The Craftsman: 23
SMCRemote Part XIII
Raggedy.

Robert C. Martin
19 February 2004

...Continued from last month. You can download last month's code from:
www.objectmentor.com/resources/articles/CraftsmanCode/Craftsman_22_BugEye.zip

---

20 Feb 2002, 14:00

Avery looked at me with a smug smile on his pimply face. His bushy red eyebrows added intensity to the look. He had gained confidence over the last hour, and his stammering had disappeared. "OK, so now the server executes compiles in an empty directory. Now I think its time to test the whole server from end to end."

"You mean you want to open a socket with the server, send in a `CompileFileTransaction` and see if the appropriate `CompilerResponseTransaction` comes out?"

"Precisely, Alphonse. Precisely." As he gained confidence he started speaking with an odd formality. It was a bit strange, but also a bit fun. Playing along I said: "I see, Avery. Then I suggest that you compose the appropriate test case, and then I'll endeavor to satisfy it."

"Excellent suggestion, Alphonse. Shall I proceed?"

"Please do."

Avery faced the screen, making a show of stretching his fingers and squaring his shoulders before touching the keyboard. He typed:

```
public void testServerEndToEnd() throws Exception {

}
```

He paused with mock intensity and said: "Do you agree with the name?"

"Indeed!" I replied. I was getting into this. He continued to type while he talked.

"Our first goal should be to instantiate the `SMCRemoteServer` and bind it to some port number that is convenient for our test case."

```
public void testServerEndToEnd() throws Exception {
   SMCRemoteServer server = new SMCRemoteServer(999);
}
```

"I appreciate the intent of that code, but notice that it fails to compile. Apparently there is no constructor that takes an `int`."

"Well spotted, Alphonse. Well spotted. We must therefore create just such a constructor."

```
public class SMCRemoteServer {
   …
 public SMCRemoteServer(int port) {
 }
   …
}
```

"Nicely done, Avery. I suggest that this test will pass in its current form."

"I believe you may be correct, Alphonse. Shall we try?"

"Yes, I believe we should."

Avery pushed the test button and the green bar flashed on the screen indicating that the test had passed.

"As expected." Avery said with a self-satisfied tone. "Next we'll have our test connect to that socket like so:"

```
public void testServerEndToEnd() throws Exception {
  SMCRemoteServer server = new SMCRemoteServer(999);
  Socket client = new Socket("localhost", 999);
}
```

"I expect, Avery, that this will fail."

"I agree, Alphonse. We never created the server socket." Pushing the test button quickly yielded a red bar on the screen. We looked at each other and nodded, pleased with our mutual prescience.

I held my hands out in front of me, as a polite gesture requesting Avery to slide the keyboard in my direction. He did so with a flourish and a knowing look.

"Thank you Avery. To make this test pass we'll use a utility named `SocketService` that Jerry and I wrote last week."

```
public SMCRemoteServer(int port) throws Exception {
  SocketService service = new SocketService(port, new SocketServer() {
    public void serve(Socket theSocket) {
      try {
        theSocket.close();
      } catch (IOException e) {
      }
    }
  });
}
```

"Nicely done, Alphonse. Shall I run the test?"

"Certainly!" I slid the keyboard back to him.

Again we smiled and nodded – making a show of it -- as the green bar flashed on the screen.

"Now then," harrumphed Avery, "The server should respond with a connection message."

"Yes indeed it should, Avery." I said, remembering back to last Thursday. "Jerry and I decided that the message would be a string that begins with SMCR."

"Well then, that's easy enough to test."

```
public void testServerEndToEnd() throws Exception {
  SMCRemoteServer server = new SMCRemoteServer(999);
  Socket client = new Socket("localhost", 999);
  ObjectInputStream is = new ObjectInputStream(client.getInputStream());
  String header = (String) is.readObject();
  assertTrue(header.startsWith("SMCR"));
```

```
}
```

"Yes, that looks quite proper. Quite proper." I said accepting the keyboard as Avery brandished it in my direction. I ran the test, and noted with satisfaction as it failed due to an `EOFException`. Now then, to make this pass we merely create the appropriate string and send it out the socket – like so!"

```
public SMCRemoteServer(int port) throws Exception {
  SocketService service = new SocketService(port, new SocketServer() {
    public void serve(Socket theSocket) {
      try {
        ObjectOutputStream os =
          new ObjectOutputStream(theSocket.getOutputStream());
        os.writeObject("SMCR");
        theSocket.close();
      } catch (IOException e) {
      }
    }
  });
}
```

The test produced a green bar, and we repeated the smile and nod ritual. I passed the keyboard back to Avery holding it as though I were passing a fencing foil to my opponent at the start of a match. "Your weapon sir."

"At your disposal, sir." He responded. "I believe we must now prepare to send the `CompileFileTransaction`. This will require us to first write a source file containing code that the SMC compiler can compile."

"Ah, indeed, Jean and I created such a file just this morning. The code is in the `testExecuteCommand()` function. We should be able to extract it into a function of its own named `writeSourceFile()`."

```
private File writeSourceFile(String theSourceFileName) throws IOException {
  File sourceFile = new File(theSourceFileName);
  PrintWriter pw = new PrintWriter(new FileWriter(sourceFile));
  pw.println("Context C");
  pw.println("FSMName F");
  pw.println("Initial I");
  pw.println("{I{E I A}}");
  pw.close();
  return sourceFile;
}
```

"Excellent! Thank you Alphonse. So now I can create a `CompileFileTransaction`, send it to the server, and expect a `CompilerResultsTransaction` back – right?"

