**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**Báo cáo kiểm thử dòng dữ liệu với độ phủ all-uses**

**cho bài toán về giao dịch ngân hàng cơ bản**

**Bộ môn: Kiểm thử và đảm bảo chất lượng phần mềm**

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **: Nguyễn Việt Anh Khoa** |
| **Mã sinh viên** | **: 21020021** |
| **Lớp tín chỉ** | **: INT3117 1** |
| **Giảng viên** | **: ThS. Nguyễn Thu Trang** |

**Hà Nội, ngày 6 tháng 10 năm 2023**

**Mục lục**

**Lời mở đầu 3**

**I. Đặt vấn đề 4**

**1. Mô tả bài toán 4**

**2. Giới thiệu công cụ và mã nguồn 4**

**3. Tiểu kết 5**

**II. Phân tích và thiết kế ca kiểm thử 5**

**1. Phân tích đặc tả yêu cầu 5**

**2. Kiểm thử dòng dữ liệu 6**

**3. Sinh ca kiểm thử 7**

**4. Tiểu kết 10**

**III. Báo cáo kết quả kiểm thử 11**

**1. Kiểm thử dòng dữ liệu 11**

**2. Nhận xét về bộ kiểm thử 11**

**IV. Tổng kết 12**

**Tài liệu tham khảo 12**

Link github project: <https://github.com/electrostrike/testing-project-2>

**Lời mở đầu**

Kiểm thử và đảm bảo chất lượng phần mềm đóng vai trò cực kỳ quan trọng trong quá trình phát triển và triển khai phần mềm. Việc phát hiện lỗi, đảm bảo tính nhất quán và hiệu suất của phần mềm là những yếu tố không thể thiếu để đạt được thành công trong mọi dự án phần mềm. Trên thực tế, việc bỏ qua hoặc không đủ tập trung vào quá trình kiểm thử và đảm bảo chất lượng có thể dẫn đến hậu quả nghiêm trọng, bao gồm các lỗi không mong muốn, sự mất lòng tin của người dùng hay thậm chí là thiệt hại tài chính đáng kể.

Trong bài báo cáo này, chúng ta sẽ cùng tìm hiểu phương pháp kiểm thử dòng dữ liệu, một trong những phương pháp quan trọng trong kiểm thử hộp trắng. Ngoài ra, chúng ta sẽ áp dụng kĩ thuật kiểm thử này để thực hiện kiểm thử cho chương trình đơn giản mô phỏng lại các chức năng cơ bản trong giao dịch ngân hàng. Các chi tiết về phân tích đặc tả yêu cầu hay cách thức sinh ca kiểm thử sẽ được mô tả rõ hơn trong các phần sau của báo cáo.

Ngoài phần mở đầu, bài báo cáo bao gồm 4 phần chính:

* Phần 1: Đặt vấn đề
* Phần 2: Phân tích và thiết kế các ca kiểm thử
* Phần 3: Báo cáo kết quả kiểm thử
* Phần 4: Tổng kết

Tuy đã có nỗ lực trong việc nghiên cứu và viết báo cáo, song do trình độ còn hạn chế nên bài báo cáo chắc chắn không thể tránh khỏi những thiếu sót nhất định. Em rất mong nhận được ý kiến đóng góp từ thầy cô để học hỏi thêm nhiều kiến thức mới và bổ ích.

**I. Đặt vấn đề**

**1. Mô tả bài toán**

Trong giao dịch ngân hàng, có rất nhiều bài toán liên quan cần được giải quyết để quá trình giao dịch được đảm bảo an toàn và hiệu quả. Một số vấn đề nổi bật trong lĩnh vực này có thể kể đến như Xác thực giao dịch, Quản lý tài khoản, Xử lý giao dịch, Bảo mật thông tin hay Quản lý rủi ro. Ở đây, chúng ta chỉ quan tâm đến việc giải quyết những chức năng đơn giản nhất trong Quản lý tài khoản là Kiểm tra số dư (display), Gửi tiền (deposit) và Rút tiền (withdraw). Trong đó:

* Một tài khoản mặc định là hợp lệ, tức là ban đầu tài khoản phải có số dư (balance) >= 0.
* Chức năng Kiểm tra số dư cho phép người dùng hiển thị đúng số dư tài khoản đang có lên trên màn hình.
* Chức năng Gửi tiền cho phép người dùng gửi tiền vào tài khoản, trong đó số tiền gửi vào phải là 1 số nguyên dương.
* Chức năng Rút tiền cho phép người dùng rút tiền từ trong tài khoản ra, trong đó số tiền rút ra phải là 1 số nguyên dương và không được phép vượt quá số dư trong tài khoản.

