文档处理

1. 一个工厂类，produce方法使用字符串字面量判断类型，返回不同的文档实例

```
if("paper".equals(type)) {
return new Paper();
} else if("magazine".equals(type)) {
return new Magazine();
} ...
```

问题：字面量

2. 使用常量判断类型

```
class Document {
public static final int PAPER = 1;
public static final int MAGAZINE = 2;
...
}
```

问题：**常量枚举**是int类型的，它不是类型安全的
类型安全比较难以解释，比如这里的Document只有两种，这个类型也只有两个合法取值，但是我们定义produce时传入的是int类型的值，显然int的取值范围是INT_MIN~INT_MAX，与我们的Document类型不同，对这些未知取值的处理Java是不管的。
Java中的**enum是类型安全的**。

3. 在接口中枚举文档类型或者使用enum

```
interface Document {
Document PAPER = new Document() {};
Document MAGAZINE = new Document() {};
...
}
```

问题：因为所有逻辑都在一个方法中（一堆if..else），所以客户端程序比较臃肿

4. 接口工厂

```
interface Document { Document produce(); }
class Paper implements Document { Document produce() {} }
class Magazine implements Document { Document produce() {} }
```

问题：代码大量重复

5. 抽象工厂
   同版本4，但是Document改为抽象类，实现子类的共有功能
6. 使用聚合组合（桥）

   每个Document对应一个ProcessUnit，并将process委托给其实现，Document本身只定义一个默认的处理
   问题：process粒度太大（干的活太多）
7. 精炼提取process过程（模板方法）

```
final process() {
preProcess();
core();
postProcess();
}
abstract preProcess() {}
abstract postProcess() {}
```

   core部分是不变的，子类通过实现preProcess()或postProcess()来达成扩展