"I'd say that was correct, Avery."

```
public void testServerEndToEnd() throws Exception {
  SMCRemoteServer server = new SMCRemoteServer(999);
  Socket client = new Socket("localhost", 999);
  ObjectInputStream is = new ObjectInputStream(client.getInputStream());
  ObjectOutputStream os = new ObjectOutputStream(client.getOutputStream());
  String header = (String) is.readObject();
  assertTrue(header.startsWith("SMCR"));
  File sourceFile = writeSourceFile("mySourceFile.sm");
  CompileFileTransaction cft = new CompileFileTransaction("mySourceFile.sm");
  os.writeObject(cft);
  os.flush();
  CompilerResultsTransaction crt =
```

```
    (CompilerResultsTransaction) is.readObject();
    assertNotNull(crt);
  }
```

With the red bar from the failing test on the screen, Avery said. "Alphonse, do you agree that this code expresses my intent."

"Yes, I think it expresses it very well. And now if you'll hand me the keyboard, I'll make it pass."

```
    public void serve(Socket theSocket) {
      try {
        ObjectOutputStream os =
          new ObjectOutputStream(theSocket.getOutputStream());
        ObjectInputStream is =
          new ObjectInputStream(theSocket.getInputStream());
        os.writeObject("SMCR");
        os.flush();
        CompileFileTransaction cft =
          (CompileFileTransaction) is.readObject();
        CompilerResultsTransaction crt = new CompilerResultsTransaction();
        os.writeObject(crt);
        os.flush();
        theSocket.close();
      } catch (Exception e) {
      }
    }
```

"And sure enough, pass it does."

"Well done Alphonse!"

"Thank you Avery. There really isn't anything to it."

"In that case, I'll give you one more challenge." He took the keyboard and began to type again.

```
  public void testServerEndToEnd() throws Exception {
    SMCRemoteServer server = new SMCRemoteServer(999);
    Socket client = new Socket("localhost", 999);
    ObjectInputStream is = new ObjectInputStream(client.getInputStream());
    ObjectOutputStream os = new ObjectOutputStream(client.getOutputStream());
    String header = (String) is.readObject();
    assertTrue(header.startsWith("SMCR"));
    File sourceFile = writeSourceFile("mySourceFile.sm");
    CompileFileTransaction cft = new CompileFileTransaction("mySourceFile.sm");
    os.writeObject(cft);
    os.flush();
    CompilerResultsTransaction crt =
      (CompilerResultsTransaction) is.readObject();
    assertNotNull(crt);
    File resultFile = new File("F.java");
    assertFalse(resultFile.exists());
    crt.write();
    assertTrue(resultFile.exists());
  }
```

"Aha!" I said. "You actually want me to invoke the compiler, eh?"

"That I do, my dear Alphonse. That I do."

"Then prepare yourself for compilation, young Avery."

```
    public void serve(Socket theSocket) {
      try {
        ObjectOutputStream os =
          new ObjectOutputStream(theSocket.getOutputStream());
```

```
        ObjectInputStream is =
          new ObjectInputStream(theSocket.getInputStream());
        os.writeObject("SMCR");
        os.flush();
        CompileFileTransaction cft =
          (CompileFileTransaction) is.readObject();
        String command = buildCommandLine(cft.getFilename());
        CompilerResultsTransaction crt = compile(cft, command);
        os.writeObject(crt);
        os.flush();
        theSocket.close();
      } catch (Exception e) {
      }
    }
```

We both leaned over the monitor as I pushed the test button.

Green Bar. Smile. Nod.

"We've done well, Avery. But I think it's time we cleaned this code up a bit."

"I quite agree, Alphonse, it has gotten a little, shall we say, raggedy?"

Together we worked through the new server code and split it up into a batch of smaller methods that worked together to tell the story. When we were done it looked like this; and all the tests still passed.

```
public SMCRemoteServer(int port) throws Exception {
  SocketService service =
    new SocketService(port, new SMCRemoteServerThread());
}

  private static class SMCRemoteServerThread implements SocketServer {
  private ObjectOutputStream os;
  private ObjectInputStream is;
  private CompileFileTransaction cft;
  private CompilerResultsTransaction crt;

  public void serve(Socket theSocket) {
    try {
      initializeStreams(theSocket);
      sayHello();
      readTransaction();
      doCompile();
      writeResponse();
      theSocket.close();
    } catch (Exception e) {
    }
  }

  private void readTransaction() throws IOException, ClassNotFoundException {
    cft = (CompileFileTransaction) is.readObject();
  }

  private void initializeStreams(Socket theSocket) throws IOException {
    os = new ObjectOutputStream(theSocket.getOutputStream());
    is = new ObjectInputStream(theSocket.getInputStream());
  }

  private void writeResponse() throws IOException {
    os.writeObject(crt);
    os.flush();
  }

  private void sayHello() throws IOException {
```

```
    os.writeObject("SMCR");
    os.flush();
  }

  private void doCompile() throws Exception {
    String command = buildCommandLine(cft.getFilename());
    crt = compile(cft, command);
  }
}
```

"OK, that's much better. I think it's time for a break, don't you?" I said to Avery.

"Yeah, I guess we've been at it for a couple of hours. Let's go to the game room and play LandCraft."

To be continued...

The code that Alphonse and Avery finished can be retrieved from:
www.objectmentor.com/resources/articles/CraftsmanCode/Craftsman_23_Raggedy.zip