**《软件体系结构与设计模式》**

**作业报告**

**作业名称：作业1 创建型设计模式应用**

**授课教师： 毛斐巧**

**报告人： 郑彦薇 学号：2020151022班级： 软件工程01班**

**报告提交时间： 2023/4/12**

**成 绩：**

**1.作业内容与要求**：

(1) 使用简单工厂模式设计一个可以创建不同几何形状(Shape)(例如圆形(Circle)、矩形(Rectangle)和三角形(Triangle)等)的绘图工具类，每个集合图形均具有绘制方法draw( )和擦除方法erase( )，要求在绘制不支持的几何图形时，抛出一个unsupportedShapeException异常。绘制类图并编程模拟实现。

(2) 在某网络管理软件中，需要为不同的网络协议提供不同的连接类，例如针对POP3协议的连接类POP3Connection、针对IMAP协议的连接类IMAPConnection、针对HTTP协议的连接类HTTP Connection等。由于网络连接对象的创建过程较为复杂，需要将其创建过程封装到专门的类中，该软件还将支持更多类型的网络协议，现采用工厂方法模式进行设计，绘制类图并编程模拟实现。

**2.解答报告正文**

**请给出1.中（1）和（2）题的设计类图和编程模拟实现代码。**

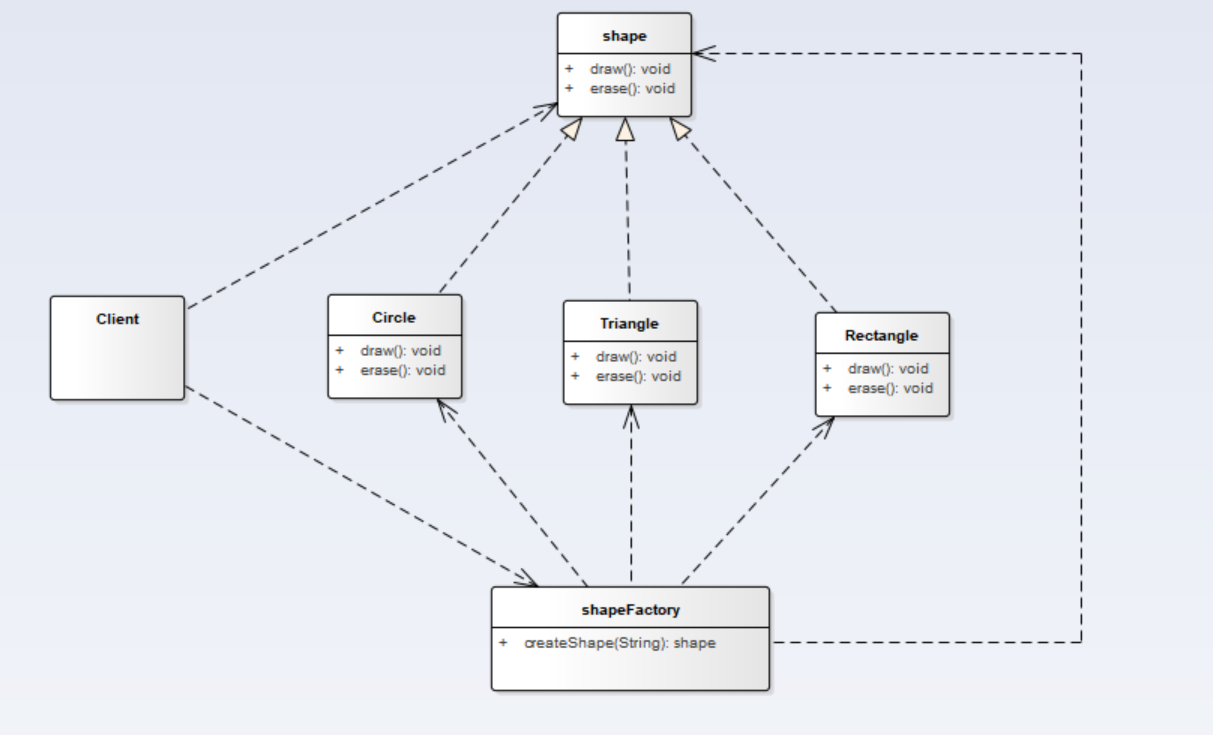
**2.1. 绘图工具类**

**2.1.1. 设计绘图工具类的类图**

①设计思路：按照简单工厂模式，定义一个shapeFactory类负责对输入进行判断，创建其他类的实例（这里包括Circle类、Rectangle类以及Triangle类）或抛出异常处理。还需要定义shape接口，包含图形的绘制方法draw()和擦除方法erase()，在shapeFactory进行调用，创建相应的shape对象（即上述所提的其他类实例）。Circle类、Rectangle类以及Triangle类需要继承shape类中的方法，然后再各自实现相应的绘制方法和擦除方法。

