Übungsblatt 2

Unvollständige Zustandsaufzeichnung

Gegenstand dieses Übungsblattes ist das Thema *Rückwärtsbehebung* durch *unvollständige Zustandsaufzeichnung* in Form der Implementierung eines verteilten Systems bestehend aus einem Rechenprozess (Knoten A), einem Absoluttest (Knoten B) und einen Rücksetzpunktverwalter (Knoten C). Die Fehlerinjektion soll sich auf Inhaltsverfälschung (Fehlerart 6) beschränken und auch nur die Nachrichten von Knoten A an Knoten B betreffen. Die Nachrichten von Knoten A an Knoten C sowie den Absoluttest B als auch der Rücksetzpunktverwalter C seien stets fehlerfrei.

In Knoten A wird ein abstrakter Rechenprozess angenommen, dessen Datenvariablen aus einem eindimensionalen int-Array a mit 8 Elementen bestehen. Die initiale Belegung sei

```
a = {10, 10, 10, 10, 10, 10, 10, 10}.
```

Die Verarbeitung erfolgt schrittweise durch eine vorgegebene Methode

```
boolean[] schritt(int[] a),
```

die im Programmgerüst bereitgestellt wird. Sie verändert bei jedem Aufruf das Array a scheinbar zufällig, aber deterministisch. Der erste Eintrag des Arrays `a[0]` kann als eine Art Befehlszeiger aufgefasst werden, der bei jedem fehlerfreien Rechenschritt um 1 erhöht wird. Weiterhin simuliert `schritt` auch eine kurze Bearbeitungsdauer. Der von `schritt` zurückgelieferte Wert ist ein 8-elementiges boolesches Array, das angibt, welche Elemente von a in dem durchgeführten Rechenschritt verändert worden sind.

Nach Ausführung eines Rechenschritts sendet Knoten A eine Ergebnis-Nachricht (Typ 'e'), die alle 8 Elemente des Vektors a enthält (durch Leerzeichen getrennt) an den Absoluttest B, der diese auf Korrektheit prüft. Dazu wird die Methode

```
boolean absoluttest(int[] a)
```

benutzt, die ebenfalls im Programmgerüst bereitgestellt wird. Bei bestandenem Absoluttest (Rückgabewert `true`) sendet Knoten B alle 8 Elemente des Vektors a an den Knoten A zurück (Nachrichtentyp 'z'). Knoten A führt mit diesen Werten dann den nächsten Rechenschritt aus. Insgesamt führt Knoten A stets genau 80 Rechenschritte auf diese Weise aus. Insbesondere läuft `a[0]` dabei von 10 bis 90.

In Knoten B soll weiterhin eine int-Variable `f` vereinbart werden, die den erreichten Fortschritt angibt. Dazu wird unmittelbar vor jedem Absoluttest

```
f = fortschritt(f, a);
```

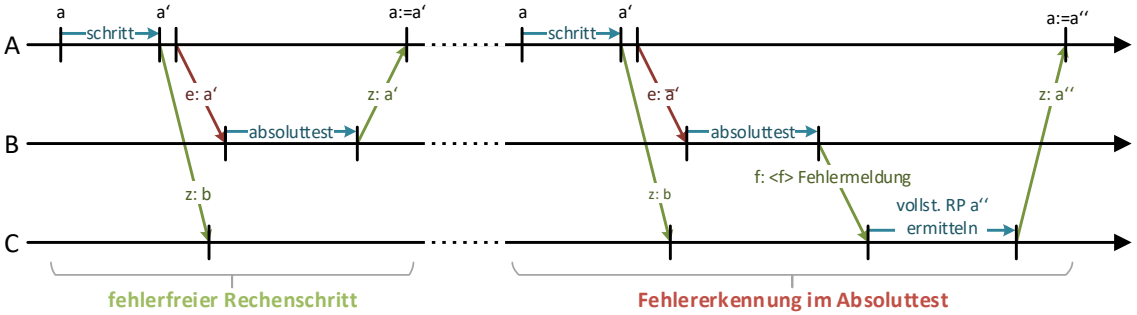
aufgerufen. Damit wird angezeigt, dass das fiktive Anwendungsprogramm mit den Daten des Arrays a einen Fortschritt erreicht (der beim Zurücksetzen wieder teilweise reduziert wird). Die Methode Fortschritt wird ebenfalls im Programmgerüst zur Verfügung gestellt.

