**Thợ lành nghề #12: Ba dòng mã xấu xí (SMCRemote – phần 2)**

Ngày 18 Tháng 3 năm 2003

Tôi nghỉ giải lao trên đài quan sát. Khi lớp chắn bằng nước đá đi xuyên qua vùng phân tử dày cộm làm cho lớp nước đá nhập nhoè trong những làn chớp xanh và những mô hình chuyển biến khắp bề mặt của lớp chắn – làm tôi nhớ đến Bắc cực quang của trái đất.

Như thường lệ, Jerry đang đợi tôi sau buổi giải lao. Gã nhìn tôi và nói, “OK, hãy gởi một tệp tin qua socket.”

“Ông xem cuộc trưng bày Cherenkov chưa?”

“Tuyệt đẹp!” gã cười mỉm. Tôi đoán lâu lâu gã cũng nghỉ giải lao – nhưng thường thường gã đi đâu?

“OK, hãy gửi một tệp tin đi” tôi nói. “Tôi sẽ viết kiểm thử tiếp theo.” Tôi giành lấy bàn phím và bắt đầu gõ. Phần đầu tiên tôi gõ là đoạn mã dùng để tạo tệp tin được gởi qua socket.

public void testSendFile() throws Exception {

File f = new File("testSendFile");

FileOutputStream stream = new FileOutputStream(f);

stream.write("I am sending this file.".getBytes());

stream.close();

}

“Tôi biết ông muốn tạo tệp tin dữ liệu trong mã kiểm thử hơn là phụ thuộc vào tình trạng chúng có hiện diện hay không,” tôi nói.

“Ðúng thế,” Jerry trả lời. “Nhưng mày có thấy mày bị mấy thứ lặp lại không?”

Tôi xem lại phần kiểm thử và thấy ngay chúng tôi viết đoạn mã gần như trùng lặp với phương thức testCountBytesInFile() mà chúng tôi đã hoàn thành trước giờ giải lao.

“Chỉ có bốn dòng mã mà thôi,” tôi nói. “Hầu như không có trùng lặp lớn”.

“Ðúng thế,” Jerry đáp. “Nhưng sự trùng lặp nên được loại bỏ ngay khi có thể được. Không thì mày sẽ ôm một mớ mã khổng lồ đầy mập mờ và lỗi.”

“Ðược rồi, sửa cái này dễ thôi.” tôi trả lời. Tôi tách một hàm mới gọi là createTestFile() và thay đổi cả testCountBytesInFile() lẫn testSendFile() để gọi hàm này.

private File createTestFile(String name, String content) throws IOException {

File f = new File(name);

FileOutputStream stream = new FileOutputStream(f);

stream.write(content.getBytes());

stream.close();

return f;

}

Tôi chạy thử kiểm thử để chắc ăn là không làm hỏng gì cả, rồi tiếp tục viết phần kiểm thử. Tôi biết nó cần giả lập main(), cho nên tôi gọi những hàm mà main() cần gọi. Thế rồi tôi thêm vào phần gọi cuối để gởi tệp tin đi.

public void testSendFile() throws Exception {

File f = createTestFile("testSendFile", "I am sending this file.");

c.setFilename("testSendFile");

assertTrue(c.connect());

assertTrue(c.prepareFile());

assertTrue(c.sendFile());

}

“Tốt,” Jerry gật đầu. “Mày liệu trước là mình sẽ cần một phương thức phía client có tên là sendFile().

“Ðúng vậy,” tôi nói. “Phương thức này sẽ gởi tệp tin được chuẩn bị trước.”

Tôi trở lại với phần kiểm thử và bị trở ngại. Làm sao tôi kiểm nghiệm được tệp tin tôi tạo ra và “gửi đi” thật sự được gởi đến server trong khi chúng tôi chẳng có tệp tin nào? Phải chăng tôi cần viết cả phần server trước khi tôi có thể kiểm nghiệm chuyện này? Tôi định kiểm thử gì vậy nhỉ?

Tôi ngồi yên khoảng một phút trong khi Jerry nhìn tôi chờ đợi. Thế rồi khi tôi xoay qua và giải thích điểm khó khăn.

Gã giải thích “Không, mày không cần phải viết cái server”. “Chúng ta chỉ kiểm thử mỗi khả năng gửi tệp tin của client, chớ chẳng phải khả năng nhận tệp tin của server.”

“Nhưng làm sao tôi gửi tệp tin trong khi chẳng có server để nhận?”

“Mày có thể tạo ra “stub” server (server sơ khai) chỉ làm tối thiểu công việc mày cần thôi,” Jerry trả lời. “Nó chẳng cần phải thực sự nhận tệp tin – nó chỉ tiếp báo là mày đã gửi tệp tin đúng cách.”

