PART I
Navigating the Interview Process

Part I covers many of the essentials for a technical developer interview, from writing an appropriate resume to more technical topics, regardless of any language specifics.

Chapter 1 discusses the different types of interviews that are used for assessing candidates.

Chapter 2 explains how to write an appropriate resume and cover letter, and highlights the points that interviewers often look for when reading resumes.

Chapter 3 describes how to approach the different interview types.

Chapter 4 covers some basic algorithms, such as sorting and searching. These are popular topics for an interview and are essential for understanding in a day-to-day programming job.

Chapter 5 covers some important data structures. These too are often discussed in interviews, and knowledge of these is essential in most development roles.

Chapter 6 is on design patterns. Design patterns are often a favorite interview topic.

Chapter 7 looks at some useful interview questions, which are often used across a variety of programming languages, not just for Java interviews.

1

Dissecting Interview Types

Simply put, employers conduct interviews to recruit new talent, or to fill a necessary gap to improve a team's productivity. Within a development team or department, you will find a wide range of skills, and on the whole, this is key to making a team gel. It is simply not possible for one person to develop and manage a professional-grade application, there is too much to manage: developing feature requests for product owners, maintaining test environments, and answering any ad-hoc queries from the operations team are all daily tasks for the team managing the development of an application. Development teams will often need one or several application developers. They may even have dedicated front-end developers and database developers. Some teams are even lucky enough to have a dedicated build manager.

Varying ranges of experience are important, too. Having developers who have put several applications live and supported them is key to any new project's success. Graduates, or developers with a couple of years' experience, are also vital: These employees can often bring a different perspective and approach to developing an application, and the more experienced team members can mentor and coach them in the delicate process of developing large enterprise-scale applications within a team of many people.

When it comes to interviewing for new talent, the process and experience for the interviewee can be quite different from company to company. Understandably, it falls on the shoulders of the team's developers to interview for new developers. First and foremost, a developer's day job is to write and produce tested, working applications, and not to interview people. As a result, a developer interviewing people for additional roles may often be under-prepared, and perhaps even uninterested in performing an interview. In a face-to-face interview, this is going to be one of the first hurdles to cross: You need to make the interviewer interested in you.

Companies, and especially technology and digital companies, are getting much better at realizing how important the recruitment process is. Some of the more progressive employers often put emphasis on recruitment within internal company objectives. This puts the responsibility on the employees to make the company an attractive place to work, with the hope that this will attract the top talent, which in turn will bring productivity, success, and profitability.

The first impression for an interviewer will be the resume, sometimes called a *curriculum vitae*, or *CV*. How to make an eye-catching resume is covered in the next chapter.

As you move through the interview process for a particular role or company, you will encounter different styles and methods of interviews. Generally, the "interview pipeline" for a candidate is designed to be as efficient as possible for the employer, with the face-to-face interviews coming late in the process.

The interview pipeline will usually start with a phone screening, followed by one or more technical tests, and finally some face-to-face interviews.

LOOKING AT THE PHONE SCREENING PROCESS

Companies often start the interview process with a telephone screening. This is advantageous from the company's side, because it can take a lot of people to organize a face-to-face interview: finding available times for all the interviewers, HR, and possibly a recruitment team, meeting rooms, and so on. This is also helpful for you, because you won't need to take much time away from work. Many people like to keep it quiet from their current employer that they are looking for jobs, so it shouldn't be too hard to find a quiet corner of the office or a meeting room for an hour to take a phone interview.

If you do an interview over the telephone, make sure you are prepared well before the time of the interview. If you *have to* take the call while at work, book a quiet meeting room or find a quiet corner of the office. If that is not possible, you could go to a local coffee shop. Make sure beforehand that the noise level is appropriate: You don't want to be distracted by baristas calling across the café or loud music playing in the background.

