**20. What is Spring AOP and proxy pattern?**

Aspect-oriented programming (AOP) is a design pattern that helps us manage aspects like logging, security, and transaction management in applications. Spring AOP provides an implementation of AOP using dynamic proxies.

The proxy pattern is a software design pattern that creates a proxy object that intercepts incoming requests and controls access to another object before reaching the bean. In Spring AOP, dynamic proxies are used to implement aspects.

**21. Explain key components of AOP.**

* **Aspect:** A building block bundles together cross-cutting concerns. It has two main parts i.e. advice and pointcut.
  + **Advice:** The code that is executed before, after, or around a method invocation.
  + **Pointcut:** Condition triggering the tasks(advice).
* **Join point:** A specific point in the program execution where an aspect can be applied. Common join points are
  + method calls
  + field access
  + object creation
* **Weaving:**Spring supports *weaving* at compile, load, and runtime for integrating *aspects* in the application at *join points*.

**23. Differentiate between Spring AOP and AspectJ AOP?**

| **Feature** | **Spring AOP** | **AspectJ AOP** |
| --- | --- | --- |
| **Programming model** | Annotation or XML configuration supported | Dedicated AspectJ compiler |
| **Weaving** | Dynamic proxy weaving at runtime | runtime weaving supported |
| **Supported features** | Aspect composition, pointcuts, advice, etc | control flow join and aspect inheritance |

**24. What are the advantages of AOP and its implementation?**

AOP helps to maintain, modify, and understand code easily,

* **Modularization:** separation of concerns like logging, security, etc from core business logic, to increase maintainability.
* **Reusability:** Bundles concerns in reusable aspects, improving code reusability.
* **Interception:** Allows interception and modification of method calls, enabling features like logging, security, and caching.