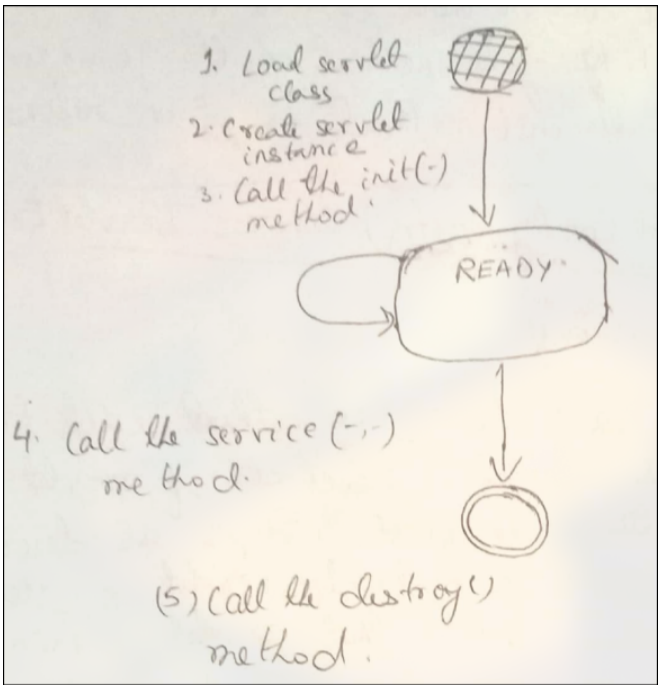
1. What is a Servlet in Java?

A Servlet is a Java class that handles HTTP requests and generates dynamic web content on a server.

2. Describe the lifecycle of a Servlet.



3. What is the difference between a Servlet and a JSP?

A Servlet is Java code handling requests, while JSP is an HTML-based file with embedded Java code, mainly for presentation.

4. How do you configure a Servlet in a web application?

You configure a servlet using the web.xml deployment descriptor or with @WebServlet annotation in the servlet class.

5. What is the web.xml file, and how is it used in configuring Servlets?

web.xml is the deployment descriptor for a Java web app. It configures servlets, mappings, filters, and other settings.

6. Explain the purpose of the HttpServlet class.

HttpServlet is an abstract class that simplifies handling HTTP requests by providing methods like doGet(), doPost(), doPut(), and doDelete(). Developers extend it to create web applications that process client requests and generate dynamic responses.

7. How do you handle GET and POST requests in a Servlet?

Override doGet(HttpServletRequest req, HttpServletResponse res) for GET requests and doPost(HttpServletRequest req, HttpServletResponse res) for POST requests in a HttpServlet class.

8. What is the ServletConfig interface?

ServletConfig provides configuration data to a servlet, like init parameters, and is accessible via getServletConfig().

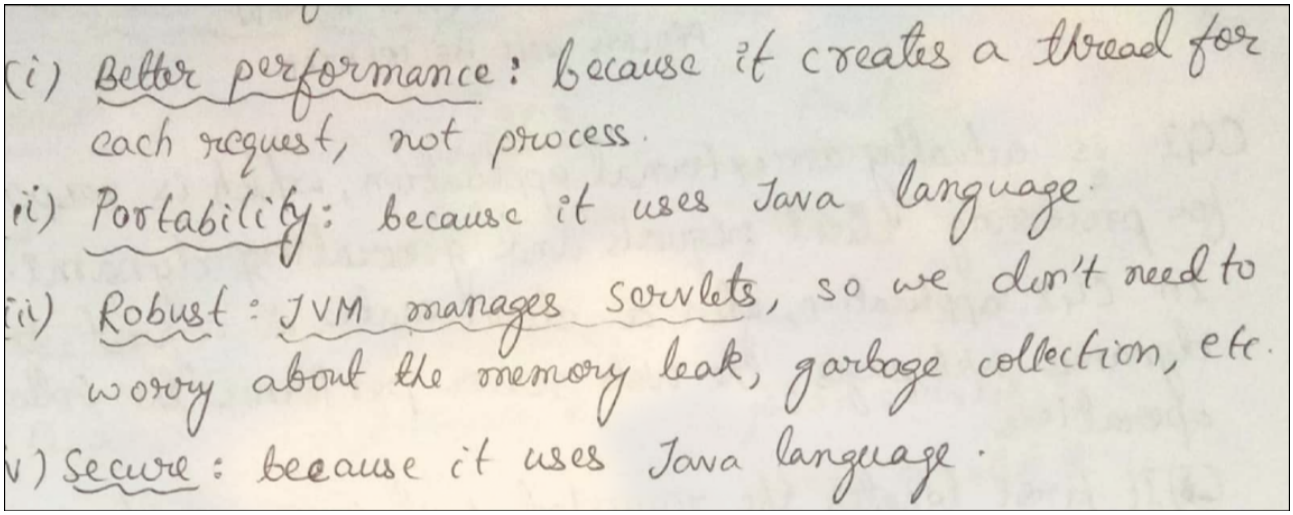
9. What is the ServletContext interface, and how is it different from ServletConfig?

ServletContext provides application-wide information, while ServletConfig is specific to a single servlet.

10. How do you initialize a Servlet?

A servlet is initialized by overriding the init() method or using web.xml or @WebServlet annotation for configuration

11. What are the advantages of using Servlets over CGI?



12. Explain the concept of session management in Servlets.

Session management tracks user data across requests using techniques like HTTP sessions (ores session data on the server and assigns a unique session ID to the client.), cookies, URL rewriting, and hidden fields.

13. What is the difference between doGet() and doPost() methods?

- doGet(): Used for retrieving data, appends parameters in the URL, and is idempotent.
- doPost(): Used for sending data, hides parameters in the request body, and is not idempotent.

14. How do you handle exceptions in Servlets?

Handle exceptions using try-catch, error-page(Define <error-page>) in web.xml, Use response.sendError() to send HTTP error codes,or @WebServlet with @WebInitParam.

15. Describe the process of file uploading using Servlets.

File uploading in servlets involves:

1. Using @MultipartConfig annotation: Enables multipart form data processing.
2. Form with enctype="multipart/form-data": Ensures file data is sent properly.
3. Parsing request with getPart() or Apache Commons FileUpload: Retrieves the uploaded file.
4. Saving the file using InputStream and FileOutputStream.

16. What are filters in Servlets, and how are they used?

Filters intercept requests and responses to modify or process data before reaching servlets. They are used for logging, authentication, compression, and request validation. Configured via web.xml or @WebFilter annotation.

17. How do you forward a request from one Servlet to another?

Use RequestDispatcher.forward(request, response) to forward a request without changing the URL.

18. What is the role of Request Dispatcher?

RequestDispatcher forwards or includes requests to another resource (Servlet, JSP, or HTML) within the same application.

19. Explain the difference between forwarding a request and redirecting a request in Servlets.

- Forward (RequestDispatcher.forward()): Happens on the server, keeps the same URL, and is faster.
- Redirect (response.sendRedirect()): Sends a new request to a different URL, visible to the client, and is slower.

20. How do you maintain security in a web application using Servlets?

Security in servlets is maintained by:

1. Authentication & Authorization: Using form-based, basic, or OAuth authentication.
2. HTTPS: Encrypts data to prevent interception.
3. Session Management: Prevents session hijacking with secure cookies and timeouts.
4. Input Validation: Prevents SQL injection and XSS attacks.
5. Role-based Access Control: Restricts access using web.xml or annotations.