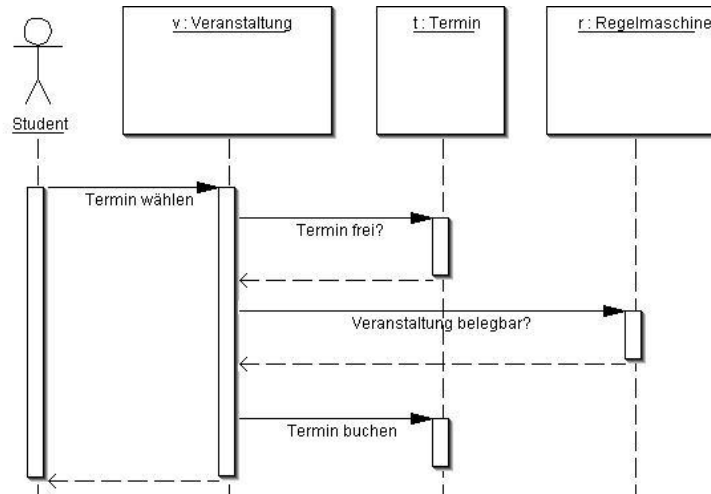


Aufgabe 11-1: (Prüfungsverwaltungssystem verifizieren)

Das nun als implementiert angenommene Prüfungsverwaltungssystem soll getestet werden.

1. Welche UML-Diagramme sind im Allgemeinen hilfreich zum Testen einer daraus entstandenen Software?
2. Folgendes Sequenzdiagramm ist eine mögliche Lösung für Aufgabe 5-2:



Das dazugehörige Szenario sieht verkürzt wie folgt aus:

Vorbedingung: Student ist angemeldet und hat eine Veranstaltung ausgewählt

1. Student wählt einen Termin (wenn es mehrere gibt, z.B. bei Übungen) für die Veranstaltung.
 2. System prüft, ob der Termin noch einen freien Platz hat.
 3. System prüft, ob die Veranstaltung für den Studenten belegbar ist.
 4. System bucht den Termin und gibt positive Rückmeldung.
- Effekt: Vorbedingung + Termin ist für Student gebucht und Termin hat einen freien Platz weniger.

Folgende Ausnahmefälle seien in dem entsprechenden Anwendungsfall angegeben:

- 2a. Der Termin ist schon komplett voll.
- 3a. Die Veranstaltung passt nicht zum Studienverlauf des Studenten.
- 3b. Wenn der Student für das Semester zu viele Leistungspunkte bucht, kommt eine Warnung.

Entwerfen Sie aus diesen Angaben Testfälle.

3. Wir wollen mittels eines Lasttests prüfen, ob das System auch bei vielen gleichzeitigen Zugriffen noch schnell genug reagiert. Nehmen Sie dazu an, dass das System mittels web-basierter Client-Server-Architektur realisiert wurde (ähnlich KVV).

- a. Schätzen Sie: Welche Belastung ist maximal zu erwarten?
- b. Formulieren Sie aufgrund Ihrer Schätzung aus a. ein konkretes Kriterium, das das System erfüllen soll?
- c. Wie würden Sie diesen Lasttest durchführen? Wie realitätsnah wäre Ihr Ergebnis?

Aufgabe 11-2: (Durchsichten)

Auf der nächsten Seite finden Sie eine Checkliste zur Durchsicht von Javacode.

1. Gegen welche der dort aufgeführten Prüfpunkte (checks) verstoßen Sie selbst gelegentlich bei Ihrer Programmierarbeit?
2. Nennen Sie mindestens einen Punkt, der nicht automatisiert geprüft werden kann.

3. Nennen Sie mindestens einen Punkt, der Defekte aufdeckt, deren potenziellen Auswirkungen (= Versagensfälle) durch Testen schwer zu entdecken sind.

4. Welche Vorteile haben Durchsichten im Allgemeinen im Vergleich zu dynamischen Tests?

Aufgabe 11-3: (Goto considered harmful)

1968 hat Edsger Dijkstra eine kurze, aber berühmt gewordene Notiz veröffentlicht über die Gefährlichkeit von Sprunganweisungen in Programmiersprachen: „Goto considered harmful“. Unter <http://www.acm.org/classics/oct95/> finden Sie den Artikel online. Lesen Sie ihn und geben Sie die Argumentation in groben Zügen wieder. Stimmen Sie ihr zu und finden Sie sie überzeugend?

Checkliste zur Durchsicht von Java-Programmen

Arithmetik

1. Sind Überlauf oder Unterlauf während der Berechnung möglich?
2. Für jeden Ausdruck mit mehr als einem Operator: Sind die Annahmen über die Ausführungsreihenfolge korrekt?
3. Wurden Klammern eingesetzt um Mehrdeutigkeit zu vermeiden?
4. Ist „Division by Zero“ möglich?
5. Geht man fälschlicherweise davon aus, dass Fließkomma-Arithmetik genau ist?

Schleifen

6. Werden vor einer Schleife alle beteiligten Variablen richtig initialisiert oder enthalten sie alle bereits einen gültigen Wert?
7. Werden alle Schleifen auf jeden Fall beendet?

Abzweigungen

8. Hat jedes switch-Statement einen default Fall?
9. Sind fehlende break Statements in switch'es korrekt und besonders kommentiert?
10. Sind alle else-Zweige richtig behandelt? Wenn einer if-Bedingung der else-Zweig fehlt, wird der Fall richtig behandelt, wenn die if-Bedingung nicht erfüllt wird?

Datenfluss

11. Kann eine Variable unter Umständen null sein und wird dieser Fall abgefangen?
12. Werden Parameterwerte auf gültige Wertebereiche (entsprechend der Vorbedingungen) überprüft?
13. Müssen Objekte mit equals() oder direkt mit == verglichen werden?
14. Kann eine Typanpassung (casting) fehlschlagen?

Arrays

15. Ist das Indexieren von Arrays außerhalb des gültigen Bereichs möglich?

Funktionsaufrufe

16. Reagiert der Aufrufer einer Methode auf alle möglichen Werte, die zurückgegeben werden können, inkl. Ausnahmen (Exceptions)?
17. Erfüllt der Aufrufer einer Methode die Voraussetzungen für die Parameter der Methode?
18. Kann ein Stack-Overflow bei rekursiven Funktionen auftreten?

Multithreading

19. Sind alle Zugriffe von mehreren Threads auf dieselben Variablen synchronisiert? Muss die Variable als volatile deklariert werden?
20. Besteht eine Verklemmungsgefahr?
21. Werden Threads, die mit wait() warten, irgendwann mal aufgeweckt?
22. Werden InterruptedExceptions behandelt?
23. Erfolgt der Zugriff auf eine Variable in zwei getrennten synchronized()-Blöcke vom selben Thread und ist das korrekt?
24. Wird ein Objekt von einem Thread freigegeben (auf null gesetzt) und von einem anderen zugegriffen?

Ausnahmebehandlung

25. Wird der Kontrollfluss richtig fortgesetzt, wenn eine Ausnahme auftritt oder abgefangen wird?
26. Werden eventuelle Ausnahmen beim Zugriff auf externe Ressourcen abgefangen?
27. Werden alle Ausnahmen, die in den von dieser Methode aufgerufenen Methoden aufgeworfen werden können, abgefangen?

Externe Ressourcen

28. Werden Eingabe-, Ausgabe-Ströme und externe Ressourcen (Datenbank-Connections, Sockets, etc) wieder geschlossen?
29. Werden gepufferte Daten ge'flush't?
30. Werden Ausnahmen (Exceptions) bei Ein- und Ausgabe richtig behandelt?