适配器模式实验

**【实验目的及要求】**

1. 掌握适配器模式的结构与实现；
2. 学会使用适配器设计模式开发应用程序；
3. 明白适配器设计模式的扩展应用。

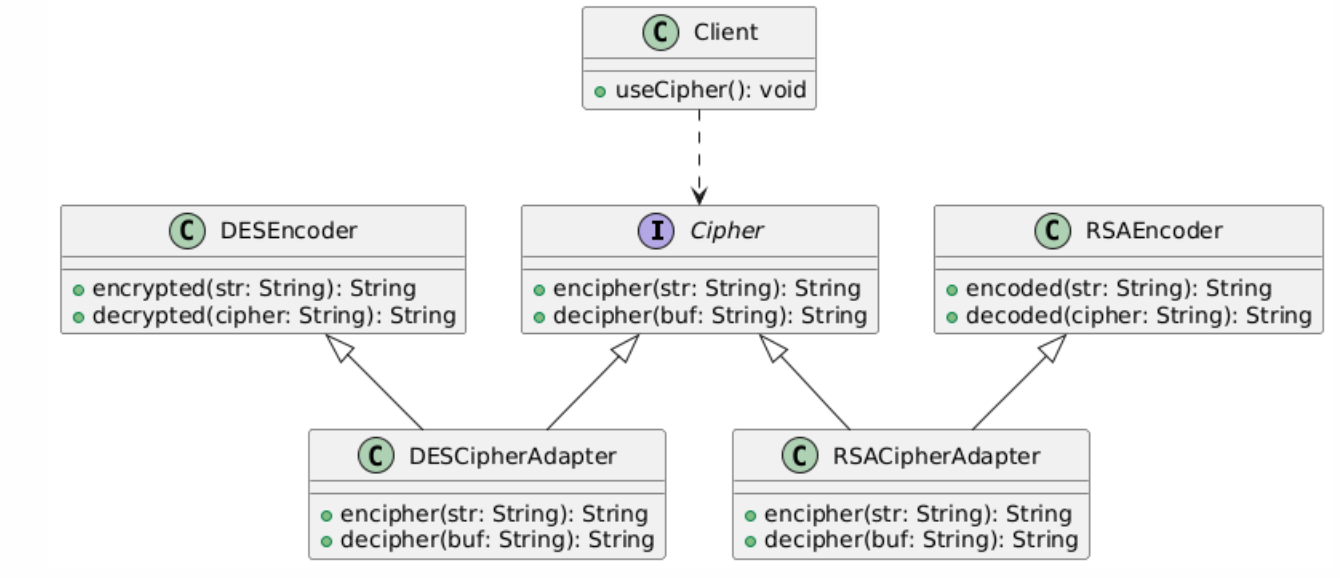
**【实验内容】**

**某系统需要提供一个加密模块，将用户信息加密之后再存储在数据库中，系统已经定义好了数据库操作类。为了提高效率，现需要重用已有的加密算法，这些加密算法已经封装，有些甚至没有源代码。要求不修改现有类的基础上重用这些加密算法。现使用适配器模式完成该设计。**

**【实验步骤】**

1. **类图**

**根据下面的代码画出类图。**

****

1. **实现代码**

**(1)已提供的适配者类**

**DESEncoder（DES加密算法类）**

**package** design.adapter;

**public** **class** DESEncoder {

**public** String encrypted(String str) {

String cipher = **null**;

/\*\*

\* ..... DES加密代码（省略）

\*/

System.***out***.println("使用DES算法进行加密！");

**return** cipher;

}

**public** String decrypted(String cipher) {

String content = **null**;

/\*\*

\* ..... DES解密代码（省略）

\*/

System.***out***.println("使用DES算法进行解密！");

**return** content;

}

}

**RSAEncoder（RSA加密算法类）**

**package** design.adapter;

**public** **class** RSAEncoder {

**public** String encoded(String str) {

String cipher = **null**;

/\*\*

\* .....RSA加密代码（省略）

\*/

System.***out***.println("使用RSA算法进行加密！");

**return** cipher;

}

**public** String decoded(String cipher) {

String content = **null**;

/\*\*

\* ..... RSA解密代码（省略）

\*/

System.***out***.println("使用RSA算法进行解密！");

**return** content;

}

}

**(2)目标类**

**Cipher类（目标类）**

**package** design.adapter;

//目标类

**public** **interface** Cipher {

**public** String encipher(String str);

**public** String decipher(String buf);

}

**(3)适配器**

**DESCipherAdapter**

**package** design.adapter;

//DES加密算法适配器类

**public** **class** DESCipherAdapter **extends** DESEncoder **implements** Cipher {

@Override

public String encipher(String str) {

return encrypted(str);

}

@Override

public String decipher(String buf) {

return decrypted(buf);

}

}

**RSACipherAdapter**

**package** design.adapter;

//DES加密算法适配器类

**public** **class** RSACipherAdapter **extends** DESEncoder **implements** Cipher {

@Override

public String encipher(String str) {

return encrypted(str);

}

@Override

public String decipher(String buf) {

return decrypted(buf);

}

}

**(4)工具类及配置文件**

**XMLUtil类**

**package** design.adapter;

**import** javax.xml.parsers.\*;

**import** org.w3c.dom.\*;

**import** java.io.\*;

**public** **class** XMLUtil {

//该方法用于从XML配置文件中提取具体类类名，并返回一个实例对象

**public** **static** Object getBean() {

**try** {

//创建DOM文档对象

DocumentBuilderFactory dFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = dFactory.newDocumentBuilder();

Document doc;

doc =builder.parse(**new** File("src//design//adapter//config.xml"));

//获取包含类名的文本结点

NodeList nl = doc.getElementsByTagName("className");

