

Assignment 1 Report
Testing Exception-Handling Policies in Concurrent Systems

Concordia University
COEN 448
Software Testing and Validation
Course Instructor: Yan Liu

By
Mohamad Edelby (40251628)
Section W WA

"I certify that this submission is my original work and meets the Faculty's Expectations of
Originality" – Mohamad Edelby (40251628), February 15, 2026

Date Submitted: Friday February 15th, 2026

Github Link: https://github.com/momerz7/448_A1

Task 2 - Implementing Policies

Fail-Fast Policy

The approach taken to implement this policy uses `CompletableFuture.allOf` in addition to propagates exceptions. The key is that no partial result is returned so that if one service fails, the whole system fails.

Fail-Partial Policy

The approach taken to implement this policy is where only successful results are returned. If there is a failure, it is ignored. The final output only has successful operations, failures aren't determinantal to the system holding.

Fail-Soft Policy

The main part of this policy is that if a failure occurs then a predefined fallback value replaces it. This ensures that the system completes even if a service fails.

Task 3 – Unit Testing

Requirements were provided to incorporate unit testing.

- No Mockito
- All futures must be awaited with timeouts
- Tests must verify policy semantics

And categories were specified:

- Fail-Fast: failure propagates (`assertThrows`)
- Fail-Partial: partial results returned
- Fail-Soft: fallback values used
- Liveness: no deadlock or infinite wait
- Nondeterminism: completion order observed (not asserted)

My implementation of Unit Testing

Function	Purpose	Strict Rules	Test Categories
Static class FailureHelper extends Microservice	Simulates microservice that always fails, returning <code>CompletableFuture.failedFuture(...)</code>	<ul style="list-style-type: none"> No Mockito Timeout-safe permit futures to still complete 	<ul style="list-style-type: none"> Fail-fast Fail-Partial Fail-Soft
<code>void testFailurePropagates()</code>	Exceptions propagates when any microservice fails	<ul style="list-style-type: none"> No Mockito Timeout: <code>(get(1, TimeUnit.SECONDS))</code> Tests policy semantics 	<ul style="list-style-type: none"> Fail-Fast: failure propagate (<code>assertThrows</code>) Liveness: no deadlock
<code>void testSuccessFailFast()</code>	Ensures policy succeeds normally when every microservices succeed	<ul style="list-style-type: none"> No Mockito Timeout used Tests policy semantics 	<ul style="list-style-type: none"> Fail-fast: full success path Liveness
<code>void testOnlySuccess_failPartial()</code>	Test that only returns when successful results and ignores failures	<ul style="list-style-type: none"> No Mockito Timeout used Tests policy semantics (partial results allowed) 	<ul style="list-style-type: none"> Fail-Partial: partial results returned Liveness
<code>void testEmptyList_failPartial()</code>	When all microservices fail → Fail-partial returns empty list	<ul style="list-style-type: none"> No Mockito Timeout Used Tests policy semantics (no exception escapes) 	<ul style="list-style-type: none"> Fail-Partial: empty output if total failure
<code>void testFallbackFailure_failsoft()</code>	Failed results replaced with a fallback value + successful results kept	<ul style="list-style-type: none"> No Mockito Timeout Used Tests policy semantics (substitute with fallback value) 	<ul style="list-style-type: none"> Fail-soft: fallback used for failures
<code>void testAllFail_failsoft()</code>	Ensures Fail-Soft policy is still returning a valid result even if all microservices fail, these valid results would be all fallback values	<ul style="list-style-type: none"> No Mockito Timeout Never throws 	<ul style="list-style-type: none"> Fail-soft: always normal completion

In addition to these unit tests that involved the three policies, there was also a nondeterministic requirement. This was achieved with `testProcessAsync()`: asynchronous pipeline works well when every microservice succeeds (formatting + ordering), `testProcessAsync_withDifferentMessages()`: expected output is produced thanks to processor

being handled correctly with corresponding inputs, and
showNondeterministic_completionOrderVaries(): microservices complete in nondeterministic
order.

Task 4 – Documentation

“docs/failure-semantics.md” was created and included in the GitHub repo. This document
defines how each of the policies work, their risks, and examples of what kind of systems they
can be incorporated in.

GitHub Workflow Screenshots

3 Github Issues

Implement Fail-Fast (Atomic Policy) #2

Closed #7

momerz7 opened 2 days ago Owner ...

If any service fails then the whole operation fails.

Create sub-issue

momerz7 mentioned this 2 days ago
Implemented Fail-Fast Policy #7

michaeliadisernia closed this as completed in #7 yesterday

Implement Fail-Soft (Fallback Policy) #4

Closed #11

momerz7 opened 2 days ago Owner ...

Replace failures with fallback value and always complete normally.

