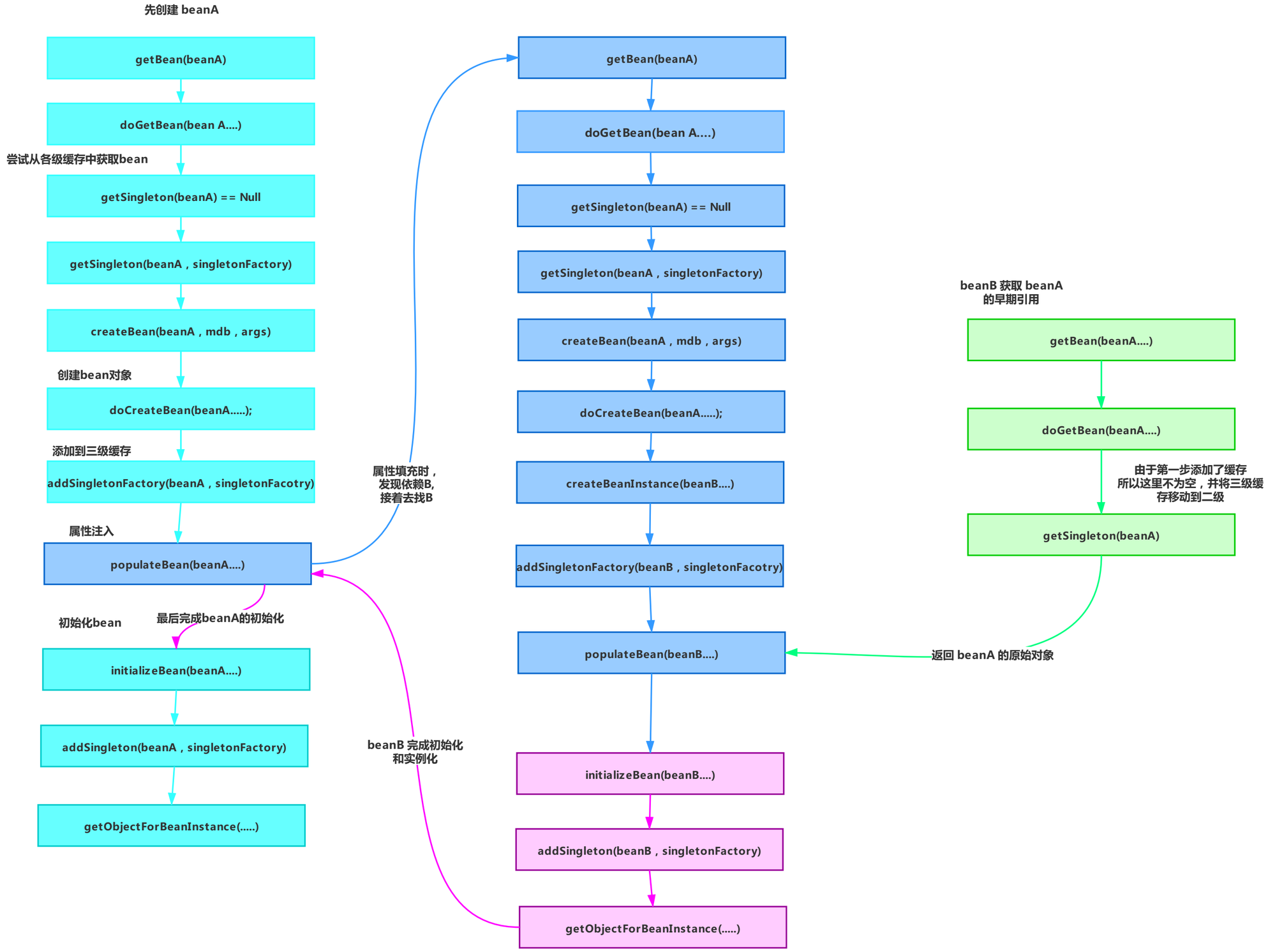

1、调用 doGetBean() 方法，想要获取 beanA ，于是调用 getSingleton() 方法从缓存中查找 beanA
2、在 getSingleton() 方法中，从一级缓存中查找，没有，返回 null
3、doGetBean() 方法中获取到 beanA 为 null ，于是走对应的处理逻辑，调用 getSingleton() 的重载方法(参数为 ObjectFactory 的)
4、在 getSingleton()方法中，先将 beanA_name 添加到一个集合中，用于标记该 bean 正在创建中，然后回调匿名内部类的 createBean 方法
5、进入 AbstractAutowireCapableBeanFactory#doCreateBean，先反射调用构造器创建出 beanA 的实例，然后判断，是否为单例，是否允许提前暴露引用（对于单例一般为true）、是否正在创建中（即是否是在第四步的集合中）判断为 true 则将 beanA 添加到【三级缓存】中
6、对 beanA 进行属性填充，此时检测到 beanA 依赖于 beanB ，于是查找 beanB
7、调用 doGetBean() 方法，和上面 beanA 的过程一样，到缓存中查询 beanB ，没有则创建，然后给 beanB 填充属性
8、此时 beanB 依赖于 beanA ，调用 getSingleton() 获取 beanA ,依次从一级、二级、三级缓存中找、此时从三级缓存中获取到 beanA 的创建工厂，通过创建工厂获取到 singletonObject ，此时这个 singletonObject 指向的就是上面在 doCreateBean() 方法中实例化的 beanA
9、这样 beanB 就获取到了 beanA 的依赖，于是 beanB 顺利完成初始化，并将 beanA 从三级缓存移动到二级缓存中
10、随后 beanA 继续他的属性填充工作，此时也获取到了 beanB ，beanA 也随之完成了创建，回到 getSingleton() 方法中继续向下执行，将 beanA 从二级缓存移动到一级缓存中·