If you are expecting to do a remote live-coding exercise, make sure you have Internet access wherever you are. Have a pen and paper handy in case you want to make notes during the call for questions to ask later. Use a hands-free kit when typing: It will make the call much clearer for the interviewer. You don't want to waste precious time repeating questions or re-explaining your answers—this is very frustrating for all involved.

It might not even hurt to have a few notes in front of you for any topics that may come up during the call. This is not cheating; it can help calm any nerves and get you settled into the uncomfortable, alien experience of having to convey technical explanations over the phone to someone you have never met before. Note, however, that if and when you have an interview face-to-face with the team, you won't be able to have any notes in front of you.

Usually a phone screen lasts for 30 to 60 minutes, and you should expect some very high-level questions. The kinds of questions that come up in a telephone interview are often about language-agnostic algorithms. You may be asked to verbally describe these, or you may even be asked to attempt some live-coding challenges in a shared, collaborative document editor, such as Google Docs, or perhaps a bespoke interview-hosting website, such as Interview Zen (`www.interviewzen.com`).

At any point in the interview process you should know the basics about a company. Look at the "about us" page on their website. Read their blog; find out if they have a Twitter account. If you are applying for a small company or a startup, try to find out some more about the senior members

of the company, such as the CEO and CTO. They can often be quite active in a local development community.

At the end of any interview, the interviewer will often ask if you have any questions for them. Answering *no* to this question will leave a bad impression; it will show that you do not care enough about the role. Considering you now know that this question will more than likely come up, think about what questions you want to ask long before you even go for the interview. Think about what you want to know about the team dynamic, how the team works together, and what the office environment is like. Only you can come up with these questions. This is not the time to talk about the salary or anything else to do with the package that comes with the job. There will be plenty of time to talk about that once you have been offered the role.

REVIEWING TECHNICAL TESTS

A technical test can be used as a supplement to or instead of a phone-screen interview. This may even happen as part of a phone screening, or you may be invited to the office for a technical test.

A technical test often consists of some simple questions covering a multitude of areas related to the nature of the role. If the role is for working on a web application, you could expect questions about the Servlet API and perhaps some frameworks such as Spring MVC or Tomcat. You should be aware of the nature of the role, and any languages and frameworks used.

These tests are usually performed alone, and can take a variety of approaches. Some interviewers rely on a pen-and-paper test, often asking some simple definite-answer questions, or others may ask you to write a simple algorithm, often around 10 to 20 lines long.

If you find yourself writing code on paper, you should take extra effort to make sure the code is understandable. Any decent employer should understand that you may make some elementary mistakes around method naming, or miss a semicolon, but you should always aim to give no excuse to an interviewer to not ask you back for any further interviews.

Another popular technique is to provide a coding test on a computer with a functioning IDE. This is a fairer way of testing, because it is a similar environment to how candidates work when doing their day job. They may not provide Internet access, but may provide offline Java documentation instead.

When you are tested with a fully functioning IDE, or even just a compiler, you have no excuse for producing code that does not compile.

Whatever you are asked to do, write unit tests. Write them *first*. Although this may take additional time, it will ensure that the code you do write is correct. Even if a test makes no mention of writing unit tests, this will show to anyone examining your test that you are diligent and have an eye for detail. It will show that you take pride in your work, and that you think for yourself.

For example, method names in JUnit 4 and onward are free from any kind of naming convention (as discussed in Chapter 9), so if you saw a question similar to the following: *Write a simple algorithm to merge two sorted lists of integers*, you could quickly start by writing a list of test cases, and these will then become the method names for your JUnit tests.

Covered in depth in Chapter 9, method names in JUnit 4 and onward are free from any kind of naming convention, so you could quickly start by writing a list of test cases, making them method names for JUnit tests. Some example test cases for the given question include:

```
twoEmptyLists
oneEmptyListOneSingleElementList
oneEmptyListOneMultipleElementList
twoSingleElementLists
oneListOfOddNumbersOneListOfEvenNumbers
oneListOfNegativeNumbersOneListOfPositiveNumbers
twoMultipleElementLists
```

The contents of each of these test cases should be easy to understand from the name. Given a well-defined API, writing the test cases should take no more than five to ten minutes, and you can run these tests as a frequent sanity check when implementing the actual merge. This will keep you on track as you are writing and refactoring your real code.

