Testen und Debugging

- Testklassen, Unit Tests
- Blackbox Test, Whitebox Test
- Regressionstesten
- Zusicherungen mit `assert`
- Debugger

Blackbox Test

Blackbox Testen:

Das gesamte Programm wird getestet ohne die Funktionsweise oder Implementierung zu berücksichtigen. Ein Unit Test überprüft die Funktionsweise einer oder mehrerer zusammenhängender Methoden.

Ein Blackbox Test steht typischerweise am Ende einer Entwicklung vor der Auslieferung an den Kunden.

Blackbox Tests sollten gerade auch unerwartete Eingaben und Randfälle berücksichtigen.

Vorteil des Blackbox Tests: er zeigt, ob das Programm wirklich als ganzes funktioniert.

Nachteile des Blackbox Tests: Teile des Programms könnten von den Testeingaben überhaut nicht oder nur in einfachen Fällen berührt werden.

Man kann Blackbox Tests erst durchführen, wenn die Anwendung ganz fertig ist.

Whitebox Test

Ein *Whitebox Test* (auch Glassbox Test) berücksichtigt die Programmstruktur.

Insbesondere können beim Whitebox Test einzelne Methoden getestet werden noch bevor ihr Anwendungskontext fertig ist.

Die Testfälle sollten den gewünschten Anwendungsbereich überstreichen und möglichst jeden Zweig des Methodenrumpfes tatsächlich erreichen.

Es gibt automatisierte Methoden zur Erzeugung solcher Testbeispiele.

Unit Test

Ein Test einer einzelnen Methode oder von eng kooperierenden Methoden heißt *Unit Test*.

Der Unit Test ist die wichtigste Testform und sollte für jede nichttriviale Methode durchgeführt werden.

Es wird empfohlen Unit Tests vor der Implementierung (aber nach der Spezifikation) einer Methode zu entwerfen.

Um Unit Tests effizient und reproduzierbar durchzuführen bieten sich Unterstützungen wie JUnit (www.junit.org) an.

Test Harness und Test Suite

Um eine Methode oder eine Klasse zu testen (im Unit Test) muss ein geeigneter Kontext aus Objekten bereitgestellt werden und ein Hauptprogramm geschrieben werden, das die zu testende Methode in diesem Kontext aufruft und das Ergebnis auswertet.

Solch ein Programm heißt *test harness* (*harness*=Geschirr (für Pferde, etc.)).

Der Aufrufkontext wird auch als *fixture* (= Vorrichtung) bezeichnet.

Oft werden mehrere Tests in einer *test suite* zusammengefasst, die mit einem einzigen Befehl ausgeführt werden kann.

Regressionstesten

Es bietet sich an, dieselben Tests nach jeder Programmänderung immer wieder durchzuführen, damit nicht früher behobene Fehler wieder eingeführt werden.

Tritt ein früher behobener Fehler erneut auf, so spricht man von *cycling*.

Das wiederholte Durchführen von Tests heißt *Regressionstesten*. (*regredi*=zurückkehren).

Dies wird durch systematische Verwaltung in Test Suites und automatisches Ablaufen der Tests erleichtert.

Basieren die Tests auf manuell eingefügten Kontrollausgaben, die man nach Beheben von Fehlern entfernt, so wird man sich kaum die Mühe eines Regressionstests machen.

Orakel

Um die Korrektheit von Testergebnissen zu bestätigen muss manchmal das erwartete Ergebnis mit einer anderen (ineffizienten) Methode berechnet werden.

Diese wird dann als *Orakel* bezeichnet.

Beispiel: Testen einer Methode zum schnellen Potenzieren mithilfe einer Methode, die die übliche Definition verwendet.

Es kann auch sein, dass man das Testergebnis nicht gut mithilfe eines Orakels berechnen kann, wohl aber dessen Korrektheit bestätigen.

Beispiel: Zu testen sei Methode m zur Matrizenmultiplikation. Man wähle Matrizen A, B und Testvektor v und prüfe, ob $m(A, B)v = A(Bv)$.

Zusicherungen

Seit Java 1.4 gibt es das *Zusicherungsstatement*.

Es hat die Form `assert` e wobei e ein Ausdruck vom Typ bool ist.

Man muss bei seiner Verwendung mit der Option `-source 1.4` kompilieren.

Führt man das Programm mit der Option `-enableassertions` aus, so wird bei jedem Zusicherungsstatement der Ausdruck e ausgewertet. Hat er den Wert `true`, so wird das Programm ohne Beinträchtigung fortgesetzt. Hat er den Wert `false` so wird das Programm abgebrochen.

Führt man das Programm ohne die Option `-enableassertions` aus, so werden alle Zusicherungsstatements übersprungen (daher kein Effizienzverlust).

Zusicherungsstatements bieten sich zur Überprüfung von Vor- und Nachbedingungen an; sie können sorgfältige Tests nicht ersetzen.

Debugger

Ein Debugger erlaubt das schrittweise Abarbeiten von Programmen und die Inspektion von Variablen zu bestimmten Zeitpunkten. Debugger unterstützen die folgenden Operationen.

- Breakpoint setzen: Ein Breakpoint ist ein Programmpunkt an dem die Programmabarbeitung angehalten wird.
- Stepping: Hat die Abarbeitung einmal einen Breakpoint erreicht, so kann mit Step-Befehlen von Befehl zu Befehl weiterschreiten und dabei Methodenaufrufe überspringen (step over) oder in deren Rumpf “hineinspringen” (step into).
- Inspektion: Beim Erreichen eines Breakpoints lassen sich die Werte aller lebendigen Variablen inspizieren.
- Fortsetzung: Man kann die Programmabarbeitung bis zum nächsten Breakpoint ohne Unterbrechung fortsetzen.

Manche Debugger erlauben auch das Umsetzen von Variablenwerten.

Zusammenfassung

- Blackbox Test: Test der Funktionalität eines ganzen Programms ohne Kenntnis der Implementierung.
- Whitebox Test: Test unter Berücksichtung der Implementierung
- Unit Test: Test einer Methode
- Test harness, fixture: Programm zum Aufbau eines Aufrufkontext für eine Methode zu Testzwecken und Interpretation des Ergebnisses.
- Test suite: Zusammenfassung mehrerer Tests. Können gemeinsam ausgeführt werden.
- Regressionstesten: Erneutes Durchführen aller früheren Tests nach Programmänderungen.
- Orakel: Methode zur Berechnung erwarteter Testergebnisse.
- Debugger: Programm zur schrittweisen Abarbeitung von Code und Inspektion des Programmzustands.