工厂运作方式：用户输入需要创建的图形，客户端调用shapeFactory中的创建方法，创建一个shape对象，并调用其中的draw()方法和erase()方法；若用户输入的图形不为以上3种，则抛出异常。

②按照以上思路设计的类图如下：



**2.1.2. 编程模拟实现代码**

根据上述所得类图，可以得到代码模拟如下：

接口shape.java：

public interface shape {  
 void draw();  
 void erase();  
}

圆形Circle.java（其他两个图形同理）：

public class Circle implements shape{  
 public Circle(){  
 System.*out*.println("圆形创建成功");  
 }  
 @Override  
 public void draw() {  
 System.*out*.println("绘制圆形");  
 }  
  
 @Override  
 public void erase() {  
 System.*out*.println("擦除圆形");  
 }  
}

简单工程shapeFactory.java:

public class shapeFactory {  
 static public shape createShape(String name) throws UnsupportedShapeException{  
 name = name.toLowerCase();  
 shape res;  
 switch (name){  
 case "circle":  
 res = new Circle();  
 break;  
 case "triangle":  
 res = new Triangle();  
 break;  
 case "rectangle":  
 res = new Rectangle();  
 break;  
 default:  
 throw new UnsupportedShapeException();  
 }  
 return res;  
 }  
}

抛出异常UnsupportedShapeException.java：

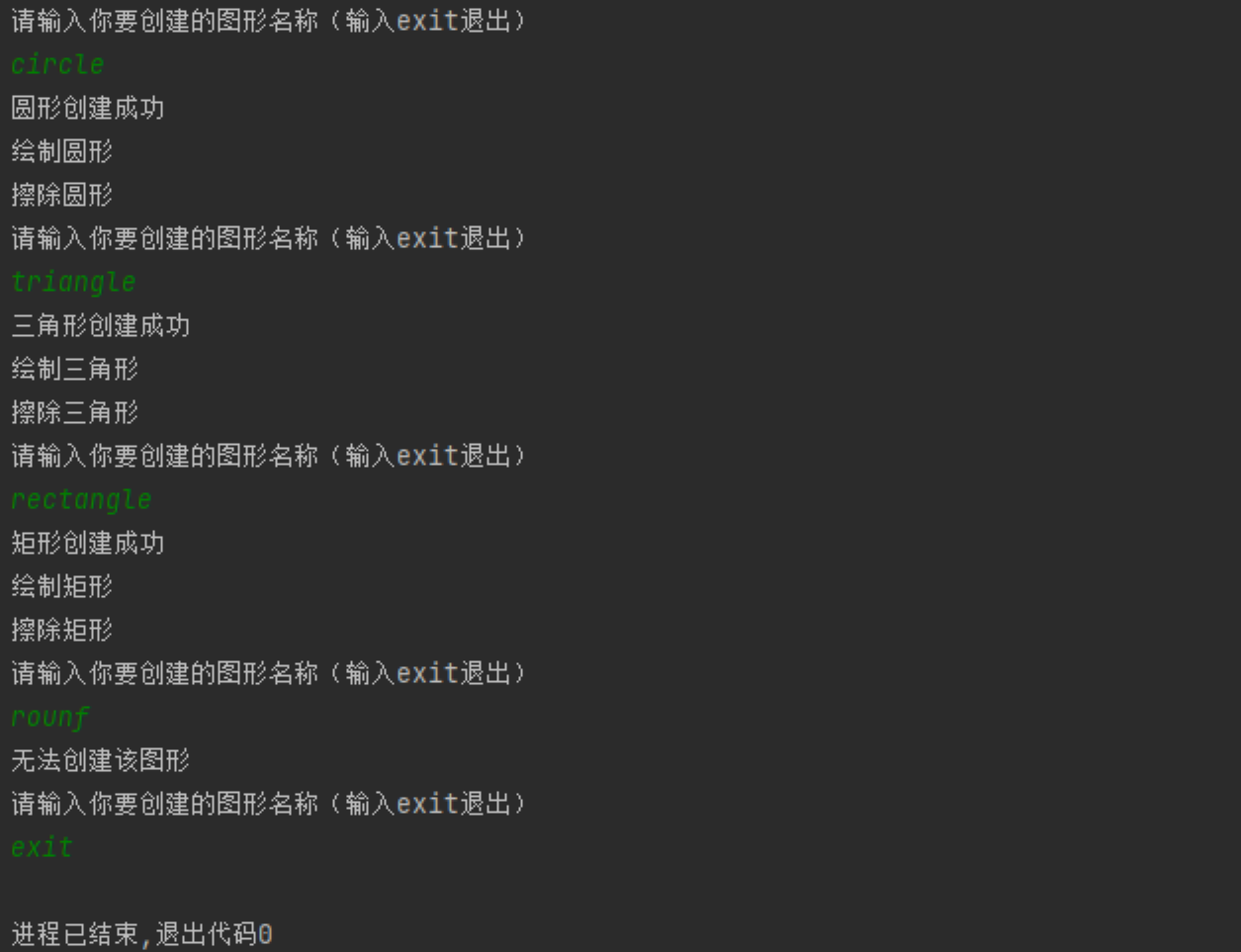
public class UnsupportedShapeException extends Exception{  
 @Override  
 public String toString() {  
 return "无法创建该图形";  
 }  
}

