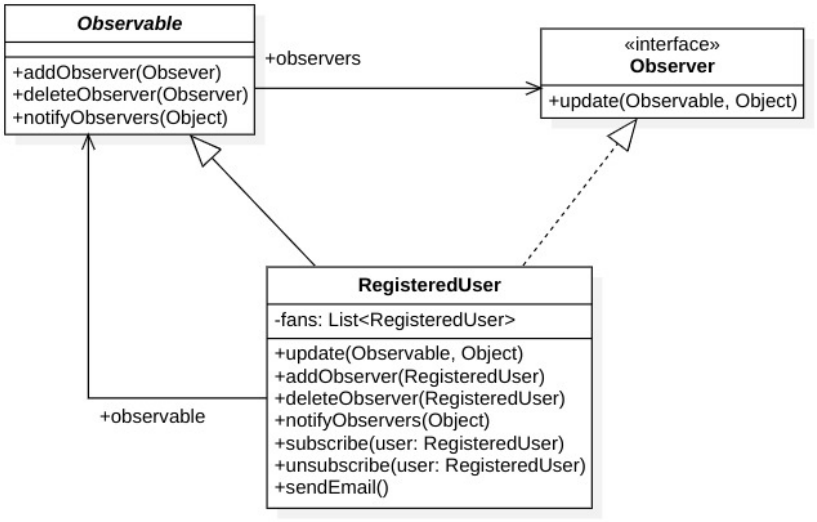
设计模式

设计模式是人们在软件开发实践中总结出的一些设计方案，用于高效、高质量地解决特定类型的问题。在 WeBlog 个人博客系统中，涉及到如下设计模式。

观察者模式

在 WeBlog 中，可以将注册用户拥有的博文视为该注册用户（即博主）的状态。当博主发布博文时，博文状态变为公开时，这一动作将通过邮件通知给关注该博主的用户。这里蕴含了一个观察者模式：主题是被关注的博主，观察者是关注了博主的用户。由于案例使用 Java 编程语言，因此我们可以使用语言自带的观察者模式相关的类；其中主题是一个名为“可观察的”（Observable）的抽象类。因此案例中这一部分可以设计为：

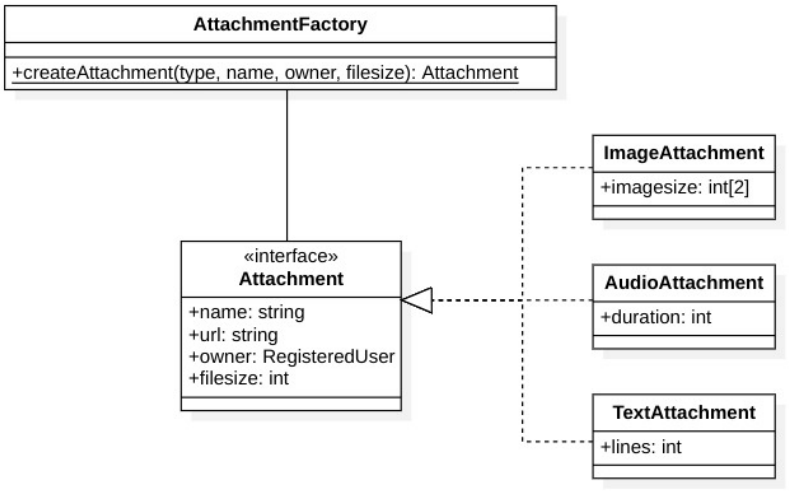


此案例的特别之处在于观察者和主题是同一个类，即注册用户，因此注册用户同时实现了 Observable 和 Observer 两个接口或抽象类。对于两个注册用户 userA 和 userB，当 userA 想关注 userB 时，将调用 userA.subscribe(userB)，其中 userB 是主题，userA 是观察者；方法内部将调用 userB.addObserver(userA)，将 userA 添加到 userB.fans 中。取消关注也是同理，当 userA 想取消关注 userB 时，将在方法 userA.unsubscribe(userB) 方法内部调用 userB.deleteObserver(userA)，从 userB.fans 中删除 userA。当 userB 发表了新的博文时，方法 userB.notifyObservers(newPost) 将会被调用，并依次调用 userB.fans 中每个注册用户的 update(newPost) 方法，该方法的实现中包含了 sendEmail()，如此 userA.update(newPost) 被调用时，将向 userA 的邮箱发送一条新通知。