**2. Giới thiệu công cụ và mã nguồn**

Với những đặc tả kể trên, em đã quyết định sử dụng 2 công cụ chính để giải quyết bài toán là JavaScript và framework Jest.

JavaScript là một ngôn ngữ lập trình phổ biến và mạnh mẽ. Nó được sử dụng để tạo ra nhiều chương trình đa dạng có khả năng mở rộng tốt, từ những ứng dụng web có khả năng tương tác với người dùng tới những chương trình tính toán đơn giản. Đây là ngôn ngữ rất thích hợp để thực hiện những yêu cầu đơn giản được nêu ở trên, đồng thời người dùng cũng có thể tái sử dụng mã nguồn (code) vào những dự án lớn hơn.

Jest là một framework kiểm thử phổ biến được sử dụng cho ngôn ngữ lập trình JavaScript. Nó là một công cụ mạnh mẽ giúp viết và chạy các ca kiểm thử đơn vị (unit tests) một cách dễ dàng và hiệu quả. Jest được tạo ra bởi Facebook và ban đầu được sử dụng trong dự án React, nhưng hiện nay nó có thể được sử dụng cho bất kỳ dự án JavaScript nào. Với khả năng tự động phát hiện và chạy tests cũng như cú pháp đơn giản và dễ đọc, Jest là một trong những framework kiểm thử cho JavaScript được ưa chuộng nhất.

Sự kết hợp của ngôn ngữ lập trình JavaScript và framework kiểm thử Jest đã giúp em tạo ra sản phẩm một cách dễ dàng và hiệu quả. Sau đây là [link github](https://github.com/electrostrike/testing-project-2) cho sản phẩm (bao gồm các 1 file code, 1 file test và các dependencies)

**3. Tiểu kết**

Với sự kết hợp của Javascript và framework Jest, em đã có thể giải quyết vấn đề được nêu cũng như thực hiện kiểm thử đơn vị một cách hiệu quả. Các phần tiếp theo em sẽ đi vào tìm hiểu chi tiết hơn về quá trình kiểm thử và báo cáo kết quả kiểm thử

**II. Phân tích và thiết kế ca kiểm thử**

**1. Phân tích đặc tả yêu cầu**

Thông qua các đặc tả về chương trình, chúng ta có các ràng buộc về đầu vào (input) và đầu ra (output) như sau:

* Trước hết, ta cần khởi tạo một tài khoản hợp lệ.
* Đầu vào: Gồm 1 xâu type trong 3 xâu ‘display’, ‘deposit’ hoặc ‘withdraw’ cho biết chức năng cần thực hiện. Trường hợp chức năng là ‘deposit’ hoặc ‘withdraw’ thì cần nhập thêm 1 số nguyên dương amount là số tiền tương ứng với giao dịch.
* Đầu ra: Một số nguyên balance thể hiện số tiền tài khoản hiện có sau giao dịch.
* Trường hợp đầu vào không hợp lệ, đầu ra mặc định là 1 xâu báo lỗi input không phù hợp (Error: Invalid input).

Dựa theo các phân tích ràng buộc nêu trên, ta có thể bắt đầu thực hiện kiểm thử dòng dữ liệu ở phần tiếp theo.

**2. Kiểm thử dòng dữ liệu**

Kiểm thử dòng dữ liệu (data flow testing) là một kĩ thuật kiểm thử quan trọng trong kiểm thử hộp trắng. Khác với kiểm thử hộp đen, kiểm thử hộp trắng cho phép người kiểm thử biết được mã nguồn của phần mềm. Việc kiểm thử chủ yếu dựa trên phân tích các cấu trúc bên trong của hệ thống, các chi tiết về từng thủ tục, từng dòng điều khiển và các trạng thái của dữ liệu. Dựa vào những hành vi của hệ thống kết hợp với những phân tích đặc tả yêu cầu, ta có thể xác định được phần mềm có đang chạy đúng như mong đợi hay không.