“Hừm… như thế này chăng?”

assertTrue(server.fileReceived);

“Như vậy được rồi,” Jerry gật đầu. “Bây giờ làm cho cái kiểm thử đạt đi.”

Tôi nghĩ về vấn đề này và nhận ra nó không quá khó, thế nên tôi viết một cái server giả chẳng làm gì hết:

class TestSMCRServer implements SocketServer {

public boolean fileReceived = false;

public void serve(Socket socket) {

}

}

Jerry nói, “À, lại thêm trùng lặp!”

Tôi xem lại và thấy trước rằng trước giờ nghỉ chúng tôi đã cài đặt một server giả tương tự trong phương thức testConnectToSMSRemoteServer() – thế nên tôi loại bỏ nó.

public void testConnectToSMCRemoteServer() throws Exception {

boolean connection = c.connect();

assertTrue(connection);

}

Thế rồi tôi bắt đầu phần server với phương thức SetUp() trong phần kiểm thử và đóng nó bằng phương thức TearDown(). Trước khi mỗi phương thức của kiểm thử được gọi, server khởi động; khi phương thức của kiểm thử trả ngược về, nó đóng lại.

protected void setUp() throws Exception {

c = new SMCRemoteClient();

server = new TestSMCRServer();

smc = new SocketService(SMCPORT, server);

}

protected void tearDown() throws Exception {

smc.close();

}

Cuối cùng tôi viết phương thức giả sendFile() trong SMCRemoteClient:

public boolean sendFile() {

return false;

}

}

Mấy cái kiểm thử bị hỏng. Tôi thở dài. “Làm gì bây giờ?”

“Gửi cái tệp tin đi,” gã nói.

“Ý ông là chỉ mở tệp tin ra và tống nó qua socket sao?” Tôi hỏi.

Jerry nghĩ một lát và nói “Không, có lẽ mình cần cho server biết để tiếp nhận tệp tin. Cho nên hãy gửi một thông điệp đơn giản và gửi tiếp theo đó nội dung tệp tin.”

Jerry lấy bàn phím vào thay đổi SMCRemoteClient như sau. “Ðầu tiên, mình cần lấy cái “stream” (luồng) ra từ socket,” gã nói.

public boolean connect() {

boolean connectionStatus = false;

try {

Socket s = new Socket("localhost", 9000);

is = new BufferedReader(new InputStreamReader(s.getInputStream()));

os = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));

connectionStatus = true;

} catch (IOException e) {

e.printStackTrace();

connectionStatus = false;

}

return connectionStatus;

}

“Sau đó chúng ta phải chuẩn bị tệp tin sẵn sàng để đọc”

public boolean prepareFile() {

boolean filePrepared = false;

File f = new File(itsFilename);

if (f.exists()) {

try {

itsFileLength = f.length();

fileReader = new BufferedReader(

new InputStreamReader(new FileInputStream(f)));

filePrepared = true;

} catch (FileNotFoundException e) {

filePrepared = false;

e.printStackTrace();

}

}

return filePrepared;

}

“Sau cùng,” gã nói, “chúng ta có thể gửi tệp tin.”

public boolean sendFile() {

boolean fileSent = false;

try {

writeSendFileCommand();

fileSent = true;

} catch (Exception e) {

fileSent = false;

}

return fileSent;

}

private void writeSendFileCommand() throws IOException {

os.println("Sending");

os.println(itsFilename);

os.println(itsFileLength);

char buffer[] = new char[(int) itsFileLength];

fileReader.read(buffer);

os.write(buffer);

os.flush();

}

“Ừ!” Tôi nói. “Đó là rất nhiều mã mà không có kiểm thử.”

Jerry ngượng ngùng nhìn tôi vào nói “Ừa, tao cũng run lắm.”

Gã nhấn nút kiểm thử và phần kiểm thử bị hỏng vì server.fileRecieved trả lại sai (false). “Ui cha!” Jerry nói. “Mình tránh nhịp đập [Muyon](https://tapchilaptrinh.vn/tho-lanh-nghe-12-ba-dong-ma-xau-xi-smcremote-phan-2/#_2_) đó!”

“Thế,” tôi nói. “Ông sẽ xử lý tệp tin với ba dòng. Dòng thứ nhất gồm string “Sending”, dòng thứ hai gồm tên tệp tin và dòng thứ ba gồm chiều dài của tệp tin. Sau đó, ông gởi tệp tin theo dạng chuỗi ký tự.”

