Preguntas y Respuestas Java POO, Spring Boot y Spring Cloud

Java POO (8 preguntas)

1. ¿Qué es POO y sus pilares?
- POO es un paradigma basado en objetos que combina datos y métodos.
- Pilares: Encapsulación, Herencia, Polimorfismo, Abstracción.

2. Clase vs Objeto
- Clase: plantilla o molde.
- Objeto: instancia de una clase.
Ejemplo: class Persona {} Persona p = new Persona();

3. Herencia
- Permite que una clase herede atributos y métodos de otra.
Ejemplo: class Animal {} class Perro extends Animal {}

4. Polimorfismo
- Un objeto puede comportarse de diferentes formas según el contexto.
Ejemplo: Animal a = new Perro(); a.hacerSonido();

5. Overloading vs Overriding
- Overloading: mismos métodos, diferente firma.
- Overriding: sobrescribir método de la superclase.
Ejemplo: class Ejemplo { void mostrar(int x) {} void mostrar(String s) {} } class Sub extends Ejemplo { @Override void mostrar(int x) {} }

6. Encapsulación
- Ocultar los datos internos de una clase y exponerlos mediante métodos.
Ejemplo: private String nombre; public String getNombre() { return nombre; } public void setNombre(String nombre) { this.nombre = nombre; }

7. Abstract class vs Interface
- Abstract: puede tener métodos con y sin implementación.
- Interface: solo métodos abstractos (Java 8+ puede tener default).

8. Constructores
- Constructor por defecto: no recibe parámetros.
- Constructor parametrizado: recibe valores para inicializar atributos.
Ejemplo: class Persona { String nombre; Persona() {} Persona(String nombre) { this.nombre = nombre; } }

Spring Boot (7 preguntas)

9. Qué es Spring Boot
- Framework que simplifica Spring Framework con configuración automática y stand-alone apps.

10. @RestController vs @Controller
- @RestController: retorna datos (JSON/XML).
- @Controller: retorna vistas (HTML).

11. @Autowired
- Inyección automática de dependencias.
Ejemplo: @Service class Servicio {} @RestController class Controlador { @Autowired Servicio servicio; }

12. application.properties / application.yml
- Configuración central de la app.
Ejemplo: server.port=8081 spring.datasource.url=jdbc:mysql://localhost/db

13. Spring Data JPA
- Facilita acceso a bases de datos mediante repositorios.
Ejemplo: public interface UsuarioRepo extends JpaRepository<Usuario, Long> {}

14. Bean
- Objeto gestionado por Spring.
Ejemplo: @Bean public Servicio servicio() { return new Servicio(); }

15. Ciclo de vida de un Bean
- Instanciación -> Inyección de dependencias -> Inicialización -> Destrucción.

Spring Cloud (5 preguntas)

16. Qué es Spring Cloud
- Conjunto de herramientas para microservicios: descubrimiento, configuración, balanceo, gateway, etc.

17. Eureka
- Servicio de descubrimiento: los microservicios se registran y pueden encontrarse entre sí.

18. Spring Cloud Config
- Centraliza la configuración de todos los microservicios.
Ejemplo:
```
server:
  port: 8888
spring:
  cloud:
```

```
  config:
    server:
      git:
        uri: https://github.com/mi-config-repo
```

19. Ribbon
- Cliente de balanceo de carga entre microservicios, por ejemplo: Round-robin.

20. Spring Cloud Gateway
- Gateway basado en Spring: enruta peticiones y aplica filtros.
- Diferencia con un API Gateway tradicional: integración nativa con microservicios Spring y facilidad de configuración en código.