工厂模式

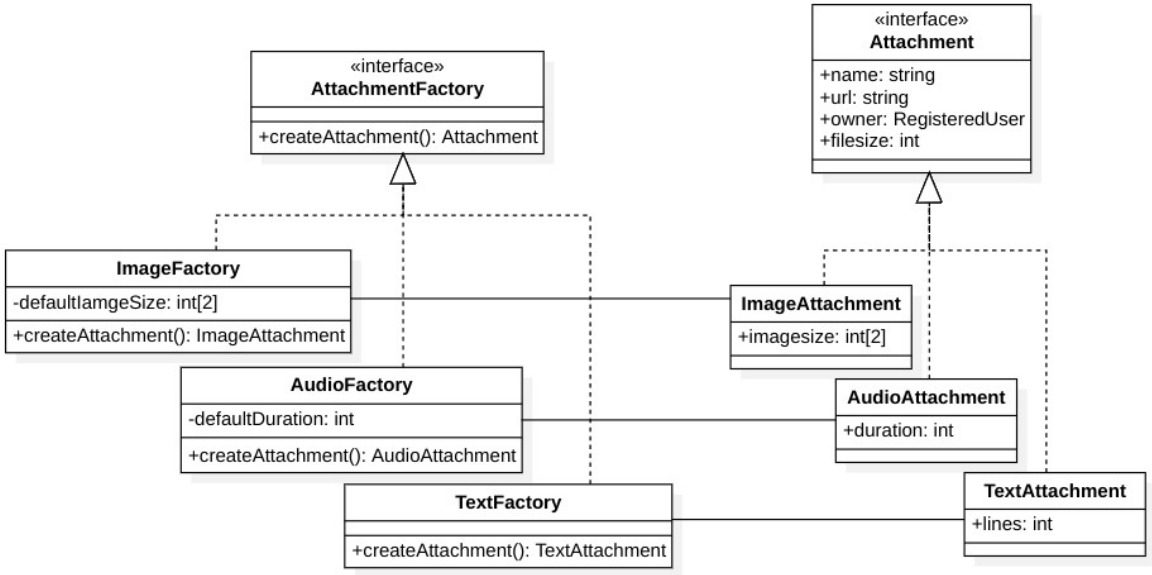
目前 WeBlog 个人博客系统中，Attachment 类被设计为只能是图片附件，被插入到博文中，或者作为注册用户的头像。如果我们允许用户上传多种类型的附件，如图片、音频、文本文件三种，则会需要使用到工厂模式。

我们首先可以使用简单工厂模式，对创建附件的逻辑做初步的设计，如下图所示：



其中，工厂类是 AttachmentFactory, 抽象产品类是 Attachment，具体的产品类分别是 ImageAttachment, AudioAttachment 和 TextAttachment。 通常情况下，简单工厂模式中的工厂中，用于创建产品的方法是静态方法。当用户需要创建附件时，只需调用 AttachmentFactory 中的静态方法 createAttachment() ，并根据是图片、音频还是文本附件确定该方法的第一个参数 type。如若需要创建一个文本附件，可以在业务中调用 AttachmentFactory.createAttachment("TEXT", "textfile.txt", userA, 65536)。注意到调用此方法时没有要求提供 url，而是由方法内部生成，这就隔离了业务代码和创建产品的细节。

然而上述设计还有待进一步完善。比如在工厂创建附件时，需要分别为三种类型的附件初始化（此例中即为 imagesize、duration、lines 赋初值，但对于更一般的情况产品的初始化可能是非常复杂的过程），而当前的设计中，三种附件初始化的过程都写在 AttachmentFactory.createAttachment() 方法中，没有将它们很好地隔离。利用工厂方法模式，我们可以重新做出如下设计：

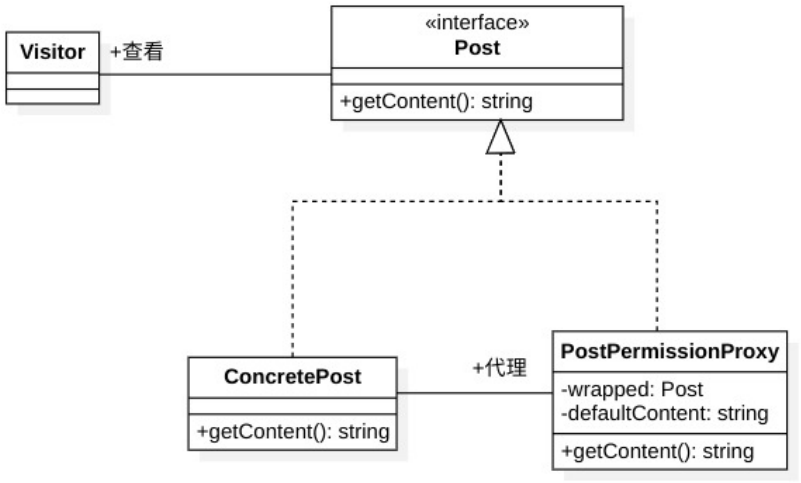


这样，将创建三种附件的工厂抽象成三个类，就可以将初始化的代码分别写在三个工厂

中，避免了功能逻辑上的耦合，也方便在将来继续扩展新的类型的附件。

代理模式

在 WeBlog 个人博客系统中，博主可以设置自己的博文是否可以被公开。我们希望实现这样的功能：当用户想要通过博文展示界面查看博文详情时，如果博文被设置为非公开，且当前用户并非博文的作者时，将不会展示博文内容，并提示“无权查看此博文”。在此案例中涉及到对博文访问权限的控制，最适合用代理模式来解决此问题。



其中，ConcretePost 是具体的博文。系统被设计为无法直接访问到具体的博文。当访客想要查看某博文内容时，只能和 PostProxy 直接交互，即调取代理的 getContent()方法获取博文内容；PostPermissionProxy 的首先检查访客的权限，如果没有权限则直接返回 defaultContent 提示访客无权查看；如果具有权限，则代替访客调取具体博文的 getContent()方法，并返回其内容给访客。