1.定义

桥接模式将抽象部分与它的实现部分分离，使他们都可以独立的变化。

1. 类的功能层次结构 和 实现层次结构

功能层次结构:

指的是类与类之间的继承关系

特点：具备父类的基本功能、子类在父类基础上添加新功能。

实现层次结构：

指父类是抽象类，负责声明一套接口，然后子类去继承父类并实现这套接口，简单讲就是父类定义接口，子类实现接口。

特点：父类声明抽象方法定义相关接口、子类通过具体方法来实现接口。

1. 组成角色

\*抽象化:该角色位于属于”类的功能层次结构”的最上层，用于，定义抽象接口，一般是抽象类而不是抽象接口。内部往往包含一个实现类接口实例（implementor）,使用委托方式进行内部调用；

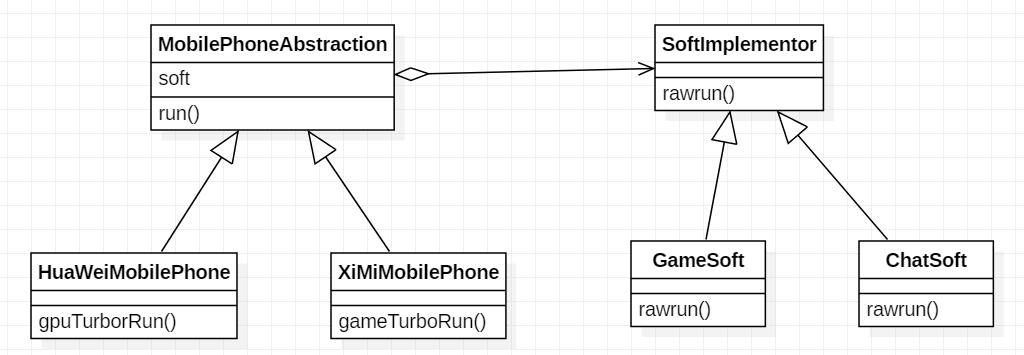
\*改善后的抽象化，也叫补充抽象类（RefinedAbstraction）:该角色用于补充Abstraction功能而存在，通常情况下不再是抽象类而是具体的实现类，在内部可以直接调用Implementor中的业务方法；

\*实现者（Implementor）：该角色位于“类得实现层次结构”的最上层，定义了用于实现Abstraction角色的接口,这里的接口并非要和Abstraction中定义的完全一致，Implementor支队这些接口进行声明，具体实现还是要交给子类。通过委托，在Abstraction中，不仅可以调用自己方法，还可以调用到Implementor中定义的方法；

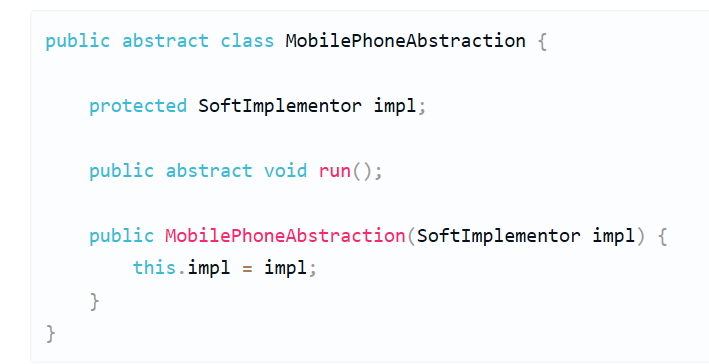
\*具体实现者（Concretelmplementor）:该角色用于实现Implementor角色中定义的接口，不同的实现类提供不同的业务处理方法，程序运行时，Concretelmplementor将替换Abstraction中的Implementor，提供给抽象类具体的业务操作方法。

1. 使用示例

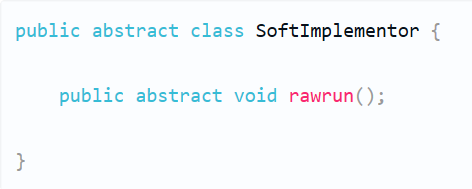
比如不同的手机品牌，每种品牌的手机上面又包含各种各样的软件，我们如何表示”在手机上运行软件”这一活动？手机的品牌和软件是两个维度上的东西，可以类比为上面的“功能层及”和“实现层级”，桥接模式的作用就是将这些不同的实现独立开来，从而应对不断地变化。



这里我们定义抽象类MobilePhoneAbstraction，表示手机抽象类，该类自带run方法来运行各种手机软件。



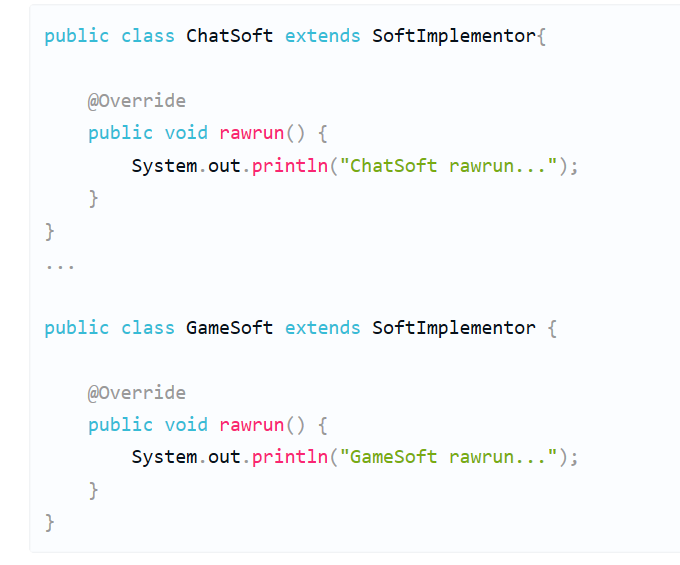
MobliePhoneAbstraction 为我们定义的类的功能层次结构的顶层，内部包含对SoftImplementor的引用，SoftImplementor就是我们定义的手机软件的抽象类。



这里我们声明rawrun方法供子类实现，同时又委托给MobilePhoneAbstraction供其调用，下面是手机品牌的实现类:



这里声明rawrun接口，实现的子类有两个，分别是游戏软件--GameSoft,和聊天软件--ChatSoft，代码如下：





由于小米手机和华为手机都属于MobilePhoneAbstraction的类的实例，因此我们可以调用他们的run方法，而小米手机、华为手机又都属于小米、华为手机的实例，所以我们还能调用其gputurborRun方法，比如后期我们软件的run实现有所变动，那么只需要修改main方法和Softlmplementor的实现类即可，七大代码就可以保持不动了。

通过MobilePhoneAbstraction中的impl桥接，就实现了抽象与行为实现的分离，这种就是桥接模式的存在意义。

1. 优缺点

\*抽象与实现相分离：抽象与实现相分离，从而让抽象与实现分别独立开来，分别定义接口，有助于系统分层及产生更好的结构化系统；

\*更好的扩展性：系统扩展时，因为抽象与实现已经分别独立，可以进行分别括展不会相互影响，从而大大提高系统扩展性