Node classNode=nl.item(0).getFirstChild();

String cName=classNode.getNodeValue();

//通过类名生成实例对象并将其返回

Class c=Class.*forName*(cName);

Object obj=c.newInstance();

**return** obj;

}

**catch**(Exception e) {

e.printStackTrace();

**return** **null**;

}

}

}

**配置文件config.xml**

<?xml version=*"1.0"*?>

<config>

<className>design.adapter.DESCipherAdapter</className>

</config>

**(5)测试类**

**Test**

**package** design.adapter;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Cipher cipher = (Cipher)XMLUtil.*getBean*();

cipher.encipher("加密字符串");

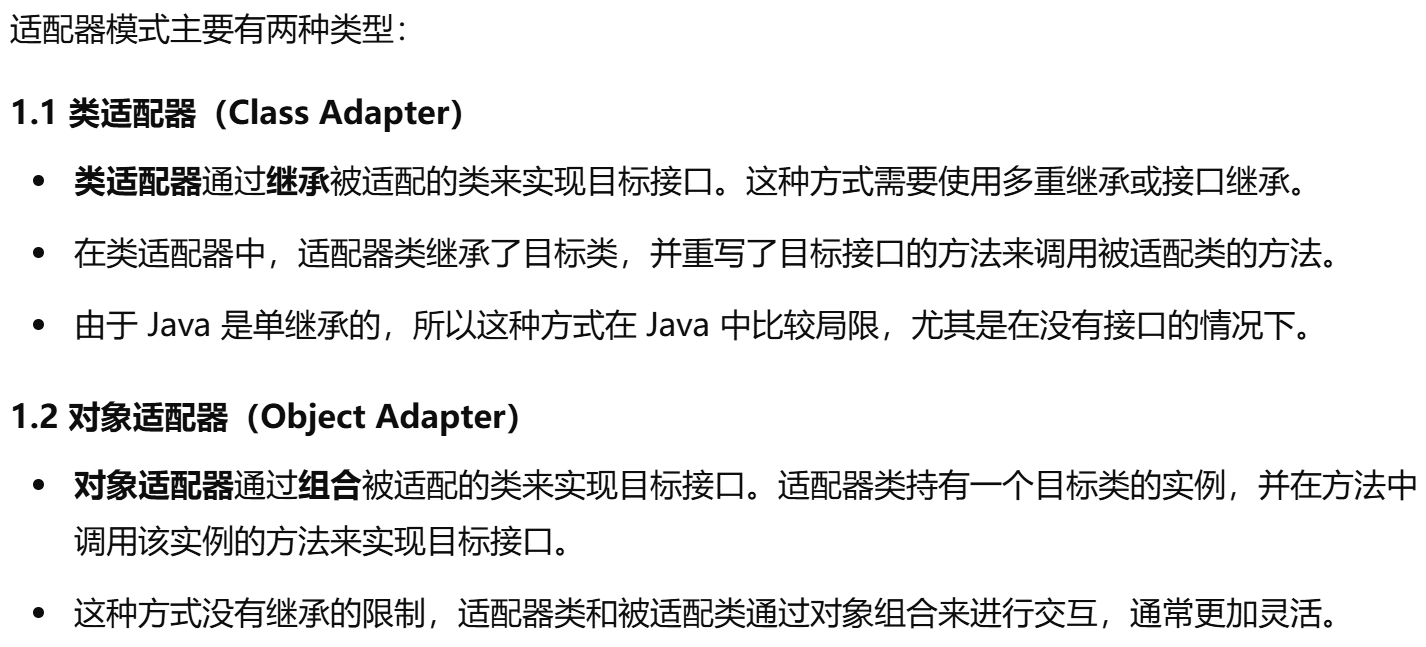
cipher.decipher("解密字符串");

}

}

**【问题】**

1.适配器有哪些类型



2.程序采用哪种适配器模式？能使用其他适配器模式吗？如果可以，请编程实现；如果不可以，请说明理由。

本程序采用的是 **类适配器模式**。可以使用 **对象适配器模式** 来实现该设计。

这时，不继承适配者，把适配者引入到适配器，依然实现目标类。

public class RSACipherObjectAdapter implements Cipher{

private RSAEncoder rsaEncoder;

public RSACipherObjectAdapter(RSAEncoder rsaEncoder) {

this.rsaEncoder = rsaEncoder;

}

@Override

public String encipher(String str) {

return rsaEncoder.encoded(str);

}

@Override

public String decipher(String buf) {

return rsaEncoder.decoded(buf);

}

}

public class DESCipherObjectAdapter implements Cipher{

private DESEncoder desEncoder;

@Override

public String encipher(String str) {

return desEncoder.encrypted(str);

}

@Override

public String decipher(String buf) {

return desEncoder.decrypted(buf);

}

}

修改一下配置文件：

<?xml version="1.0"?>  
<config>  
 <className>com.fancy.experiment8\_adapter.DESCipherObjectAdapter</className>  
</config>

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | **实验总结：**  本实验深入探讨了适配器模式的结构与实现，通过将现有的加密算法类与目标接口 Cipher 进行适配，成功掌握了适配器模式的使用。适配器模式主要有两种类型：类适配器和对象适配器，分别通过继承和组合的方式实现。我们在实际应用中，通过对象适配器模式，成功实现了对 DESEncoder 和 RSAEncoder 的适配，确保加密算法能够通过统一的 Cipher 接口进行调用。通过该实验，我不仅学会了如何将不兼容的类通过适配器模式进行整合，还理解了适配器模式在扩展应用中的重要性，例如在现有代码的基础上实现对不同加密算法的支持。适配器模式在软件开发中提供了更高的灵活性和可扩展性，尤其适用于需要兼容不同接口或模块的场景。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |