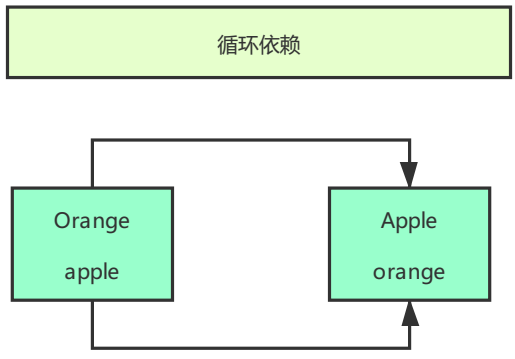

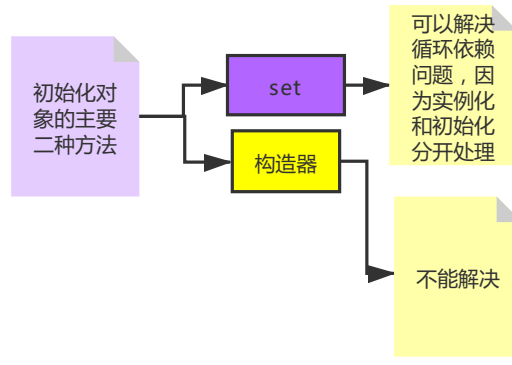

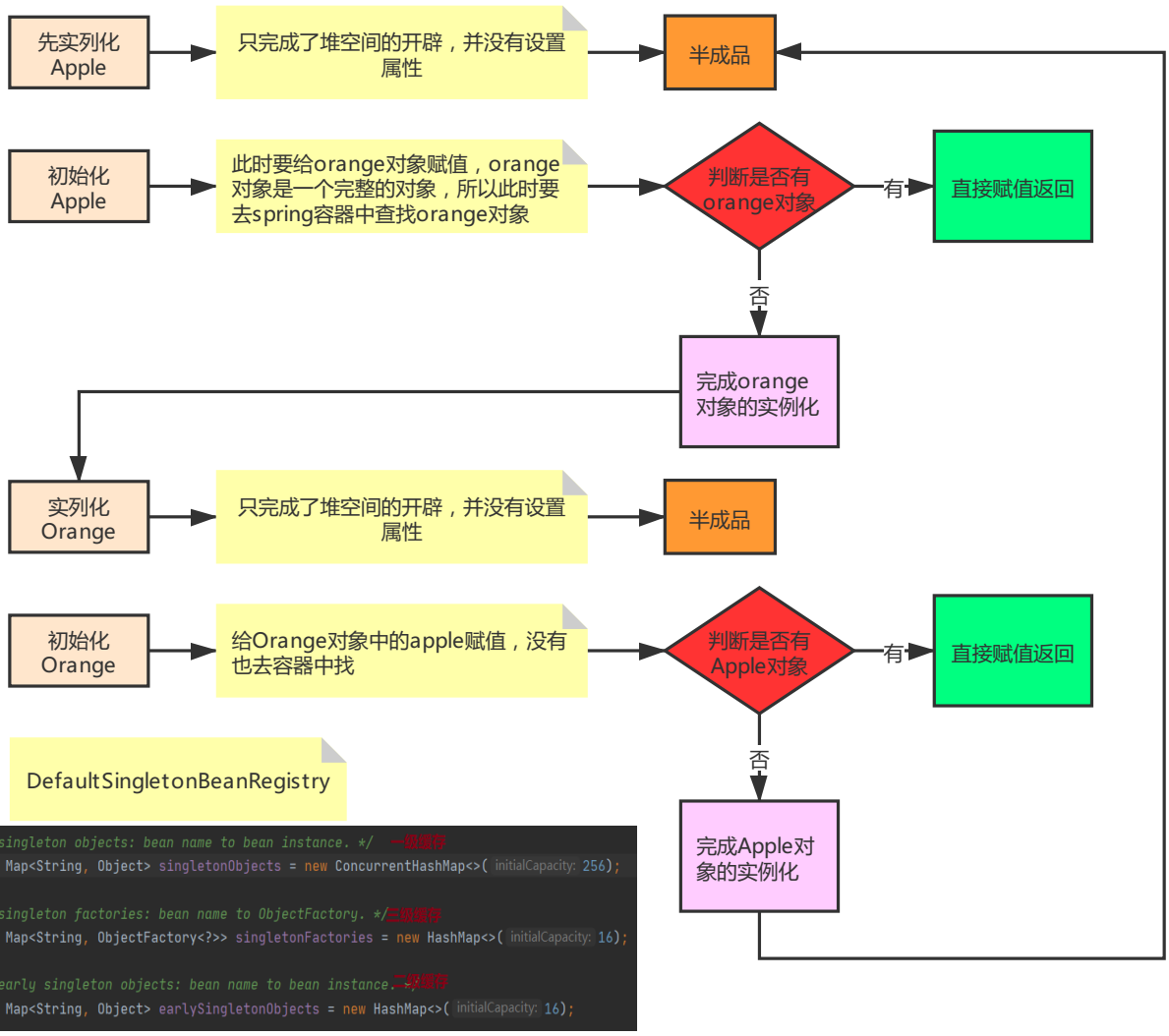

DefaultSingletonBeanRegistry

```
/** Cache of singleton objects: bean name to bean instance. */  一级缓存
private final Map<String, Object> singletonObjects = new ConcurrentHashMap<>( initialCapacity: 256);

/** Cache of singleton factories: bean name to ObjectFactory. */三级缓存
private final Map<String, ObjectFactory<?>> singletonFactories = new HashMap<>( initialCapacity: 16);

/** Cache of early singleton objects: bean name to bean instance.二级缓存
private final Map<String, Object> earlySingletonObjects = new HashMap<>( initialCapacity: 16);
```

ObjectFactory是一个函数式接口，可以传入一个lamda表达式或者一个匿名内部类，通过getObject()方法来执行具体逻辑

核心方法：
AbstractApplicationContext.refresh()
finishBeanFactoryInitialization(beanFactory);
beanFactory.preInstantiateSingletons();
DefaultListableBeanFactory.getBean(beanName)
doGetBean()
getSingleton(beanName)
createBean()
doCreateBean()
createBeanInstance(beanName, mbd, args)
populateBean(beanName, mbd, instanceWrapper)
applyPropertyValues(beanName, mbd, bw, pvs)
resolveValueIfNecessary(pv, originalValue)

重要的lambda表达式
() ->
getEarlyBeanReference(
beanName, mbd, bean)