Wenn der Absoluttest einen Fehler erkennt, sendet er keine Nachricht an A, sondern eine Fehlermeldung (Nachrichtentyp 'f') an den Rücksetzpunktverwalter C mit dem Inhalt `"<f> + Fehlermeldung"`, wobei <f> der Wert der Fortschrittsvariablen ist. Als Antwort darauf sendet der Rücksetzpunktverwalter an den Knoten A einen vollständigen Rücksetzpunkt, bestehend aus 8 Zahlen (Nachrichtentyp 'z'), die A in das Array a übernimmt. Anmerkung: Letztlich übernimmt Knoten A vor jedem Rechenschritt die 8 Zahlen für das Array a aus der eintreffenden Nachricht, gleichgültig ob diese von Knoten B oder Knoten C kommt.

Rücksetzpunkte werden gemäß dem Verfahren „unvollständige Zustandsaufzeichnung“ dadurch erstellt, dass Knoten A nach jedem Rechenschritt die Elemente des Arrays a, die sich verändert haben, in einer

Rücksetzpunkt-Nachricht (Nachrichtentyp 'z' ohne Fehlerinjektion) an den Rücksetzpunktverwalter C sendet – und zwar unmittelbar bevor er das Array a in einer Ergebnis-Nachricht an den Absoluttest B sendet. In der Rücksetzpunkt-Nachricht wird eine kommaseparierte Liste verwendet, wobei jedes Element aus dem Index der veränderten Zahl und der veränderten Zahl selbst besteht, welche wiederum durch Leerzeichen getrennt werden.

Nachfolgende Abbildung skizziert das Verhalten im fehlerfreien und fehlerhaften Fall.



Ist beispielsweise in obiger Skizze

```
a = {14, 16, 14, 12, 15, 14, 13, 11}
```

und nach der Ausführung der Methode `schritt(a)` ist

```
a‘ = {15, 18, 14, 13, 17, 15, 13, 11},
```

so ist folglich der Rückgabevektor der Methode `schritt(a)` das boolsche Array

```
{true, true, false, true, true, true, false, false}
```

und A sendet an C den Nachrichteninhalt `b = 0 15, 1 18, 3 13, 4 17, 5 15`, d.h.

```
z: 0 15, 1 18, 3 13, 4 17, 5 15.
```

Knoten C speichert die ihm zugesandten (meist unvollständigen) Rücksetzpunkte in einem zweidimensionalen Array RP. Der erste Index von RP bezeichnet den (meist unvollständigen) Rücksetzpunkt, wobei ältere Rücksetzpunkt einen kleineren Index aufweisen. Insgesamt werden bis zu 50 (meist unvollständige) RP gespeichert. Der zweite Index von RP bezeichnet das Element von a. Bei unvollständigen Rücksetzpunkten wird für alle nicht verfügbaren Elemente von a der Wert –1 eingetragen. Wenn der Rücksetzpunkt-Speicher RP voll ist und ein weiterer Rücksetzpunkt gespeichert werden soll, muss der mit Index 1 abgespeicherte Rücksetzpunkt `RP[1][...]` aus RP gelöscht werden. Alle nachfolgenden Rücksetzpunkte `RP[x][...]` rücken dann um eine Position nach vorn. Der älteste Rücksetzpunkt `RP[0][...]` enthält die Anfangsbelegung `{10, 10, 10, 10, 10, 10, 10, 10}` von a und wird niemals gelöscht.

Wenn der Absoluttest durch eine Fehlermeldung (Nachrichtentyp 'f') eine Rückwärtsbehebung verlangt, dann soll aus den unvollständigen Rücksetzpunkten ein vollständiger zusammengesetzt werden. Außerdem ist die Rücksetzweite in geeigneter Weise zu bestimmen. Dazu soll der mit der Fehlermeldung übermittelte Fortschritt ausgewertet werden.

