　　桥接（Bridge）是用于把抽象化与实现化解耦，使得二者可以独立变化。这种类型的设计模式属于结构型模式，它通过提供抽象化和实现化之间的桥接结构，来实现二者的解耦。

　　这种模式涉及到一个作为桥接的接口，使得实体类的功能独立于接口实现类。这两种类型的类可被结构化改变而互不影响。

　　个人理解：桥接是一个接口，它与一方应该是绑定的，也就是解耦的双方中的一方必然是继承这个接口的，这一方就是实现方，而另一方正是要与这一方解耦的抽象方，如果不采用桥接模式，一般我们的处理方式是直接使用继承来实现，这样双方之间处于强链接，类之间关联性极强，如要进行扩展，必然导致类结构急剧膨胀。采用桥接模式，正是为了避免这一情况的发生，将一方与桥绑定，即实现桥接口，另一方在抽象类中调用桥接口（指向的实现类），这样桥方可以通过实现桥接口进行单方面扩展，而另一方可以继承抽象类而单方面扩展，而之间的调用就从桥接口来作为突破口，不会受到双方扩展的任何影响。

　　详见demo04的实例,能真正体现这种模式的好处，现在我们要添加来源地和目的地，只要继续继承AreaA和实现Qiao即可，之前我所说的绑定，正是此处将桥与目的地绑定在一起，使用一个接口完成。

　　其实要完成桥接模式，注意点并不多，重在理解模式的使用场景。

　　注意点：

　　　　1、定义一个桥接口，使其与一方绑定，这一方的扩展全部使用实现桥接口的方式。

　　　　2、定义一个抽象类，来表示另一方，在这个抽象类内部要引入桥接口，而这一方的扩展全部使用继承该抽象类的方式。

其实我们可以发现桥接模式应对的场景有方向性的，桥绑定的一方都是被调用者，属于被动方，抽象方属于主动方。

　　下面来理解一下开头的概念：

　　桥接（Bridge）是用于把抽象化与实现化解耦，使得二者可以独立变化。这种类型的设计模式属于结构型模式，它通过提供抽象化和实现化之间的桥接结构，来实现二者的解耦。

　　这种模式涉及到一个作为桥接的接口，使得实体类的功能独立于接口实现类。这两种类型的类可被结构化改变而互不影响。

　　理解：此处抽象化与实现化分别指代实例中的双方，而且实现化对应目的地方（通过实现桥接口进行扩展），抽象方对应来源地方（通过继承抽象类来进行扩展），如果我们不使用桥接模式，我们会怎么想实现这个实例呢？很简单，我们分别定义来源地A1、A2、A3类和目的地B1、B2、B3，然后具体的实现就是，A1到B1一个类，A1到B2一个类，等，如果我们要扩展了A和B ,要直接增加An类和Bn类，如此编写不说类内部重复性代码多，而且还会导致类结构的急剧膨胀，最重要的是，在通过继承实现路径的时候，会造成双方耦合性增大，而这又进一步加剧了扩展的复杂性。使用桥结构模式可以很好地规避这些问题：重在解耦。