

# Interview Exam: RandomByteGenerator Code

Interviewer

March 13, 2025

## Instructions

Answer the following questions to the best of your ability. Provide clear and concise explanations. You may refer to the code provided below.

## Code Reference

```
1 #include <iostream>
2 #include <vector>
3 #include <random>
4 #include <thread>
5 #include <chrono>
6
7 class RandomByteGenerator {
8 public:
9     using DataCallback = std::function<void(std::vector<
    uint8_t>&)>;
10
11     RandomByteGenerator() : isRunning_(false) {}
12
13     void registerOnByteDataRandomCallback(DataCallback
    callback) {
14         if (callback) {
15             callback_ = callback;
16         } else {
17             std::cerr << "[RandomByteGenerator] Invalid
    callback." << std::endl;
18         }
19     }
20 }
```

```

21     void receiveData(std::vector<uint8_t>& data) {
22         std::cout << "[RandomByteGenerator] Received data.
Not implemented." << std::endl;
23     }
24
25     void transitionToNextModule(std::shared_ptr<IModule>
nextModule) {
26         registerOnByteDataRandomCallback([nextModule](std::
vector<uint8_t>& byteVector) {
27             if (nextModule) {
28                 nextModule->receiveData(byteVector);
29             } else {
30                 std::cerr << "[RandomByteGenerator] Next
module is null." << std::endl;
31             }
32         });
33     }
34
35     std::vector<uint8_t> generateRandomLengthByteVector() {
36         std::random_device rd;
37         std::mt19937 gen(rd());
38         std::uniform_int_distribution<size_t> lengthDistrib
(1, 100);
39         std::uniform_int_distribution<uint8_t> byteDistrib(0,
255);
40
41         size_t length = lengthDistrib(gen);
42         std::vector<uint8_t> byteVector;
43         byteVector.reserve(length);
44
45         for (size_t i = 0; i < length; ++i) {
46             byteVector.push_back(byteDistrib(gen));
47         }
48         return byteVector;
49     }
50
51     void generateRandomBytes() {
52         while (isRunning_) {
53             std::vector<uint8_t> byteVector =
generateRandomLengthByteVector();
54             if (callback_) {
55                 try {
56                     callback_(byteVector);
57                 } catch (const std::exception& e) {

```

```

58         std::cerr << "[RandomByteGenerator]
Callback error: " << e.what() << std::endl;
59     }
60     } else {
61         std::cerr << "[RandomByteGenerator] No
callback set." << std::endl;
62     }
63     std::this_thread::sleep_for(std::chrono::
milliseconds(50));
64     }
65 }
66
67 void start() {
68     isRunning_ = true;
69     generationThread_ = std::thread([this]() {
generateRandomBytes(); });
70 }
71
72 void stop() {
73     isRunning_ = false;
74     std::cout << "[RandomByteGenerator] Stopping thread."
<< std::endl;
75
76     if (generationThread_.joinable()) {
77         generationThread_.join();
78         std::cout << "[RandomByteGenerator] Thread
stopped." << std::endl;
79     } else {
80         std::cout << "[RandomByteGenerator] Thread not
joinable." << std::endl;
81     }
82 }
83
84 private:
85     std::thread generationThread_;
86     bool isRunning_;
87     DataCallback callback_;
88 };

```

## Questions

### 1. Class Design and Purpose

- What is the purpose of the RandomByteGenerator class? Explain

its main functionality.

- (b) Why is the `DataCallback` type defined as `std::function<void(std::vector<uint8_t>`  
What does it represent?

## 2. Thread Management

- (a) How does the `start()` method work? What happens when it is called?
- (b) What is the role of the `isRunning_` variable in the `generateRandomBytes()` method?
- (c) Explain the purpose of the `stop()` method. What happens if the thread is not joinable?

## 3. Callback Mechanism

- (a) What is the purpose of the `registerOnByteDataRandomCallback` method? How is it used in the code?
- (b) In the `transitionToNextModule` method, why is the callback registered with a lambda function? What does this lambda function do?

## 4. Random Data Generation

- (a) How does the `generateRandomLengthByteVector` method generate random bytes? Explain the role of `std::random_device`, `std::mt19937`, and `std::uniform_int_distribution`.
- (b) What is the range of the random bytes generated by this method? How is the length of the byte vector determined?

## 5. Error Handling

- (a) How does the code handle errors in the callback function? What happens if an exception is thrown?
- (b) What happens if the `callback_` is not set when `generateRandomBytes()` is called?

## 6. Code Improvements

- (a) Are there any potential issues with the current implementation of `generateRandomBytes()`? How would you improve it?

- (b) How would you modify the code to allow for configurable sleep durations between byte generation?

## **Scoring**

Each question is worth 5 points. The total score is out of 30 points.

**Good Luck!**