Knoten A führt so lange Rechenschritte aus, bis ihm nach 80 Rechenschritten die Endwerte `{90, 46, 19, 13, 46, 23, 12, 16}` des Arrays a zugesandt werden. Danach terminiert A mit

```
return s + " Rechenschritte durchgefuehrt.";
```

wobei s die Anzahl der tatsächlich durchgeführten Rechenschritte ist. Bedingt durch das Zurücksetzen ist sie oftmals größer als 80. Die Knoten B und C verwenden für ihre Empfangsoperationen eine Zeitschranke, so dass sie ebenfalls terminieren, wenn sie von A keine Nachricht mehr erhalten. Knoten B terminiert mit

```
return f + " wurde erreicht.";
```

wobei f der letzte Wert der Fortschrittsvariablen f ist. Knoten C terminiert mit

```
return anzZurueck + " mal zurueckgesetzt.";
```

wobei `anzZurueck` die Anzahl der durchgeführten Rücksetzoperationen ist. Die result-Methode soll ebenfalls heruntergeladen werden. Sie zeigt als Summary an, wie oft beim letzten Lauf, in der Summe aller Läufe und im Durchschnitt bei allen Läufen zurückgesetzt wurde. Außerdem liefert sie als Result:

	Beschreibung
5	Programmausführung war nicht sinnvoll
4	Die Anzahl der Schritte war kleiner als der erzielte Fortschritt (eine Inkonsistenz)
3	Die Anzahl der Schritte war inkonsistent zu der Anzahl der Rücksetzoperationen
2	Der erreichte Fortschritt war gering ($<$ 60 Schritte) oder überstieg 81 Schritte
1	Programm terminiert erfolgreich und Rücksetzoperationen fanden statt
0	Programm terminiert erfolgreich und es wurde nie zurückgesetzt

Aufgabe 1: Rücksetzpunktverwaltung

Beschreiben Sie präzise das von Ihnen gewählte Verfahren, wie der Rücksetzpunktverwalter (Knoten C) vorgeht, wenn einen Rücksetzpunkt erhält (von A) und wenn er eine Fehlermeldung erhält (von B).

Aufgabe 2: Implementierung des Systems

Implementieren Sie in dem bereitgestellten Gerüst die Funktionalitäten der drei Systemkomponenten gemäß der zuvor Beschriebenen Systemspezifikation.

Aufgabe 3: Fehlerinjektionsexperimente

Führen Sie Fehlerinjektionsexperimente für den Fehler "corrupt contents" für

$$p \in \{0, 0.01, 0.02, 0.05, 0.1\}$$

mit jeweils mindestens 200 Wiederholungen durch. Dokumentieren Sie tabellarisch folgende Statistiken:

- die eingestellte Fehlerwahrscheinlichkeit p,
- die Anzahl der Experimente mit Result j, $j \in \{0, ..., 5\}$,
- die durchschnittliche Anzahl der Rücksetzoperationen.

Wichtige Hinweise:

1. Die Aufgaben sind bis zum 29.06.2017 zu bearbeiten und bei Moodle bis 10:00 Uhr als Gruppe einzureichen.
2. Es wird in Programmgerüst mit dem Namen `Gxx_A2.java` bereitgestellt. Ersetzen Sie hierbei vor Abgabe `xx` zweistellig durch Ihre Gruppennummer. Analog benennen Sie bitte die Dokumentation der Rücksetzpunktverwaltung und Ihrer Ergebnisse in der Form `Gxx_A2.pdf`.
3. Ab dem 29.06.2016 werden Ihre Ergebnisse von Ihnen in den Übungen präsentiert. Die Reihenfolge der Präsentationen wird willkürlich ausgewählt.
4. Sie dürfen bei Ihrer Implementierung den kompletten Umfang der Sprache JAVA benutzen.
5. Es wird ausdrücklich empfohlen, die von SoFT bereitgestellten Methoden zu verwenden. Schauen Sie dafür zwingend in die Dokumentation von SoFT. Dies erspart Ihnen im Allgemeinen bereits vorhandene Funktionen neu zu implementieren.
6. Achten Sie bei Ihrer Implementierung darauf, im Rahmen des bereitgestellten Konstrukts zu bleiben und keine neuen Dateien zu erzeugen. Vermeiden Sie ebenfalls die Benutzung von Threads. Unerlassen Sie weiterhin ein Rafactoring des bereitgestellten Gerüsts.