Preparation for a Job Interview at Advantest Software R&D

Purpose

As preparation for a job interview at Advantest software R&D, we would like you to spend some time on the assignment described below and send us the results.

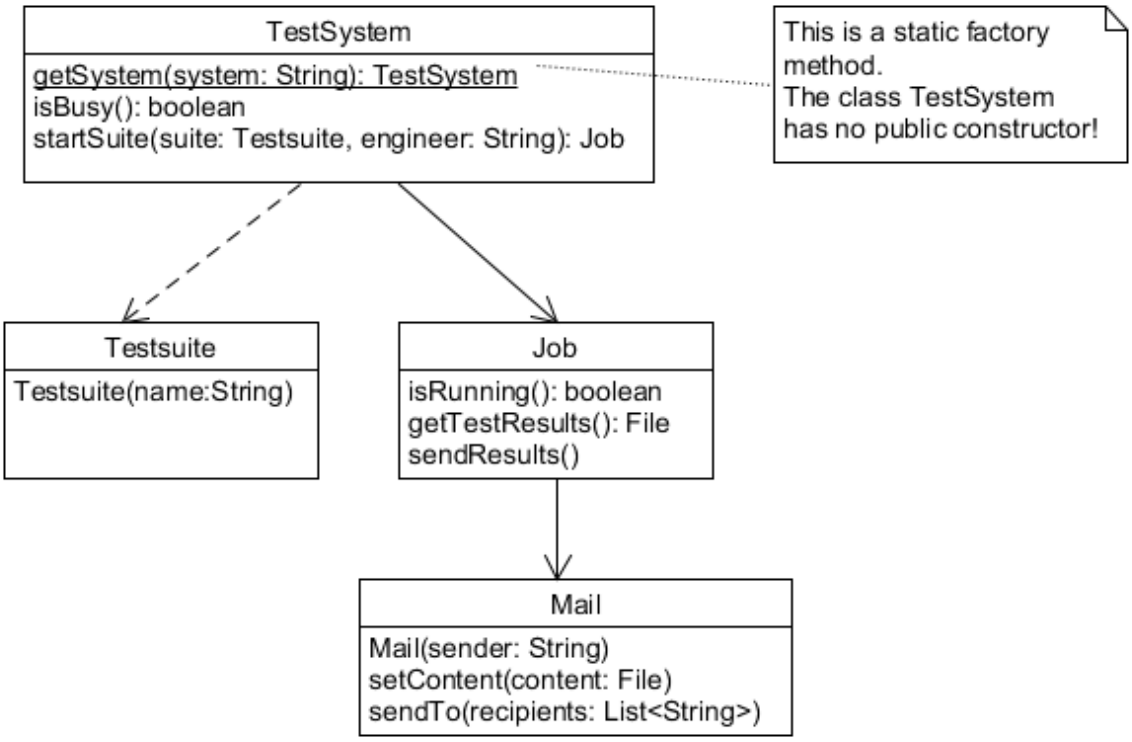
Why are we doing this? We would like to learn how you approach software development, about your technical skills and experiences. Therefore we need an artifact to talk about. Constructing software takes time and pondering, that's why handing out some programming puzzles during the interview will not work well. So, take your time. We don't expect a polished presentation, though. But what you show us should be well thought out and communicate how you could really build this application. The design will be the basis for the discussion during the job interview.

Situation

Suppose you are a member of a software development team. Your product is a security application. It runs on several operating systems (Windows XP, Windows 7, Windows 8, Windows 10, …) and it can control several hardware devices (cameras, electronic locks, automatic doors, …). For every module of your application, there is a test suite. So, when you change a module you are obliged to run the associated test suite. But this is a bit tricky: First, you must determine on which operating systems the test suite can be executed. Then you must determine which hardware devices are required to run the test suite. And finally, you must find an unoccupied test system in the test lab with the right operating system and all necessary hardware devices attached so that you can run the test suite there.

