2.3 Testen

Einstieg:

- Es sind individuelle Lösungen möglich, z.B. Mars Rover, Ariane 5, Schokoladenberg dank ERP-Bug, Patriot-Abwehrsystem verfehlt SCUD-Raketen (1.Golfkrieg), Strahlungstot wegen Softwarefehler (8 Tote, 20 Schwerverletzte), 22 Verhaftungen unschuldiger in Australien wegen fehlerhafter Datenbankmigration, Mariner 1 Crash, …
- Es sind individuelle Lösungen möglich.
  Gewünschtes Fazit: Klare Anforderungen und ständiges Testen sind unabdingbar.

A1

a) selbstgeschriebene Methoden:

- Kategorie `Bewegung`:
  `geheRueckwaerts`, `geheNachOben`, `geheNachUnten`, `geheNachRechts`, `geheNachLinks`
- Kategorie `Fühlen`:
  `beruehrtInRot`, `beruehrtInGruen`, `beruehrtInLila`

b)

Komponente	gewünschte Funktionsweise	Test
`geheRueckwaerts`	Figur (Affe) bewegt sich rückwärts	✗
`geheNachOben`	Figur (Affe) bewegt sich nach oben	✓
`geheNachUnten`	Figur (Affe) bewegt sich nach unten	✗
`geheNachRechts`	Figur (Affe) bewegt sich nach rechts	✓
`geheNachLinks`	Figur (Affe) bewegt sich nach links	✗
`beruehrtInRot`	gibt `WAHR` zurück, falls die Figur (Banane oder Spinne) auf rotem Hintergrund vom Affen berührt wird	✗
`beruehrtInGruen`	gibt `WAHR` zurück, falls die Figur (Banane oder Spinne) auf grünem Hintergrund vom Affen berührt wird	✗
`beruehrtInLila`	gibt `WAHR` zurück, falls die Figur (Banane oder Spinne) auf lila Hintergrund vom Affen berührt wird	✓

c) siehe Snap!-Datei `2-3_A1_Affe_Loesung.xml`

© C.C.Buchner Verlag, Bamberg

A2

a) Fehler in der Funktionalität:
- Die Figur (Hund) kann sich nicht nach unten bewegen.
- Die Figur (Hund) kann sich nicht korrekt nach oben bewegen.
- Nach Verwendung eines pinken Teleporters hängt die Figur (Hund) am Teleporter links oben fest.
- Nach Verwendung eines gelben Teleporters hängt die Figur (Hund) am Teleporter rechts oben fest.
- Es erfolgt keine Reaktion (Punktegewinn, Neuplatzierung) auf die Berührung des Balls durch die Figur (Hund).

b) siehe Snap!-Datei `2-3_A2_Hund_Loesung.xml`

Aufgabe 1

a) Markus‘ Vorgehen ist nicht empfehlenswert. Auch wenn er während des Programmierens keine Fehlermeldung angezeigt bekommen hat, heißt das nicht automatisch, dass sein Programm fehlerfrei funktioniert. Ein Compiler testet nur auf syntaktische Fehler (d.h. Fehler in Programmiersprache) und nicht auf semantische Fehler (d.h. Fehler im Inhalt, z.B. Verwendung der falschen Methode). Außerdem testet er nur kurz die wichtigsten Funktionalitäten – dadurch ist nicht sichergestellt, dass sein Programm für jeden Sonderfall funktioniert und nicht abstürzt.

b) Sinas Vorgehen ist besser als das von Markus – jedoch sollte sie ihr Programm neben den Tests der einzelnen Komponenten auch noch in seiner Gesamtheit testen, um sicherzugehen, dass sich die einzelnen Komponenten nicht gegenseitig behindern/verfälschen/etc. und nicht nur die einzelnen Komponenten, sondern auch das Gesamtprogramm fehlerfrei läuft.

c) Franz programmiert gemäß dem Prinzip „Pair-Programming“, was grundsätzlich durch ein Vier-Augen-Prinzip dazu beiträgt, Fehler zu vermeiden bzw. zu reduzieren. Trotzdem darf auf ausführliches Testen nicht verzichtet werden.

d) Linhuas Vorgehen ist der Idealfall und beim Entwickeln von Software dringend zu empfehlen.

© C.C.Buchner Verlag, Bamberg