Spring AOP (Aspect Oriented Programming):

1. It breaks the program logic into distinct parts called as **Concerns.**
2. Repeatedly used functions over the application are called as **Cross Cutting Concerns**
3. Examples for aspects – **Logging, Auditing, Declarative transaction, security and caching.**
4. **Class in OOP = Aspect in AOP**
5. **AOP helps to decouple cross cutting concerns from the objects they affect.**

**Aspect:** Set of APIs providing the cross cutting requirements. An application can have any number of aspects. Types of aspects are -

1. **After** = run the advice after method execution.
2. **Before** = run the advice before method execution.
3. **After-returning** = run the advice after method execution, only if it executes successfully.
4. **After-throwing** = run the advice after method execution exits by throwing the exception
5. **Around** = run the advice before and after the advised method is invoked.

**Join Point:** The actual place in application where AOP is used.

**Advice:** The actual action to be taken before or after the method execution. This piece of code will be executed by Spring AOP framework.

**Point cuts**: Set of one or more Join Points where an advice will be executed.