You decide to do your team a big favor by developing a tool that simplifies this process. When called with a test suite (e.g. on the command line: `scheduletest HomeSecuritySimple`), this tool (scheduletest) determines the possible operating systems (e.g. Windows 7 and 10) and the required hardware devices (e.g. camera and electronic lock) of this test suite (HomeSecuritySimple). Then it searches in some kind of database for an appropriate test system (e.g. test system Jupiter that runs Windows 7 and that has a camera, electronic locks and smoke sensors connected to it). If no such test system exists, the tool flags an error. Otherwise, it queues the test suite somehow. As soon as an appropriate test system is available, the test suite is dispatched to this test system and executed on it, while the tool returns. When the test suite is finished, the test results are sent by email to the person who scheduled the test.

Your team loves this idea and your boss approves to spend a limited amount of effort on this tool. So it has to be rather simple. A few team members offer help. Jerry has already implemented the code to run test suites on any test systems, locally or remotely, and to send the test results by mail. He shows you this UML diagram and a bit of example code. If necessary, he'd be more than willing to rewrite his code or adapt it to what you need.



To run the test suite HomeSecuritySimple on test system Jupiter, you could use this code (it's Java code, but if you don't do the assignment in Java assume it's written in the language you have chosen):

```
// Returns login name of the current user – email is sent there
String engineer = System.getProperty("user.name");
TestSystem system = TestSystem.getSystem(“Jupiter”);
Job testExecution;
if (!system.isBusy()) {
    Testsuite suite = new Testsuite(“HomeSecuritySimple”);
    // test suite can be started only if the system is not busy
    testExecution = system.startSuite(suite, engineer);
}
```

To notify the engineer when the test has finished, you simply need to run this code:

```
if (!testExecution.isRunning()) {
    sendResults();
}
```

Helen, another team member, offers to write the database access. You just need to tell her which interfaces you would like to use to access the data. The database stores this information:

- The test system names along with the operating system installed on it and the connected hardware devices.
- The test suite names along with possible operating systems and the list of hardware devices required to run the test suite.

Assignments

1. Please describe how you would structure the software for this tool and how scheduling a test suite would work in this design. You can use diagrams (UML, lines & boxes, …), code, pseudo-code, text, videos, telepathy, … – whatever works to communicate the design. Hand-drawn diagrams are fine, don't waste time or even money for sophisticated tools.
   Details like how to parse the command line or how to find out who the current user is or how to start services/daemons on certain operating systems should be ignored. Also don't get carried away, for example, with designing sophisticated interprocess communication protocols. Assume there is somebody who can do the implementation for you, as soon as you can describe the interface and its behavior.
2. Somebody will implement the database access to find the test systems and their capabilities or the test cases and their requirements respectively. Please describe the interfaces you want this person to provide for you.
3. Please write the code that determines on which test system the test suite will be executed. Choose a language you are familiar with and write it in a style that you would consider good coding standard. You should use here the interfaces defined in 2.
4. Please explain how you would approach testing this tool. A few test cases might be helpful as examples, but please don't bother writing an extensive test plan.

If you have any questions, please don't hesitate to contact us! Note there is no right or wrong solution. It's all about judgment.

Technical Details

If this description is too vague for you to provide a solution, here are some more details:

- There are about a dozen test systems, all accessible over the network. Every developer can use them.
- The development team consists of 20 people.
- There are about 100 test suites. The quickest one does its job within 2 minutes, the longest one takes roughly 25 minutes. Occasionally new test suites are added or existing ones are renamed, split or removed.
- Development systems and test systems run an operating system you are familiar with. Dealing with the peculiarities of operating systems consumes quite some time of a developer's professional life but it's not part of this exercise here.
- There are about 50 hardware devices supported by the security application. Occasionally new ones are added.
- The test suites are implemented as scripts stored in a repository of a version control system like Git or SVN. To run them, the latest version is checked out from the repository and the scripts are called.
- Assume you have administrator permissions on all systems and thus can install and run anything. But don't bother with administrative tasks like installing the security application. "I create a service which does …" or "I configure a cron job which does …" is detailed enough.