Lazily or Eagerly instantiated?

Lazy and Eager Instance Creation vs Scope Type:

- Singleton Beans are eagerly instantiated by default
- Prototype Beans are lazily instantiated by default (instance is created when bean is requested)
  - …however, if Singleton Bean has dependency on Prototype Bean, then Prototype Bean Instance will be created eagerly to satisfy dependencies for Singleton Bean

Altering Behavior:

- You can change default behavior for all beans by @ComponentScan annotation

```
@ComponentScan(lazyInit = true)
```

  - Setting lazyInit to true, will make all beans lazy, even Singleton Beans
  - Setting lazyInit to false (default), will create Singleton Beans Eagerly and Prototype Beans Lazily

- You can also change default behavior by using @Lazy annotation:
  - @Lazy annotation takes one parameter - Whether lazy initialization should occur
  - By default @Lazy is used to mark bean as lazily instantiated
  - You can use @Lazy(false) to force Eager Instantiation – use case for @ComponentScan(lazyInit = true) when some beans always needs to be instantiated eagerly
- @Lazy can be applied to:
  - Classed annotated with @Component – makes bean Lazy or as specified by @Lazy parameter
  - Classes annotated with @Configuration annotation – make all beans provided by configuration lazy or as specified by @Lazy parameter
  - Method annotated with @Bean annotation – makes bean created by method Lazy or as specified by @Lazy parameter