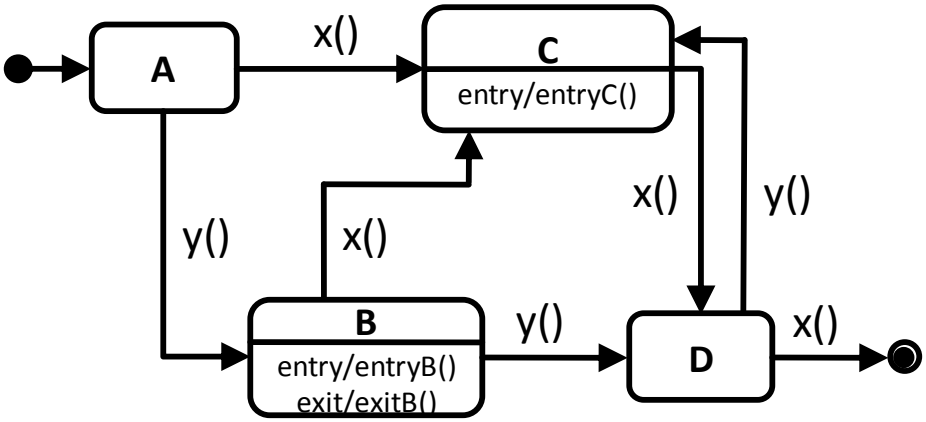
c) Betrachten Sie nun den booleschen Ausdruck in der Methode `isLeapYear(…)`. Geben Sie eine minimale Testfallmenge für die einfache Bedingungsüberdeckung an. Geben Sie für jede Teilbedingung an, ob sie wahr, falsch oder nicht ausgewertet wird. (2 Punkte)

Aufgabe 2: Komplexe Zahlen (8 Punkte)

In der Datei Complex.zip finden Sie eine JAR-Datei, die eine Bibliotheksklasse zum Rechnen mit komplexen Zahlen enthält sowie die dazugehörige Dokumentation in Form von Javadoc. Drei der öffentlichen Methoden und Konstruktoren zeigen nicht das in der Dokumentation beschriebene Verhalten, sind also fehlerhaft implementiert. Erstellen Sie eine JUnit-Testklasse, welche alle öffentlichen Methoden und Konstruktoren testet. Verwenden Sie JUnit 4. Nennen Sie die Methoden und Konstruktoren, die nicht korrekt implementiert sind. Bestimmen Sie den zugrundeliegenden Fehler möglichst genau. D.h. „Methode `xyz()` funktioniert nicht“ wäre keine ausreichende Antwort.

Aufgabe 3: Abbildung von UML-Modellen auf Code (11 Punkte)

Gegeben sei folgender UML-Zustandsautomat:



a) Realisieren Sie den UML-Zustandsautomaten als eingebettete explizite Speicherung mit Java-Code. (4,5 Punkte)

b) Realisieren Sie den UML-Zustandsautomaten als ausgelagerte explizite Speicherung mit Java-Code. (3,5 Punkte)

Bitte umblättern.

c) Gegeben sei folgende Implementierung des UML-Zustandsautomaten (ohne entry/exit-Aktionen) mittels einer Zustandstabelle. Vergleichen Sie diese Implementierung mit den Implementierungen aus den Aufgabenteilen a) und b). Wägen Sie die Vor- und Nachteile der jeweiligen Implementierung gegenüber den anderen Implementierungen ab. Geben Sie an, wie der unten gegebene Code zu erweitern ist, damit die entry- und exit-Aktionen aus dem UML-Zustandsautomaten ebenfalls realisiert sind. (3 Punkte)

```
public class StateMachine {
  private static final int A = 0;
  private static final int B = 1;
  private static final int C = 2;
  private static final int D = 3;
  private static final int END = 4;
  private static final int ERROR = -1;
  private static final int X = 0;
  private static final int Y = 1;

  int[][] stateTable = new int[][] {
      { C, C, D,     END, ERROR },
      { B, D, ERROR, C,   ERROR } };

  private int s = A;

  public void x() {
    makeTransition(X);
  }

  public void y() {
    makeTransition(Y);
  }

  private void makeTransition(int i) {
    s = stateTable[i][s];
    if (s == ERROR) {
      throw new IllegalStateException();
    }
  }
}
```

Hinweis: **Bitte lesen Sie sich den folgenden Text aufmerksam durch. Missachtung der Anweisungen wird Sie Punkte kosten!**

Die Lösung geben Sie bitte nicht nur als E-Mail sondern auch über den Praktomaten ab. Die Anmeldung am Praktomaten erfolgt unter folgender Adresse:

https://swt1.ipd.uka.de/

Die Oberfläche zum Einreichen der Lösungen finden Sie unter

https://swt1.ipd.uka.de/praktomat/

Den Quelltext Ihres Programmes drucken Sie bitte aus und geben ihn zusammen mit dem Übungsblatt ab. Zusätzlich müssen Sie sowohl den Quelltext als auch die Java-Anwendung an folgende E-Post-Adresse senden:

swt1@ipd.uka.de

Als Betreff geben Sie bitte **[Uebungsblatt 6] <TUTNR> <MATRNR> <VORNAME> <NACHNAME>** an.

Hängen Sie den Java-Quelltext als ZIP-Archiv an die E-Mail an. Der Name des ZIP-Archivs muss folgendes Format haben: **<TUTNR>_<MATRNR>_<VORNAME>_<NACHNAME>.zip**. Das ZIP-Archiv darf nur den Java-Quelltext (d.h. Dateien mit der Endung .java) und evtl. benötigte Ressourcen (wie bspw. Icons) enthalten. Sofern Sie mit Paketen gearbeitet haben, muss diese Ordnerstruktur ebenfalls in Ihrem ZIP-Archiv vorhanden sein.

Verwenden Sie in Ihrem Java-Quelltext nur ASCII-Zeichen und vermeiden Sie IDE-spezifische Java-Annotationen. Die lauffähige Java-Anwendung hängen Sie als **ausführbares JAR-Archiv** zusätzlich zum ZIP-Archiv an die E-Mail an. Es werden nur Java-Anwendungen akzeptiert, welche als ausführbares JAR-Archiv an die E-Mail angehängt sind! Das JAR-Archiv muss als separater Anhang gesendet werden. Der Name des JAR-Archivs muss wie folgt lauten: **<TUTNR>_<MATRNR>_<VORNAME>_<NACHNAME>.jar**

Beispiel: Sie heißen Hans Müller, gehen in das Tutorium 7 und Ihre Matrikelnummer ist 123456. Ihr Betreff sieht dann wie folgt aus:

[Uebungsblatt 6] 07 123456 Hans Mueller

Die Anhänge heißen dann:

07_123456_Hans_Mueller.zip / 07_123456_Hans_Mueller.jar

Die E-Mail mit beiden Anhängen muss bis zum offiziellen Abgabetermin bei uns eingegangen sein. Vergewissern Sie sich, dass Sie alle Code-Dateien, die zur Übersetzung der Java-Anwendung notwendig sind, mitschicken und der Quelltext übersetzt werden kann. Prüfen Sie, dass die Java-Anwendung aus dem JAR-Archiv lauffähig ist. Andere Dateiformate werden nicht akzeptiert.

Bei Missachtung der obigen Anforderungen oder Teilen davon wird die gesamte Aufgabe mit 0 Punkte bewertet.