Có 2 phương pháp chính trong kiểm thử dòng dữ liệu là kiểm thử dòng dữ liệu tĩnh và kiểm thử dòng dữ liệu động. Kiểm thử dòng dữ liệu tĩnh là việc thực hiện phân tích mã nguồn mà không chạy chương trình. Phương pháp này có thể giúp người lập trình phát hiện các vấn đề phổ biến khi sử dụng dữ liệu như gán giá trị liên tiếp, sử dụng giá trị khi chưa gán giá trị hay khai báo biến mà không sử dụng. Tuy nhiên, việc kiểm thử tĩnh không hoàn toàn đảm bảo tìm được mọi lỗi liên quan đến dữ liệu, do đó chúng ta có thêm phương pháp kiểm thử dòng dữ liệu động - tức là tiến hành chạy chương trình - để đảm bảo dữ liệu được tạo ra và sử dụng theo đúng mong đợi.

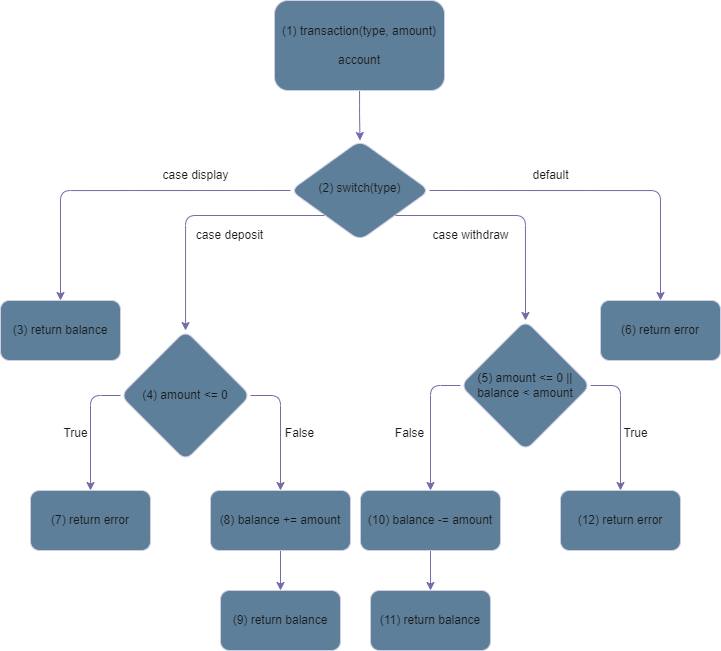
Trước khi tiến hành kiểm thử, chúng ta cần tìm hiểu một số khái niệm quan trọng sau:

* Def(v) (Define): là một câu lệnh gán giá trị cho biến v.
* Use(v): là một câu lệnh sử dụng biến v, trong đó:
  + P-use(v) (Predicate use): là một câu lệnh sử dụng biến v trong biểu thức   
    điều kiện.
  + C-use(v) (Computation-use): là một câu lệnh sử dụng biến v trong các trường hợp còn lại (không phải p-use).
* Du-pair(v): là một cặp đỉnh (d, u) tương ứng (def(v), use(v)).
* Def-clear path: một đường đi trên đồ thị dòng điều khiển (Control Flow Graph), trong đó:
  + Đỉnh bắt đầu là một câu lệnh def(v).
  + Trong các đỉnh còn lại được đi qua trong đường đi, không có câu lệnh def(v) nào khác, tức là câu lệnh def(v) ở đỉnh bắt đầu là duy nhất.
* Def-use path (Du-path): (d, u) là một cặp def-use của biến v nếu:
  + d là đỉnh chứa def(v).
  + u là đỉnh chứa use(v).
  + Tồn tại def-clear path của biến v giữa 2 đỉnh này.
* Complete path: là đường đi bắt đầu ở node Entry, kết thúc ở node Exit.