客户端client.java：

import java.util.Scanner;  
public class Client {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 String shapeName = "";  
 while (true){  
 System.*out*.println("请输入你要创建的图形名称（输入exit退出）");  
 // 接收输入图形名称  
 shapeName = scanner.next();  
 // 如果为“exit”程序退出  
 if (shapeName.equals("exit"))  
 break;  
 try {  
 // 使用简单工厂创建shape对象  
 shape s = shapeFactory.*createShape*(shapeName);  
 // 调用方法  
 s.draw();  
 s.erase();  
 } catch (UnsupportedShapeException e) {  
 System.*out*.println(e.toString());  
 }  
 }  
 }  
}

测试该绘图工具类的实现：

运行客户端client，得到运行结果如下：

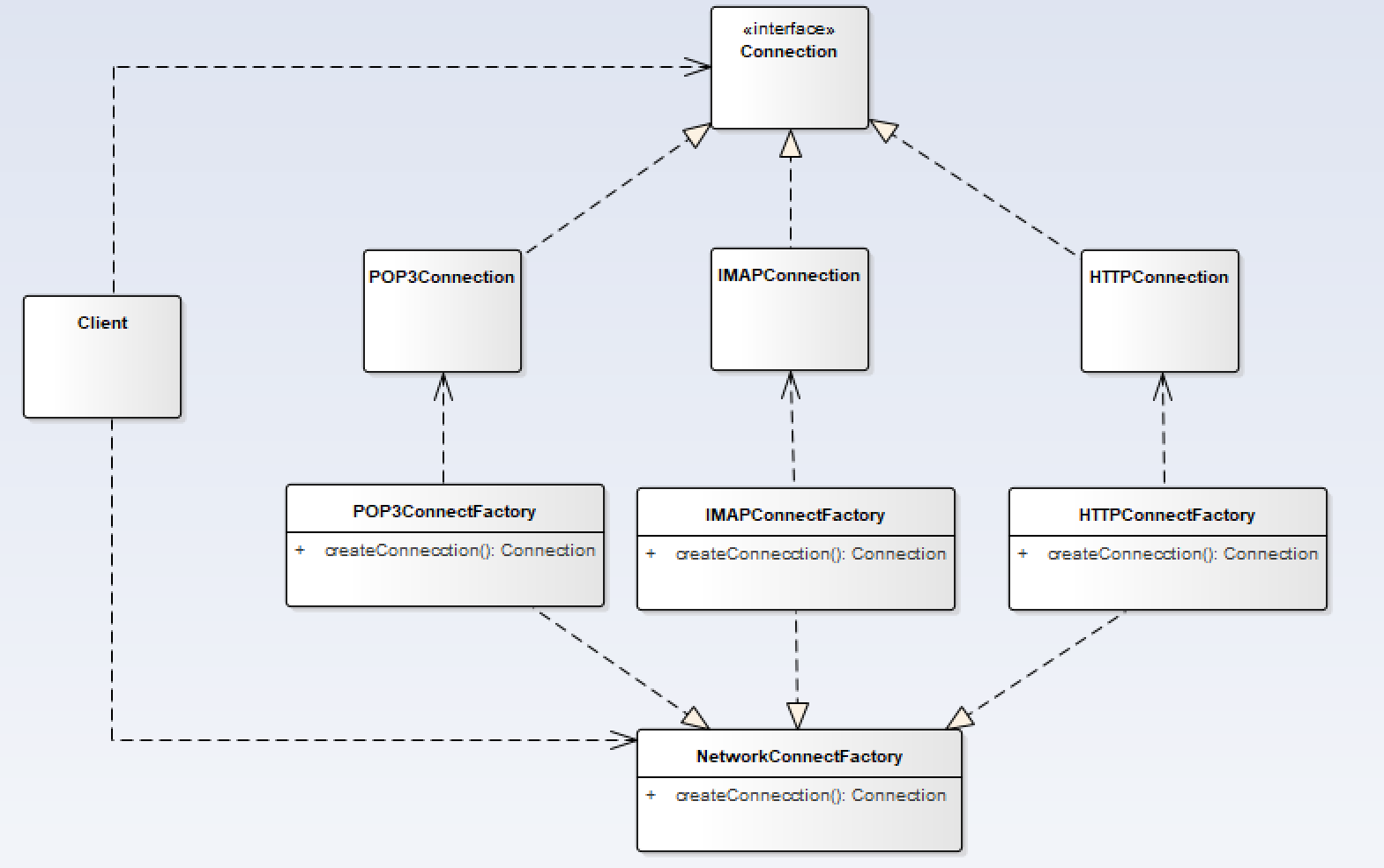


**2.2. 网络连接类**

**2.2.1. 设计网络连接类的类图**

①设计思路：按照工厂模式，工厂父类负责定义创建产品对象的公共接口，工厂子类负责生成具体的产品对象。要为不同的网络协议提供不同的连接类，可以设定一个工厂父类为NetworkConnectFactory，包含创建连接方法createConnection。然后为每个协议提供一个子工厂（如POP3ConnectionFactory），继承NetworkConnectFactory。将不同协议下的网络连接操作延迟到工厂子类中完成，当增加不同的协议时，直接增加具体的工厂子类，原有工厂不做修改。

②根据以上思路设计的类图如下：



**2.2.2. 编程模拟实现代码**

根据上述所得类图，可以得到代码模拟如下：

连接接口Connection.java：

public interface Connection {  
}

具体连接类POP3Connection.java（其他两个同理）：