“Rồi. Tao đã nói với mày trước buổi giải lao là mình cần chiều dài của tệp tin rồi mà.”

“Hừm, tôi đoán thế” tôi nói.

“Bây giờ mình chỉ cần nhận tệp tin từ server giả. Mày muốn thử một phát không?” Jerry hỏi.

Tôi khá chắc nên phải làm gì. Nên đầu tiên tôi đổi cái kiểm thử để chắc ăn chúng tôi có tên tệp tin, chiều dài và nội dung tệp tin:

public void testSendFile() throws Exception {

File f = createTestFile("testSendFile", "I am sending this file.");

c.setFilename("testSendFile");

assertTrue(c.connect());

assertTrue(c.prepareFile());

assertTrue(c.sendFile());

Thread.sleep(50);

assertTrue(server.fileReceived);

assertEquals("testSendFile", server.filename);

assertEquals(23, server.fileLength);

assertEquals("I am sending this file.", new String(server.content));

f.delete();

}

Kế tiếp tôi đổi cái server giả cho nó phân giải dữ liệu vào và bảo đảm nó đúng:

class TestSMCRServer implements SocketServer {

public String filename = "noFileName";

public long fileLength = -1;

public boolean fileReceived = false;

private PrintStream os;

private BufferedReader is;

public char[] content;

public String command;

public void serve(Socket socket) {

try {

os = new PrintStream(socket.getOutputStream());

is = new BufferedReader(new InputStreamReader(socket.getInputStream()));

os.println("SMCR Test Server");

os.flush();

parse(is.readLine());

} catch (Exception e) {

}

}

private void parse(String cmd) throws Exception {

if (cmd != null) {

if (cmd.equals("Sending")) {

filename = is.readLine();

fileLength = Long.parseLong(is.readLine());

content = new char[(int)fileLength];

is.read(content,0,(int)fileLength);

fileReceived = true;

}

}

}

}

Cuối cùng tôi điều chỉnh SMCRemoteClient.connect() để nó đợi thông điệp SMCR được gởi từ server giả:

public boolean connect() {

...

String headerLine = is.readLine();

connectionStatus = headerLine != null &&

headerLine.startsWith("SMCR");

...

}

Tôi không gõ hết những thứ trên cùng một lúc. Tôi không muốn có một nhịp đập Muyon. Nên tôi thay đổi từng bước nhỏ, chạy kiểm thử giữa mỗi thay đổi. Tôi biết Jerry có ấn tượng tốt, đặc biệt vì gã còn bị quê chuyện thay đổi to lớn ở trên. Sau cùng, khi mọi kiểm thử đều đạt, tôi cảm thấy hơi hưng phấn một tí, tôi đánh liều bằng một nhận xét.

“Jerry,” tôi nói. “Ðoạn mã này xấu xí quá.”

“Ý mày thế nào?” gã hỏi.

“Hèm, gửi ba dòng: tên tệp tin, chiều dài và dòng “Sending”.”

Jerry nhìn tôi một cách nhún nhường. “Cứ cho là mày biết cách hay hơn.”

“Tôi nghĩ thế.” tôi mỉm cười và bắt đầu gõ….

Bạn có thể tải mã nguồn mà Jerry và Alphonse đã hoàn thành ở [đây](http://www.objectmentor.com/resources/craftman/Craftsman_12_SMC_II_ThreeUglyLines.zip).

-1- Cherenkov display: thuộc nghiên cứu vật lý cao cấp. Một đề tài hết sức thú vị và được nhiều nhóm nghiên cứu quan tâm. Có một bản pdf phân tích Chrenkov display rất cụ thể ở: www.lip.pt/~varela/projfc/Showers/Auger-3.pdf. Ngoài ra còn có vô số tài liệu về vấn đề này trên Internet cho những ai thích đào sâu.

-2- Muyon: Một “muyon” là một phân tố không ổn định trong vùng phản xạ gần bề mặt trái đất. Nó có trọng lượng hơn 207 lần trọng lượng một electron và tồn tại trong cả thể dạng âm hoặc dương.

**Bài tiếp:** [Thợ lành nghề #13: Đối tượng (SMCRemote – phần 3)](https://codegym.vn/blog/2018/08/11/tho-lanh-nghe-13-doi-tuong-smcremote-phan-3/)

**Bài trước:** [Thợ lành nghề #11: Dùng hàm main để làm gì? (SMCRemote – phần 1)](https://codegym.vn/blog/2018/08/09/tho-lanh-nghe-11-dung-ham-main-de-lam-gi-smcremote-phan-1/)

**Tác giả:** Robert C. Martin

**Người dịch:** Hoàng Ngọc Diêu (conmale)