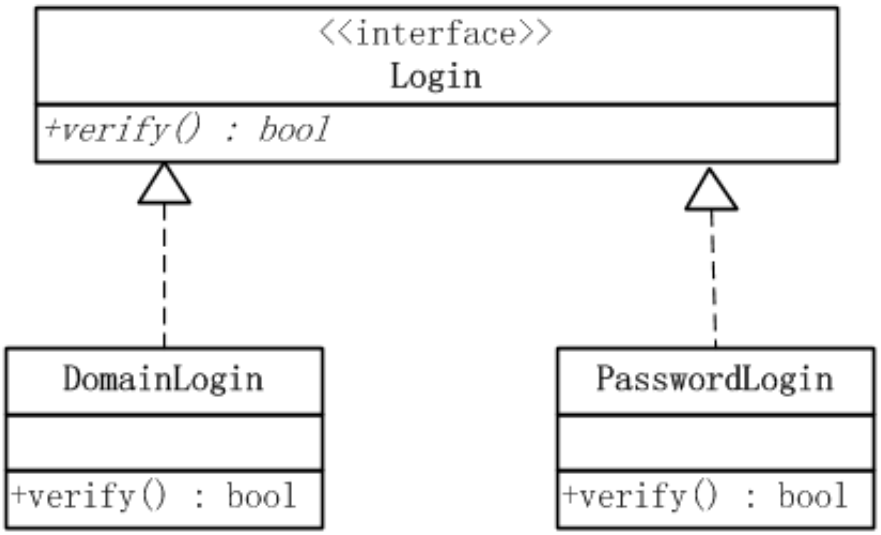
简单工厂模式

> *整理自：《java与模式》之简单工厂模式*

在阎宏博士的《JAVA与模式》一书中开头是这样描述简单工厂模式的：**简单工厂模式是类的创建模式，又叫做静态工厂方法（Static Factory Method）模式。简单工厂模式是由一个工厂对象决定创建出哪一种产品类的实例。**

那么 **简单工厂模式** 是在什么场景下使用呢，下面就以本人的理解举例说明:

就拿登录功能来说，假如应用系统需要支持多种登录方式如：口令认证、域认证（口令认证通常是去数据库中验证用户，而域认证则是需要到微软的域中验证用户）。那么自然的做法就是建立一个各种登录方式都适用的接口，如下图所示：




```
public interface Login {
    //登录验证
    public boolean verify(String name , String password);
}
```

```
public class DomainLogin implements Login {

    @Override
    public boolean verify(String name, String password) {
        // TODO Auto-generated method stub
        /**
         * 业务逻辑
```

```
         */
        return true;
    }

}
```

```
public class PasswordLogin implements Login {

    @Override
    public boolean verify(String name, String password) {
        // TODO Auto-generated method stub
        /**
         * 业务逻辑
         */
        return true;
    }

}
```

我们还需要一个工厂类LoginManager，根据调用者不同的要求，创建出不同的登录对象并返回。而如果碰到不合法的要求，会返回一个Runtime异常。

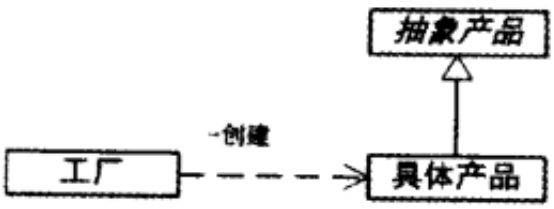
```
public class LoginManager {
    public static Login factory(String type){
        if(type.equals("password")){

            return new PasswordLogin();

        }else if(type.equals("passcode")){

            return new DomainLogin();

        }else{
            /**
             * 这里抛出一个自定义异常会更恰当
             */
            throw new RuntimeException("没有找到登录类型");
        }
    }
```

```
}
```

