ACS Assignment 1

ndp689 mnp553

November 2021

1. Provide a short description of your implementation and tests.

(a) How does your implementation and tests address the all-or-nothing semantics?

Implementation:

- getBooksInDemand: We used *Stream* API and filtered by the missed sales and return a list of books.
- rateBooks: in this function, we use *validate* function to check if each rating's rating score is between 0 and 5, and to check if each rated book has valid ISBN and in stock. If one of them has a validation error, a BookStoreException will be thrown, and we will save none of the ratings. Otherwise, all the ratings will be saved.
- getTopRatedBooks: in this function, we just check the parameter *numBooks*. If it is less than 0, a BookStoreException will be thrown. Otherwise, we return the top *numBooks* rated books.

Testing:

- testing getBooksInDemand: In the test case *testGetBooksInDemand*, we have two books added(the first one with 0 numSaleMisses and the second one with 1 numSaleMisses). We call getBooksInDemand method and only get the second book called Python.
- testing rateBooks: In the test case *testRateNonExistingISBN*, we have two ratings (the first one has a valid ISBN and a valid rating score, and the second one has a non-existing ISBN in stock and a valid rating score). So, there is a validation error, and no books

are rated. In the test case *testRateBooks*, all the ratings are legal, so we can save such ratings.

- tesing getTopRatedBooks: In the test case *testGetTopRatedBooks*, first, we add a new book and its copies in stock, and then, we rate the default book with 2 score and rate the new book with 5 score. Then, we want to get the top 1 rated book and call *getTopRatedBooks* function, which returns the new book. In the test case *testGetNegativeTopRatedBooks*, it throws an exception, because its parameter is illegal (negative).

(b) How did you test whether the service behaves according to the interface regardless of use of RPCs or local calls?

We did it by checking the results of each test case. We set the *localTest* true to use local calls and set it false to use RPCs calls, and all the test cases we implemented pass.

2. We have stated above that the architecture achieves strong modularity. Explain this in the context of the following questions.

(a) In which sense is the architecture strongly modular?

We use a Client-Server architecture based on RPCs. A client sends a request to the server, and the server handles the request and calls the corresponding function. Each module (CertainBookStore, BookStore-HTTPProxy, and so on) does its own work clearly. So the architecture is strongly modular.

(b) What kind of isolation and protection does the architecture provide between the two types of clients and the bookstore service?

By defining the two types of books: mutable and immutable books. The bookstore clients can only get the information of immutable books, so they cannot change such information. But the stockmanager clients are allowed to mutate data of books.

(c) How is enforced modularity affected when we run clients and services locally in the same JVM, as possible through our test cases?

If an error happens from clients or services, the clients and the services will be shut down.

3.

(a) Is there a naming service in the architecture? If so, what is its functionality?

Yes. The *BookStoreHTTPMessageHandler* class provides the service, and then, we can use *messageTag* to call the corresponding function.

(b) Describe the naming mechanism that allows clients to discover and communicate with services.

First, use the server's address and a message tag to generate a *urlString*. And then, send a request to the server. The server can use the message tag to call the corresponding function that clients want.

4. We have studied three types of RPC semantics: at-least-once, at-most-once, and exactly-once semantics. What RPC semantics is implemented in the architecture? Justify your answer.

"At-most-once" is implemented.

- Operations have side-effects, which modify the state or data in the server.
- Operations will not be completed if any exceptions are thrown, thus making sure the transmission is duplicate-free.

5. Services employing HTTP as a communication mechanism often deploy web proxy servers for scalability in the number of simultaneous client connections.

(a) Is it safe to use web proxy servers with the architecture of Figure 1?

Yes.

(b) If so, explain why this is safe and describe in between which components these proxy servers should be deployed. If not, why not?

The modifier *synchronized* is used to guarantee the success of execution, and the the server only makes part of the interface public.

The web proxy can be implemented between *BookStoreHTTPProxy* and *BookStoreHTTPServer*, *StockManagerHTTPProxy* and *BookStoreHTTPServer*.

6. Given the discussion in the question above, consider now the following questions:

(a) Is/are there any scalability bottleneck/s in this architecture with respect to the number of clients?

Yes.

(b) If so, where is/are the bottleneck/s? If not, why can we infinitely scale the number of clients accessing this service?

The bottleneck is the server. Because its state is stored in the memory. If the number of clients accessing the service is so large that it runs out of the memory, the server cannot process more requests in time.

But we can infinitely scale the number of clients accessing this service. Because the server can process more requests when it processes enough requests and the memory has remaining capacity.

7. Suppose the server-side of the architecture fails by a crash of the server machine where the CertainBookStore class is being run. In this context, explain the following.

(a) Would clients experience failures differently if web proxies were used in the architecture?

Yes, proxies have their own way for dealing with failures, which can make the response from the proxies different from the response directly from the web server.

(b) Could caching at the web proxies be employed as a way to mask failures from clients?

Yes. When clients send bad requests, only the proxies need to response without communicating between the server and proxies.

(c) How would the use of web caching affect the semantics offered by the bookstore service?

Using web caching will not affect the semantics since it will bring down the number of requests sent to the web server and reduce the time that is needed for processing. When some requests which modify the state or data of the server are made, it runs normally.