## 第三章 简单工厂模式

1. **在简单工厂模式中，如果需要增加新的具体产品，通常需要修改（C）的源代码。**
   1. 抽象产品类
   2. 其他具体产品类
   3. 工厂类
   4. 客户类
2. **以下关于简单工厂模式的叙述错误的是（C）。**
   1. 简单工厂模式可以根据参数的不同返回不同产品类的实例。
   2. 简单工厂模式专门定义一个类来负责创建其他类的实例，被创建的实例通常都具有共同的父类。
   3. 简单工厂模式可以减少系统中类的个数，简化系统的设计，使得系统更易于理解
   4. 系统的扩展困难，在添加新的产品类时需要修改工厂的业务逻辑，违背了开闭原则
3. **以下代码使用了（A）模式。**

|  |
| --- |
| public abstract class Product{  public abstract void process();  }  public class ConcreteProductA extends Product{  public void process(){…}  }  public class ConcreteProductB extends Product{  public void process(){…}  }  public class Factory{  public static Product creatProduct(char type){  switch(type){  case ‘A’:  return new ConcreteProductA();break;  case ‘B’:  return new ConcreteProductB();break;  …  }  }  } |

* 1. Simple Factory
  2. Factory Method
  3. Abstract Factory
  4. 未用任何设计模式

1. **使用简单工厂模式模拟女娲（Nvwa）造人（Person），如果向人的工厂方法传入参数“M”，则返回一个男人（Man）对象，如果传入参数“W”，则返回一个女人（Woman）对象，绘制相应的类图并使用java语言模拟实现该场景。现需要增加一个新的机器人（Robot）类，如果传入参数“R”，则返回一个机器人对象，对代码进行修改并注意“女娲”类的变化。**

参考类图如下：

其中Nvwa类充当工厂类，其中定义了工厂方法makePerson（），Person类充当

抽象产品类，Man、Woman和Robot充当具体产品类。代码见代码实现文件夹。

1. **使用简单工厂模式设计一个可以创建不同几何图形（Shape）的绘图工具类，例如圆形（Circle）、矩形（Rectangle）和三角形（Triangle）等，每个几何图形均具有绘制draw（）和擦除erase（）两个方法，要求在绘制不支持的几何图形是抛出一个UnSupportShapeException异常，绘制类图并使用java语言编程模拟实现。**

参考类图如下：

其中，Shape 接口充当抽象产品，其子类 Circle、Rectangle 和 Triangle 等充当具体产品， ShapeFactory 充当工厂类。代码见代码实现文件夹。