For preparation related questions: I was recently reading through one of the experiences of a person who got selected into top tech companies with a very high annual CTC and he hailed from a service-based company. His title is something that got stuck in my head "Life is a Marathon not a sprint". My answer below will also be on the same lines. It took me more than 6 months of dedicated serious preparation and 5 years of experience for it to happen. Preparation Guide:  
Coding:

Coding is a MUST HAVE skill. You should practice writing code as much as possible. If you are able to come up with an algorithm and not being able to code it. It's the same as not being able to solve questions at all. Almost all companies expect you to write BUG free, PRODUCTION ready code quickly.

DS/Algo:  
1. Focus on problem solving: build your concepts first and then solve as many questions as possible.  
2. Most companies do not ask a very difficult question. Focus on main topics first and then if you have time then go to very advance data structures like BTree, RB tree, AVL etc (Graph is not a advance DS). These DS has  
3. Read interview experiences of the company if you can find them. Solve questions yourself if possible.  
4. I used leetcode myself, interviewbit is just as good. Scaler has a really good course so that should set up a good base for you.  
5. If you are leetcoding - Even after submitting the questions, read the solution and check the discuss section for other approaches even when yours is more optimal. You may end up applying those tricks in further questions.  
6. If you have time, post your solution. This will enhance your understanding of the concept.  
7. Give as many contests as possible.

Desgin:

LLD:  
Read the basic desgin patterns and try to build atleast 2-3 examples on each and get it reviewed by a senior or a peer.  
LLD is what we learn everyday so learn things at your work, there is no better way of learning OO design then actually building a solution around it.  
Code reviews are very good way to get better at code design, get your code reviewed and do code reviews.

HLD:  
I used YT for system design.  
Read ground breaking white papers to understand the reasoning behind design choices.  
Read good tech blogs of top tech companies.  
Read about the design of real life systems. A lot of them are available on the public internet. Understand why is the developer making a specific design choice.  
I personally had a group where we solved two interview question every week. We would all discuss each other's solution and try to make other system fail and that gave a lot of confidence during real interview.I have also accumulated the HLD resources in a public github repo, its very crude right now but i will make it better, for all resources, in upcoming few weeks(hopefully) and you are welcome to make any changes too.  
Link to repo: <https://github.com/imkgarg/Awesome-Software-Engineering-Interview>Al last keep in mind that things take time to happen and there may be failures but if you persist, good things will definitely happen