Create sub-issue

momerz7 mentioned this 17 hours ago
implemented fail soft policy #11

momerz7 closed this as completed in #11 17 hours ago

Write Unit Tests #5

Closed #12

momerz7 opened 2 days ago Owner ...

Strict Rules

- No Mockito
- All futures must be awaited with timeouts
- Tests must verify policy semantics

Required Test Categories


















- Fail-Fast: failure propagates (assertThrows)
- Fail-Partial: partial results returned
- Fail-Soft: fallback values used
- Liveness: no deadlock or infinite wait
- Nondeterminism: completion order observed (not asserted)

Create sub-issue

momerz7 mentioned this 4 hours ago
implemented unit testing #12

momerz7 closed this as completed in #12 4 hours ago

2 Feature Branches

Branches					
<div>OverviewYoursActiveStaleAll</div>					
<div>Q Search branches...</div>					
Branch	Updated	Check status	Behind	Ahead	Pull request
documentation 	 20 minutes ago		0	3	
feature/unit-test 	 4 hours ago		1	0	 #12
feature/fail-soft 	 17 hours ago		3	0	 #11
feature/fail-partial 	 yesterday		5	0	 #10
feature/fail-fast 	 2 days ago		8	0	 #7
Github_setup 	 2 days ago		10	0	 #1

2 Pull Requests

Implemented Fail-Fast Policy #7

Merged [michaeliadisernia](#) merged 1 commit into `main` from `feature/fail-fast` yesterday

Conversation 2 Commits 1 Checks 0 Files changed 5

momerz7 commented 2 days ago (Owner) ...

This commit has the implementation of the fail-fast (atomic policy) while using `CompletableFuture.allOf`, propagates exception, and no partial result returned.

Closes #2

Implemented fail fast using `CompletableFuture` 895d709

michaeliadisernia commented yesterday (Collaborator) ...

Well structured and complete. Implementation runs on my machine without issues. Merged to main.

michaeliadisernia merged commit `ca58159` into `main` yesterday (Revert)

michaeliadisernia self-requested a review yesterday

michaeliadisernia reviewed yesterday (View reviewed changes)

michaeliadisernia left a comment (Collaborator) ...

Current implementation successfully does the following:

- Executes all services concurrently
- Preserves message-service pairing
- Correctly aggregates successful results
- Clean and readable functional style

Current implementation is concurrent but not strictly fail-fast. To be improved to meet fail-fast semantics, it must propagate failure immediately without waiting for all tasks to complete.

Reviewers
[michaeliadisernia](#)

Assignees
No one—[assign yourself](#)

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request will create a new branch off this branch.

Notifications
[Unsubscribe](#)
You're receiving notifications because you authored the thread.

2 participants
[momerz7](#) [michaeliadisernia](#)

[Lock conversation](#)

implemented fail soft policy #11

Merged [momerz7](#) merged 1 commit into `main` from `feature/fail-soft` 17 hours ago

Conversation 0 Commits 1 Checks 0 Files changed 2

momerz7 commented 17 hours ago (Owner) ...

Failed microservice calls are replaced with a fallback value so the overall computation always completes normally.

Closes #4

implemented fail soft policy 86bd4ef

momerz7 merged commit `a59a401` into `main` 17 hours ago (Revert)

Reviewers
No reviews

Assignees
No one—[assign yourself](#)

Labels
None yet

Projects
None yet


Milestone

1 Peer Code Review


Feature/fail partial #10


Merged **michaeliadisernia** merged 2 commits into `main` from `feature/fail-partial` yesterday


Conversation 1 Commits 2 Checks 0 Files changed 3



**momerz7** commented yesterday Owner ...

All microservices invoked concurrently.
system only returns successful results.
Failures captured and skipped.
Closes [#3](#).

**{momerz7}** added 2 commits yesterday

`Implemented fail-partial policy` [4c470c7](#)



`implemented fail-partial` [8cfb6a8](#)


**michaeliadisernia** approved these changes yesterday View reviewed changes

michaeliadisernia left a comment Collaborator ...

The Fail-Partial method implements partial failure tolerance by handling exceptions individually for each microservice call. Using `.handle()`, any failed service returns null instead of propagating the exception, allowing other services to complete normally. This is good and as required in the assignment.


Because failures are converted into null values, `CompletableFuture.allOf()` completes successfully even if some services fail. The final result filters out null values, ensuring only successful responses are returned.

**michaeliadisernia** merged commit **d157900** into `main` yesterday Revert

**Pull request successfully merged and closed** Delete branch

You're all set — the `feature/fail-partial` branch can be safely deleted.

Reviewers

**michaeliadisernia**

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects


None yet

Milestone


No milestone

Development

Successfully merging this pull request will resolve these issues.



 **Implement Fail-Partial (Best-Effort)**


Notifications

 **Unsubscribe**

You're receiving notifications because you authored the thread.

2 participants



 **Lock conversation**