These given test cases probably do not cover every possible code path for merging two sorted lists, but should give you and the interviewer confidence in your actual implementation.

Most modern IDEs allow you to automatically import the JUnit JAR without needing any Internet access at all; it comes bundled with the IDE. Make sure you know how to do this for a few IDEs, because you may not get a choice of which one to use in a test situation. At a bare minimum, you should be able to do this for Eclipse and IntelliJ. They are both smart enough to recognize the `@Test` annotation and prompt you to import the JUnit JAR.

Try to write test cases concisely and quickly. Any reasonable interviewer will not mind the odd missed test case, understanding that you want to show the range of your abilities in the time given.

If you are ever given a paper test and a test on a computer, you have the advantage of being able to check your code on paper by writing it on the laptop! Some companies do take this approach—they will provide some simple, exploratory Java questions on paper, and then ask you to write a more in-depth, small application.

Considering you are applying for a job as a developer, it is reasonable to expect to demonstrate your technical ability during the interview process. It is worth noting that this is not always the case—interview standards can be so varying that it is not unheard of to have a brief chat with a hiring manager, and that is all. This approach does not normally work out well for the employer, and thankfully this does not happen as often as it used to.

Like all parts of the interview process, you must be prepared. The job specification page should contain a lot of technical information about what languages, technologies, and frameworks are used. A job specification page should be available on the company's recruitment website pages, or if you cannot find it, you should ask the company for it. You should not be expected to know every bullet point on a job specification inside and out, but the more you can show, the better. Employers will be looking for a willingness to learn and adapt, so make sure you convey this in an interview. Most importantly, you must *want* to learn the technologies the company uses.

Again, understand what the company does; you may even be asked specifically for what you understand about the company, or specific applications developed by the team you are interviewing for. If

the company has a public-facing website, make sure you have used it. If the role is part of a game, make sure you have played it. Some employers have a rule that candidates are a definite "no" if they have not used the product developed by the team.

HANDLING FACE-TO-FACE INTERVIEWS

Following a successful technical test, you should be asked for a more personal interview, meeting members of the immediate team you would be working with as well as a hiring manager, and perhaps some members of a recruitment team or human resources. This can often happen over the course of half a day, or a full day. It may even be set over several meetings on several days, usually depending on the office location—how easy it is for you to keep returning for more interviews—the availability of the interviewers, and your availability, too.

As a rule, the interviews will get less technical as the process progresses. A typical set of interviews may be as follows:

- A technical interview, covering any work done in a pre-screening or an individual test
- Several more technical interviews, covering more topics and questions of interest to the hiring team
- You may be expected to write some code on a whiteboard or to do some pair programming.
- An interview with one or several members of the business side of the team, typically a product manager or project manager
- An interview with a hiring manager. This will cover some softer skills, away from the technical side. This is to see if you are a good fit for the team.
- A final debrief from an in-house recruiter or HR representative. Salary expectations and other contractual conversations may happen here, as well as any information about next steps.

In preparation for any upcoming interview, try to think about possible questions that will be asked. If you are expecting to cover any code you wrote in a phone screening, online, or in a previous technical test, try to remember what you actually wrote. It may help to try to write out from memory any small applications that you wrote as part of your test. It does not matter if it is not exactly as you submitted it the first time, but it will help jog your memory, and you may find some improvements to discuss in the interview.

When doing any face-to-face interview, be prepared to be stretched to the limit of your ability. Answering with "I don't know" is not an admission of failure; it is simply that the interviewer is trying to understand the limits of your knowledge. Of course, the later you reach that point, the better. However, you cannot expect to be an expert in all domains, so there should not be much concern if you can demonstrate a deep understanding of, say, Spring and Hibernate, but your knowledge of database transactions and concurrency is not up to the same level.