public class POP3Connection implements Connection{  
}

工厂父类接口NetworkConnectFactory.java：

public interface NetworkConnectFactory {  
 Connection createConnection();  
}

创建网络连接的具体工厂子类POP3ConnectFactory.java（其他两个同理）：

public class POP3ConnectFactory implements NetworkConnectFactory {  
 public Connection createConnection(){  
 System.*out*.println("创建POP3连接");  
 return new POP3Connection();  
 }  
}

客户端Client.java：

为了使创建不同网络连接工厂时不用修改到源代码，使用java的反射机制来创建对应的工厂子类。

import java.util.Scanner;  
  
public class Client {  
 public static void main(String[] args) throws Exception{  
 Scanner scanner = new Scanner(System.*in*);  
 String factoryName;  
 while (true){  
 System.*out*.println("请输入你要创建的协议连接工厂名（输入exit退出）");  
 //接收输入的协议名，使用反射机制创建工厂  
 factoryName = scanner.next();  
 if(factoryName.equals("exit"))  
 break;  
 Class<?> factoryClass = Class.*forName*(factoryName);  
 NetworkConnectFactory factory = (NetworkConnectFactory) factoryClass.newInstance();  
 //使用工厂创建具体网络连接对象  
 Connection connection = factory.createConnection();  
 }  
 }  
}

测试该网络连接类的实现：

运行客户端client，得到运行结果如下：

