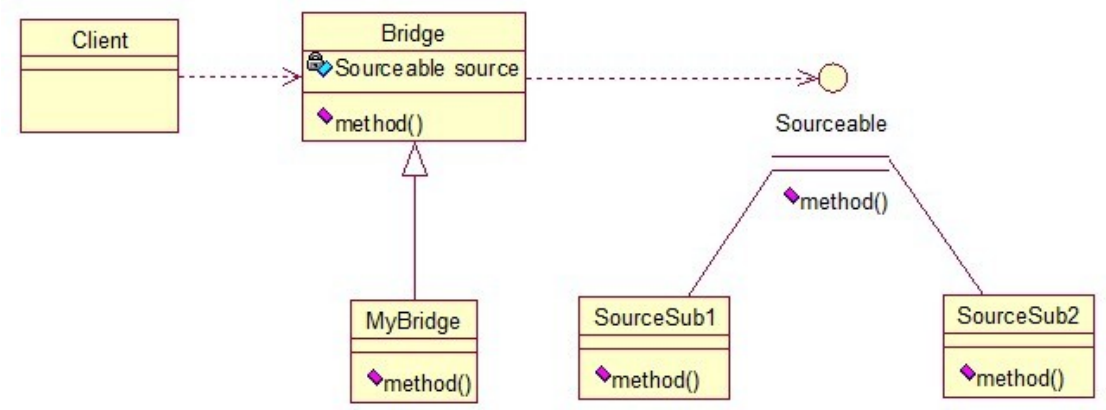
桥接模式就是把事物和其具体实现分开，使他们可以各自独立的变化。桥接的用意是：**将抽象化与实现化解耦，使得二者可以独立变化**，像我们常用的JDBC桥DriverManager一样，JDBC进行连接数据库的时候，在各个数据库之间进行切换，基本不需要动太多的代码，甚至丝毫不用动，原因就是JDBC提供统一接口，每个数据库提供各自的实现，用一个叫做数据库驱动的程序来桥接就行了。我们来看看关系图：



实现代码：

先定义接口：

[java] view plaincopy

```
public interface Sourceable {
    public void method();
}
```

分别定义两个实现类：

[java] view plaincopy

```
public class SourceSub1 implements Sourceable {

    @Override
    public void method() {
        System.out.println("this is the first sub!");
    }
}
```

[java] view plaincopy

```
public class SourceSub2 implements Sourceable {
```

```
    @Override
    public void method() {
        System.out.println("this is the second sub!");
    }
}
```

定义一个桥，持有Sourceable的一个实例：

[java] view plaincopy

```
public abstract class Bridge {
    private Sourceable source;

    public void method(){
        source.method();
    }

    public Sourceable getSource() {
        return source;
    }

    public void setSource(Sourceable source) {
        this.source = source;
    }
}
```

[java] view plaincopy

```
public class MyBridge extends Bridge {
    public void method(){
        getSource().method();
    }
}
```

测试类：

[java] view plaincopy

```
public class BridgeTest {

    public static void main(String[] args) {

        Bridge bridge = new MyBridge();

        /*调用第一个对象*/
        Sourceable source1 = new SourceSub1();
        bridge.setSource(source1);
        bridge.method();

        /*调用第二个对象*/
        Sourceable source2 = new SourceSub2();
        bridge.setSource(source2);
        bridge.method();
    }
}
```

output：

this is the first sub!

this is the second sub!

这样，就通过对Bridge类的调用，实现了对接口Sourceable的实现类SourceSub1和SourceSub2的调用。接下来我再画个图，大家就应该明白了，因为这个图是我们JDBC连接的原理，有数据库学习基础的，一结合就都懂了。