The interview process itself is not perfect. This is not an examination; just knowing the right answer is not enough. The interview team is checking to see if you will be a valuable addition, perhaps

looking to see if you would be a suitable mentor if you are interviewing for a more senior role, or if you show drive, determination, and passion if you are trying for a junior role.

Sometimes even this is not enough; perhaps the interviewer is in a bad mood, or having a bad day. The interviewer could even be mentally distracted from a broken build that happened just minutes before stepping into the interview. Unfortunately there is not much you can do in these circumstances. Always be polite and try not to upset the interviewer during any discussions.

Often late in the process, once the developers are happy that your ability matches their expectations, you will meet with the hiring manager. The role of this interview is to discuss your fit within the team, talk about some of the expectations and softer skills that are required for the role, and discuss your own career development within the company.

Like in all steps of the process, your manner and engagement are being examined here. The hiring manager wants to see that you are keen, smart, and able to hold a decent conversation. It is very rare that a developer sits alone and writes code all day with no interaction with anyone else. Talking to other developers, product managers, and many other people is necessary on a daily basis, and one of the main points of this conversation is to ensure that your communication with others is at an acceptable level.

Sometimes, if the interviewing team feels that you are not quite the right fit, they may terminate the interview early, informing you that they are not going to pursue the role any further. If this happens, it can be quite a shock, especially if you are not expecting it. Be assured that it is nothing personal, and is not a slight against your ability; it is purely that your skills do not match up with what they need. Try not to let this affect your confidence, and ask yourself if there is anything you need to brush up on before any interviews with a different employer.

MAKING THE DECISION

Being offered a role is not the end of the process. First, you must be absolutely sure that you want the role. If you have any doubts, even after an offer stage, approach the hiring manager or recruitment agent. They can often allay any fears you may have, or fill in the answers to any questions you didn't ask during the interview process.

Even after any further discussions, if you still feel that the role is not right for you, do not take it. You would be better to carry on looking rather than taking the job, being unmotivated, and then looking to move soon after, or worse, not making it through any probationary period.

If you are happy with the role, you will be offered a salary and perhaps other benefits as part of your employment package, such as stock options, a pension, and medical benefits. You may have some room to negotiate on the package at this point. Remember, the hiring manager and the rest of the team now want you to work for them. They have invested a lot of time and effort in examining you, and if they cannot provide you with an attractive offer now, they will have to start the process all over again. Do not push your luck, though: A developer salary and package will fall only within a certain range in the team's budget.

SUMMARY

If you are inexperienced with interviews, they can be quite daunting. The best way to remove any fear is simply with experience; the more interviews you have under your belt—successful or otherwise—the more you understand exactly what interviewers are looking for, and how to tackle the questions given. Although the questions in this book will provide a good insight into the types and styles of questions you might encounter during an interview, the actual execution of answering these questions effectively will only come with practice and experience.

Remember at all times that interviews are a two-way process. Although the interviewing team dictates how and when the interview process happens, you are interviewing the team, too. It is solely up to you to decide if you actually *want* to work for this team. Make sure you answer that question before you commit and move jobs.

People can find rejection hard from interviews, especially when they have had much success in academic life. Interviewing for a job is not the same as taking a university or college exam; getting the question right is not always sufficient. Every single aspect of you is being examined, from your personality to your skills, but also whether your skills match with what the hiring team needs. There is no shame in being rejected for a role. Getting the questions right is only half of the battle.

The biggest key to acing any interview is preparation. Make sure you understand the domain properly, and try to visualize what exactly the interviewers are going to want from the interview. Think about the questions they may ask, and make sure you have some questions of your own to ask them.

Chapter 3 visits the different interview types discussed in this chapter, and delves a little deeper into your conduct and how to produce good and relevant answers. Before that, however, Chapter 2 looks at the very first stage of the hiring process, before any interview takes place: how to write an appropriate resume and cover letter to get noticed by the recruiting team.