Ở đây, chúng ta đặc biệt quan tâm đến việc kiểm thử dòng dữ liệu động với độ phủ all-uses, tức là các ca kiểm thử cần đảm bảo với mỗi biến v trong chương trình, ít nhất 1 def-clear path giữa mọi def(v) và mọi use(v) được kiểm tra.

**3. Sinh ca kiểm thử**

Trước hết, để có thể sinh ca kiểm thử cho kiểm thử dòng dữ liệu, ta cần xác định được đồ thị dòng điều khiển của chương trình. Dựa vào những đặc tả ở phần II.1, ta có đồ thị dòng điều khiển cho chương trình như sau:



Tiếp theo, dựa vào đồ thị dòng điều khiển, ta xác định được các biến, du-pair, def-clear, complete path cần kiểm thử như bảng dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Var | No | Du-pair | Def-clear path | Complete path |
| type | **1** | 1, 2(display) | 1, 2(display) | **1, 2(display), 3** |
| **2** | 1, 2(deposit) | 1, 2(deposit) | **1, 2(deposit), 4(T), 7** |
| **3** | 1, 2(withdraw) | 1, 2(withdraw) | **1, 2(withdraw), 5(F), 10, 11** |
| **4** | 1, 2(default) | 1, 2(default) | **1, 2(default), 6** |
| amount | **5** | 1, 4(T) | 1, 2(deposit), 4(T) | 1, 2(deposit), 4(T), 7 |
| **6** | 1, 4(F) | 1, 2(deposit), 4(F) | **1, 2(deposit), 4(F), 8, 9** |
| **7** | 1, 8 | 1, 2(deposit), 4(F), 8 | 1, 2(deposit), 4(F), 8, 9 |
| **8** | 1, 5(T) | 1, 2(withdraw), 5(T) | **1, 2(withdraw), 5(T), 12** |
| **9** | 1, 5(F) | 1, 2(withdraw), 5(F) | 1, 2(withdraw), 5(F), 10, 11 |
| **10** | 1, 10 | 1, 2(withdraw), 5(F), 10 | 1, 2(withdraw), 5(F), 10, 11 |
| balance | **11** | 1, 3 | 1, 2(display), 3 | 1, 2(display), 3 |
| **12** | 1, 5(T) | 1, 2(withdraw), 5(T) | 1, 2(withdraw), 5(T), 12 |
| **13** | 1, 5(F) | 1, 2(withdraw), 5(F) | 1, 2(withdraw), 5(F), 10, 11 |
| **14** | 8, 9 | 8, 9 | 1, 2(deposit), 4(F), 8, 9 |
| **15** | 10, 11 | 10, 11 | 1, 2(withdraw), 5(F), 10, 11 |
| error | **16** | 1, 6 | 1, 2(default), 6 | 1, 2(default), 6 |
| **17** | 1, 7 | 1, 2(deposit), 4(T), 7 | 1, 2(deposit), 4(T), 7 |
| **18** | 1, 11 | 1, 2(withdraw), 5(T), 12 | 1, 2(withdraw), 5(T), 12 |

Như vậy, ta có tổng cộng 18 du-pair cần kiểm tra để đạt được độ phủ all-uses. Tuy nhiên các du-pair có complete-path trùng nhau nên thực tế ta chỉ cần sinh ca kiểm thử sao cho luồng thực hiện 6 đường đi được bôi đỏ trong bảng là:

* **T1: 1, 2(display), 3**
* **T2: 1, 2(deposit), 4(T), 7**
* **T3: 1, 2(deposit), 4(F), 8, 9**
* **T4: 1, 2(withdraw), 5(T), 12**
* **T5: 1, 2(withdraw), 5(F), 10, 11**
* **T6: 1, 2(default), 6**

