接口的作用

接口的作用简单一点就是:接口是用来标记类的，不同的类属于不同的接口（通过向上转型），管理接口比管理各种各样的类方便多了，接口体现了抽象的观点,什么是抽象？抽象就是"抽去像的部分"。

使用接口解决问题

问题:现在我们要写个连接数据库的类给用户使用，有两个函数:一个返回 Connection 对象，另一个是关闭数据库,close(),一般的解决方法是:给每个数据库写一个类，再根据用户使用的数据库决定使用具体的类。

好的，我们看看这样有什么不好之处:

(1).首先每个类都要有重复的代码，造成代码的膨胀；

(2).其次最重要的是我们并不知道用户使用什么数据库，可能是 Oracle,可能是 mysql,也可能是 sqlserver 等，这个问题很难解决。

解决方案:

首先我们定义接口:

```
public interface DataBase
    {
    java.sql.Connection openDB(String url,String
user,String password);
    void close();
    }
```

我们定义了两个方法，openDB 返回 Connection 对象,close()关闭数据库;

具体的实现在实现 DataBase 接口的类中;

下面看看实现:

```
import java.sql.*;
    public class Mysql implements DataBase
    {
    private String url=”jdbc:mysql:localhost:3306/test”;
    private String user=”root”;
    private String password=””;
```

```
private Connection conn;
public Connection openDB(url,user,password)
{
    //连接数据库的代码
}

public void close()
{
    //关闭数据库
}
}
```

类 mysql 实现了 DataBase 接口，下面还有实现了 DataBase 接口的 oraclesql 等类；

这些类都归于 DataBase 接口了，如何在应用程序中使用呢?

我们要定义 DataBase 对象 myDB,通过 myDB 来操纵数据库，可以不要分清是哪个类了。

另外的问题:Java 中不许我们实例化接口，如 DataBase myDB=new DataBase();

我们只能 myDB=new Mysql()或者 myDB=new Oracle()。这样我们还必须指定实例化哪个对象，好像前面的努力都白费了啊!!那怎么办呢，我们需要一个工厂：

```
public class DBFactory
    {
    public static DataBase Connection getConn()
    {
        Return(new Mysql());
    }
    }
```

实例化的代码变成：myDB=DBFactory.getConn()；

整个过程中接口不负责任何具体操作，其他的程序要连接数据库的话，只需要构造一个 DB 对象就 OK，而不管工厂类如何变化。这就是接口的意义----抽象。