@Bean

@Bean methods are also discovered on base classes of a given component or configuration class, as well as on Java 8 default methods declared in interfaces implemented by the component or configuration class. This allows for a lot of flexibility in composing complex configuration arrangements, with even multiple inheritance being possible through Java 8 default methods as of Spring 4.2.

This statement highlights the flexibility provided by Spring in defining multiple `@Bean` methods for the same bean within a single class. Here's a breakdown of the key points:

- **Multiple @Bean methods**: In Spring, it's possible to define multiple `@Bean` methods within a single class. Each of these methods can produce a bean of the same type or different types.

- **Variants based on dependencies**: These multiple `@Bean` methods can serve as alternative factory methods, each potentially creating the bean with different configurations or dependencies.

- **Selection algorithm**: At runtime, Spring selects the appropriate `@Bean` method to use based on the availability of dependencies. The method with the largest number of satisfiable dependencies is chosen. This selection process is similar to how Spring determines the most suitable constructor or factory method when injecting dependencies using `@Autowired`.

- **Constructor selection analogy**: The process of selecting the appropriate `@Bean` method is analogous to how Spring chooses the "greediest" constructor or factory method during bean construction. Spring analyzes the dependencies required by each `@Bean` method and selects the one with the most satisfiable dependencies.

In summary, by defining multiple `@Bean` methods within a single class, developers can provide different variations of beans based on runtime conditions or available dependencies, and Spring's container will select the most appropriate method to use. This approach enhances the flexibility and configurability of Spring applications.