Cuối cùng, dựa vào các complete path cũng như đặc tả yêu cầu, ta có thể thiết kế các ca kiểm thử một cách hoàn chỉnh:

|  |  |  |  |
| --- | --- | --- | --- |
| No | Đường đi | Input | Expected Output |
| T1 | 1, 2(display), 3 | display  balance = 100 | 100 |
| T2 | 1, 2(deposit), 4(T), 7 | deposit  amount = -50  balance = 100 | error |
| T3 | 1, 2(deposit), 4(F), 8, 9 | deposit  amount = 50  balance = 100 | 150 |
| T4 | 1, 2(withdraw), 5(T), 12 | withdraw  amount = 150  balance = 100 | error |
| T5 | 1, 2(withdraw), 5(F), 10, 11 | withdraw  amount = 50  balance = 100 | 50 |
| T6 | 1, 2(default), 6 | transfer | error |

**3. Tiểu kết**

Có thể thấy, quá trình phân tích và thiết kế ca kiểm thử đóng vai trò vô cùng quan trọng trong việc phát triển phần mềm. Bằng việc nắm rõ cấu trúc và hành vi của mã nguồn, ta có thể xây dựng sơ đồ dòng điều khiển của chương trình. Kết họp với các phân tích đặc tả yêu cầu, ta sinh được bộ ca kiểm thử chất lượng và hoàn chỉnh, đảm bảo độ phủ all-uses của kiểm thử dòng dữ liệu. Tiếp theo là phần báo cáo kết quả kiểm thử.

**III. Báo cáo kết quả kiểm thử**

**1. Kiểm thử dòng dữ liệu**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Expected Output | Actual Output | Pass/Fail |
| T1 | 100 | 100 | Pass |
| T2 | error | error | Pass |
| T3 | 150 | 150 | Pass |
| T4 | error | error | Pass |
| T5 | 50 | 50 | Pass |
| T6 | error | error | Pass |

**2. Nhận xét về bộ kiểm thử**

Kết quả của bộ kiểm thử được thiết kế ở trên là tất cả các test case được thực hiện đều pass. Đây là một kết quả có phần đáng tin cậy, bởi bộ kiểm thử đã được thiết kế một cách cẩn thận và hoàn thiện, đúng theo các đặc tả yêu cầu và dòng dữ liệu.

Tuy nhiên, điều này chưa chắc đã đảm bảo chương trình không có lỗi tiềm ẩn. Mặc dù chương trình đã cho ra kết quả đúng như mong đợi, nếu ta để ý kĩ mã nguồn thì ở trường hợp chương trình chạy theo đường *T1* (*1, 2(display), 3*), biến amount được khởi tạo ở đầu nhưng lại không được sử dụng tại bất kì vị trí nào trên đường đi này. Việc này tuy rằng chưa gây ra bất kì hành vi không mong muốn nào cho chương trình, nhưng lỗi này cần được khắc phục sớm để tránh các thiệt hại có thể gây ra khi chương trình này được phát triển thêm.

**IV. Tổng kết**

Như vậy, bài báo cáo đã phân tích đầy đủ quá trình kiểm thử, từ việc cần phải xác định rõ vấn đề cho đến phân tích, thiết kế các ca kiểm thử và cuối cùng là đưa ra kết luận sau khi thực hiện kiểm thử xong. Có thể thấy, kiểm thử hộp trắng là một khâu quan trọng trong quá trình kiểm thử, giúp lập trình viên phát hiện và khắc phục những rủi ro tiềm ẩn mà phần mềm có thể gây ra từ sớm, từ đó còn có thể tiết kiệm tài nguyên và nhân lực trong toàn bộ quá trình phát triển phần mềm.

Em xin cảm ơn thầy cô đã dành thời gian cho bài báo cáo này. Một lần nữa, nhận thức được rằng bài làm vẫn còn nhiều thiếu sót, em rất mong nhận được nhận xét từ thầy cô để học hỏi thêm nhiều kiến thức mới và bổ ích.

**Tài liệu tham khảo**

1. Slide bài giảng của khóa học
2. [Viblo Code - Tìm hiểu về Jest](https://viblo.asia/p/tim-hieu-ve-jestjs-viet-unit-test-cho-javascript-gDVK2pkXlLj)
3. [Viblo Asia - Kiểm thử hộp trắng](https://viblo.asia/p/kiem-thu-hop-trang-jvElaLGAZkw)