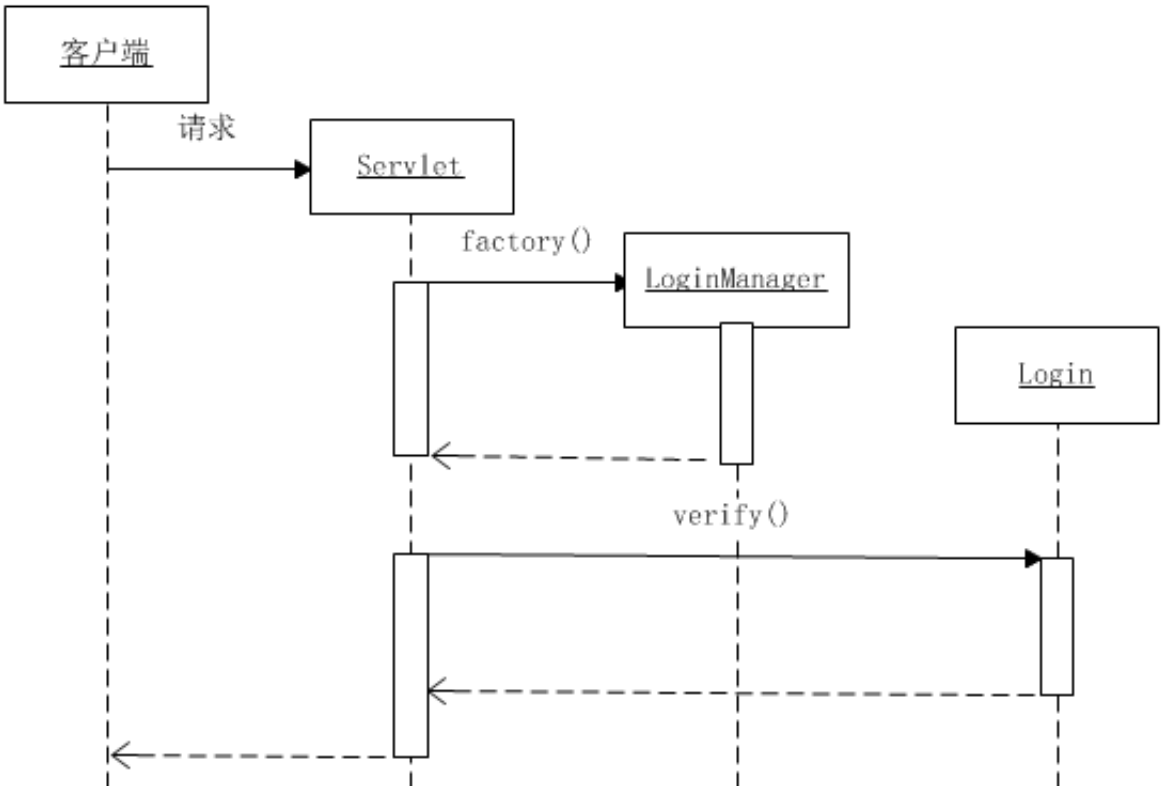
测试类：

```
public class Test {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String loginType = "password";
        String name = "name";
        String password = "password";
        Login login = LoginManager.factory(loginType);
        boolean bool = login.verify(name, password);
        if (bool) {
            /**
             * 业务逻辑
             */
        } else {
            /**
             * 业务逻辑
             */
        }
    }
}
```

简单工厂模式的结构如下图：



我们可以设想一下真实的场景，如果把上面的Test当做一个servlet的话，当客户端发起登录请求——>请求交给服务端的Servlet——>Servlet根据客户端传递的loginType调用工厂类LoginManager的factory()方法——>factory()方法根据参数loginType创建相应的登录验证类(DomainLogin或PasswordLogin)并返回——>登录验证类调用方法verify()验证用户名密码是否正确



假如不使用简单工厂模式则验证登录Servlet代码如下（假设Test为一个Servlet，变量loginType、name、password表示从客户端传递过来的参数）：

```
public class Test {
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        String loginType = "password";
        String name = "name";
        String password = "password";
        //处理口令认证
        if(loginType.equals("password")){
            PasswordLogin passwordLogin = new PasswordLogin();
            boolean bool = passwordLogin.verify(name, password);
            if (bool) {
                /**
                 * 业务逻辑
                 */
            } else {
                /**
                 * 业务逻辑
                 */
```

```
            }
        }
        //处理域认证
        else if(loginType.equals("passcode")){
            DomainLogin domainLogin = new DomainLogin();
            boolean bool = domainLogin.verify(name, password);
            if (bool) {
                /**
                 * 业务逻辑
                 */
            } else {
                /**
                 * 业务逻辑
                 */
            }
        }else{
            /**
             * 业务逻辑
             */
        }
    }
}
```

上面的代码会不会很蛋疼啊。。。呵呵

《JAVA与模式》一书中使用java.text.DataFormat类作为简单工厂模式的典型例子叙述。

简单工厂模式的优点

模式的核心是工厂类。这个类含有必要的逻辑判断，可以决定在什么时候创建哪一个登录验证类的实例，而调用者则可以免除直接创建对象的责任。简单工厂模式通过这种做法实现了对责任的分割，当系统引入新的登录方式的时候无需修改调用者。

简单工厂模式的缺点

这个工厂类集中了所以的创建逻辑，当有复杂的多层次等级结构时，所有的业务逻辑都在这个工厂类中实现。什么时候它不能工作了，整个系统都会受到影响。