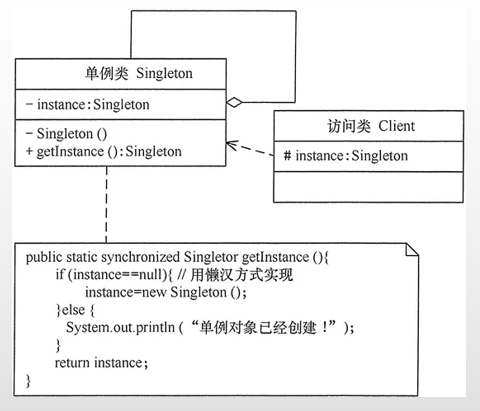
**单例模式：**

ULM：



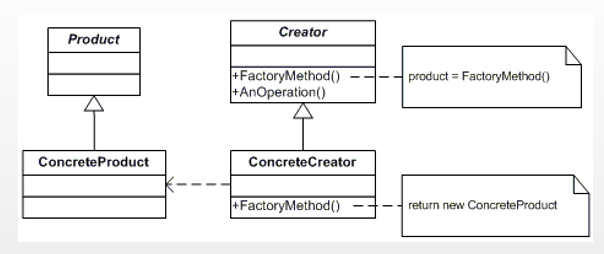
应用场合:

（1）资源共享的情况下: 单例模式可以保证全局只有一个实例，避免了对资源的多重占用。

（2）控制资源访问: 单例模式可以用于控制对资源的访问，例如，一个只读数据库连接池的设计。

（3）配置管理器: 单例模式可以用于创建一个全局的配置管理器，确保系统中的各个模块使用相同的配置。

**工厂方法模式：**ULM：



应用场合：

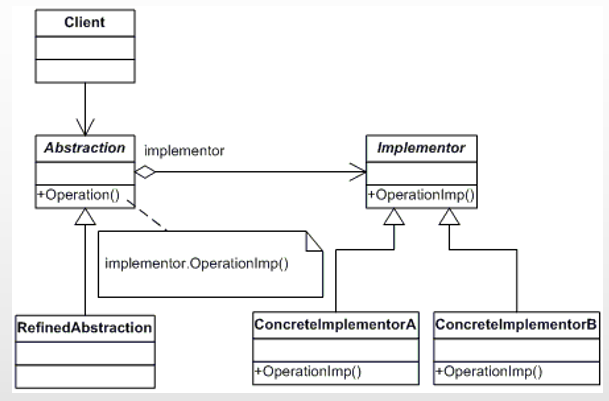
（1）对象的创建延迟到子类: 工厂方法模式允许一个类的实例化延迟到其子类，从而在子类中选择创建哪种类型的对象。

（2）避免客户端与具体类耦合: 客户端代码通过工厂方法来创建对象，而不是直接实例化具体类，从而减少与具体类的耦合。

（3）应对变化: 工厂方法模式使得系统易于扩展，添加新的产品类无需修改现有的代码，只需添加新的具体工厂和产品类即可。

**桥接模式：**

ULM：



应用场合：

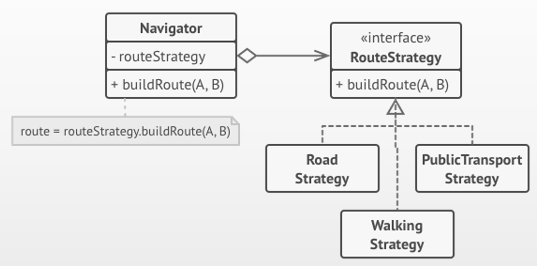
（1）抽象和实现分离: 桥接模式主要用于抽象和实现两个维度的变化，使得它们可以独立演化。例如，一个抽象类可以有多个不同的实现类。

（2）减少子类数量: 桥接模式可以用于减少类的数量，通过将抽象部分和实现部分分离，可以独立地变化，减少了子类的数量。

（3）多维度的变化: 当一个类有多个独立变化的维度时，可以使用桥接模式。例如，操作系统和文件系统之间存在两个独立变化的维度。

**策略模式：**

ULM：



应用场合:

（1）算法的自由切换: 策略模式允许客户端在运行时选择算法，实现了算法的自由切换。

（2）避免使用条件语句: 当有多个相关的类，并且它们的行为只有在运行时才能确定时，可以使用策略模式来避免使用复杂的条件语句。

（3）算法实现的隔离: 将不同的算法实现分离开来，使得每个算法的实现都可以独立变化而不影响其他部分。