**Spring Boot Interview Questions**

1) Explain the term ‘Spring Boot’.

2) Mention some advantages of Spring Boot

3) How to create a Spring Boot application using Spring Initializer?

4) How can you change JDK version in Spring Boot?

5) What is the process that you need to follow to run Spring Boot application on the

custom port?

6) Can you use Spring Boot with applications which are not using Spring?

7) What are the important features of Spring Boot?

8) What is a Spring Boot Actuator?

9) How can you connect Spring Boot to the database using JPA?

10) Where can you define properties in Spring Boot application?

11) What is @pathVariable?

12) How to handle exception in Spring Boot.

13) What are the Spring Boot key components?

14) Why Spring Boot over Spring?

15) What is the starter dependency of the Spring boot module?

16) How does Spring Boot works?

17) What does the @SpringBootApplication annotation do internally?

18) What is the purpose of using @ComponentScan in the class files?

19) How does a spring boot application get started?

20) What are starter dependencies?

21) What is Spring Initializer?

22) What is Spring Boot CLI and what are its benefits?

23) What are the most common Spring Boot CLI commands?

**Advanced Spring Boot Questions**

1. What are the Basic Annotations that Spring Boot Offers?

2. What is Spring Boot dependency management?

3. Can we create a non-web application in Spring Boot?

4. Is it possible to change the port of the embedded Tomcat server in Spring Boot?

5. What is the default port of tomcat in spring boot?

6. Can we override or replace the Embedded tomcat server in Spring Boot?

7. Can we disable the default web server in the Spring boot application?

8. How to disable a specific auto-configuration class?

9. Explain @RestController annotation in Sprint boot?

10. What is the difference between @RestController and @Controller in Spring Boot?

11. Describe the flow of HTTPS requests through the Spring Boot application?

12. What is the difference between RequestMapping and GetMapping?

13. What is the use of Profiles in spring boot?

14. What is Spring Actuator? What are its advantages?

15. How to enable Actuator in Spring boot application?

16. What are the actuator-provided endpoints used for monitoring the Spring boot application?

17. How to get the list of all the beans in your Spring boot application?

18. How to check the environment properties in your Spring boot application?

19. How to enable debugging log in the spring boot application?

20. Where do we define properties in the Spring Boot application?

21. What is dependency Injection?

22. What is an IOC container?

23. Explain the term ‘Spring Boot’.

- It is a Spring module that offers Rapid Application Development to Spring framework. Spring module is used to create an application based on Spring framework which requires to configure few Spring files.

24. Mention some advantages of Spring Boot Here are some major advantages of using spring-boot:

- Helps you to create a stand-alone application, which can be started using java.jar.

- It offers pinpointed‘started’ POMs to Maven configuration.

- Allows you to Embed Undertow, Tomcat, or Jetty directly.

- Helps you to configure spring whenever possible automatically.

25. How to create a Spring Boot application using Spring Initializer?

- It is a web tool provided by Spring on its official website. However, you can also create Spring Boot project by entering project details.

26. How can you change JDK version in Spring Boot?

- To change the JDK version in Spring Boot, you can overwrite it by adding a java. version property tag as given: 1.8

27. What is the process that you need to follow to run Spring Boot application on the custom port?

- In order to run a Spring Boot application, you require to put server.port properties in application.properties. For example, server.port=8050

28. Can you use Spring Boot with applications which are not using Spring?

- No, it is not possible as Spring Boot is limited to Spring application only.

29. What are the important features of Spring Boot?

- Important features of Spring Boot are:

* Web Development
* Spring Application
* Application occasions and listeners
* Admin highlights
* YAML Support
* Type-safe Configuration
* Externalized Configuration
* Properties Files
* Logging and Security

30. What is a Spring Boot Actuator?

- Spring Boot Actuator allows you to monitor and manage your application when you want to push it for the production. It helps you to control your application by using HTTP endpoints.

31. How can you connect Spring Boot to the database using JPA?

- Spring Boot supports spring-boot-data-JPA start, which helps you to connect spring application with a relational database.

32. Where can you define properties in Spring Boot application?

- You can define properties of Spring Boot into a file called application.properties. It helps you to create this file manually, or you can use Spring Initializer to create this file.

33. What is @pathVariable?

- @PathVariable annotation helps you to extract information from the URI directly.

34. How to handle exception in Spring Boot.

- Spring Boot provides a very useful way to handle exceptions using @ControllerAdvice annotation.

35. Problem – 01.

You have to complete the code by writing down the handlers for exceptions thrown by the code. The exceptions the code may throw along with the handler message are listed below:

1. Division by zero: Print "Invalid division".
2. String parsed to a numeric variable : Print "Format mismatch".
3. Accessing an invalid index in string : Print "Index is invalid".
4. Accessing an invalid index in array : Print "Array index is invalid".

Solution:-

import java.util.Scanner;

import java.lang.Exception;

class TestClass

{

static void solve(int arr[]) throws Exception

{

int ans=0;

for(int i=0;i<10;i++)

for(int j=i+1;j<10;j++)

ans+=arr[i]/arr[j];

throw new MyException(ans);

}

public static void main(String args[])

{

try

{

Scanner in=new Scanner(System.in);

int n=in.nextInt();

int []arr= new int[n];

in.nextLine();

for(int i=0;i<10;i++)

arr[i]=Integer.parseInt(in.nextLine());

String s=in.nextLine();

System.out.println(s.charAt(10));

solve(arr);

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is invalid");

}

catch(NumberFormatException e)

{

System.out.println("Format mismatch");

}

catch(StringIndexOutOfBoundsException e)

{

System.out.println("Index is invalid");

}

catch(ArithmeticException e)

{

System.out.println("Invalid division");

}

catch(MyException e)

{

System.out.println(e);

}

catch(Exception e)

{

System.out.println("Exception encountered");

}

finally

{

System.out.println("Exception Handling Completed");

}

}

}

class MyException extends Exception

{

private int detail;

MyException(int a)

{

detail=a;

}

public String toString()

{

return "MyException["+detail+"]";

}

}

36. How to disable a specific auto-configuration class?

-You can use exclude attribute of @EnableAutoConfiguration if you want auto-configuration not to apply to any specific class.

//use of exclude

@EnableAutoConfiguration(exclude={className})

37. How to enable debugging log in the spring boot application?

Debugging logs can be enabled in three ways -

- We can start the application with --debug switch.

- We can set the logging.level.root=debug property in application.property file.

- We can set the logging level of the root logger to debug in the supplied logging configuration file.

38. How we can override or replace the Embedded tomcat server in Spring Boot?

Yes, we can replace the Embedded Tomcat server with any server by using the Starter dependency in the pom.xml file. Like you can use spring-boot-starter-jetty as a dependency for using a jetty server in your project

39. How we can disable the default web server in the Spring boot application?

Yes, we can use application.properties to configure the web application type i.e spring.main.web-application-type=none.

40. How to disable a specific auto-configuration class?

- You can use exclude attribute of @EnableAutoConfiguration if you want auto-configuration not to apply to any specific class.

//use of exclude

@EnableAutoConfiguration(exclude={className})