

**软件体系结构作业12**

姓 名 ： 洪伟鑫

专 业 ： 软件工程

年 级 ： 2022级

学 号 ： 37220222203612

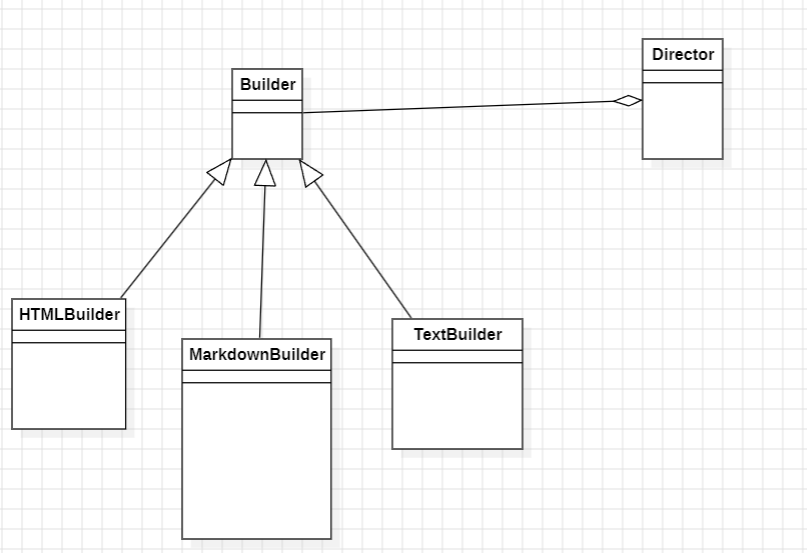
**2025年4月20日**

1. 修改本例，增加一个新的concrete的Builder。

添加了一个新的具体Builder类——MarkdownBuilder，它能够将内容输出为Markdown格式。现在我们可以通过三种方式生成文档：

1. 普通文本格式 (TextBuilder)
2. HTML格式 (HTMLBuilder)
3. Markdown格式 (MarkdownBuilder，这是我们新添加的)

这展示了Builder模式的灵活性，当需要添加新的构建方式时，只需创建一个新的具体Builder类并实现抽象Builder中定义的方法即可，而不需要修改Director类的代码。这符合开闭原则，对扩展开放，对修改关闭。

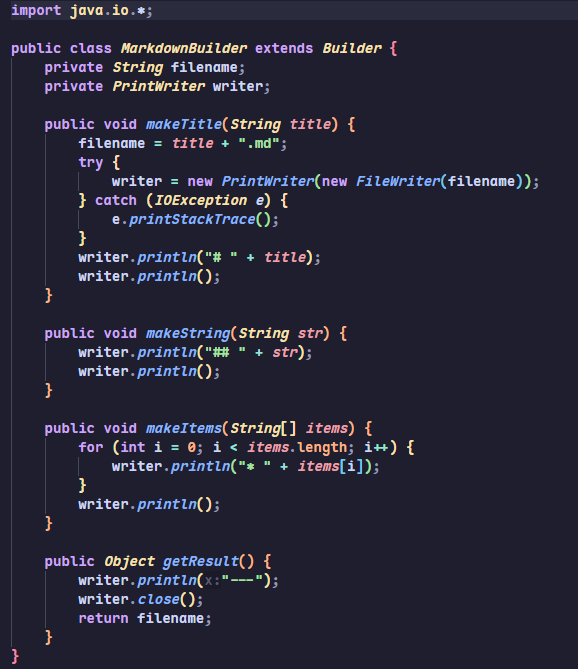


Builder设计模式，旨在将一个复杂对象的构建过程与其表示分离开来，使得同样的构建过程可以创建出不同的表示。模式的核心在于定义了一个抽象的 Builder 类（或接口），它声明了用于创建产品各个部分的抽象接口方法，例如 makeTitle(String title) 用于创建标题、makeString(String str) 用于创建普通文本内容、以及 makeItems(String[] items) 用于创建列表项。此外，Builder 还通常包含一个用于获取最终构建结果的方法，如 getResult()。

为了实现具体的构建逻辑，需要提供若干继承自抽象 Builder 的具体构建器类，例如 TextBuilder、HTMLBuilder 和 MarkdownBuilder。每一个具体构建器类都会实现 Builder 定义的抽象方法，负责以特定的格式（如纯文本、HTML 标签或 Markdown 语法）来构建产品的各个部分，并维护构建过程中的内部状态，最终通过 getResult() 方法返回相应格式的产品。

该模式还包含一个 Director 类，它并不负责各部分的具体创建，而是指导构建过程。Director 类通常包含一个 Builder 类型的成员变量，并通过其 construct() 方法来调用 Builder 接口中定义的一系列构建方法（makeTitle, makeString, makeItems 等），按照预定的顺序来组装产品。由于 Director 仅依赖于抽象的 Builder 接口，因此同一个 Director 可以与不同的具体 Builder 类（如 TextBuilder 或 HTMLBuilder）协作，从而构建出不同表示形式的最终产品，实现了构建算法与产品表示之间的解耦。

代码:





运行截图：



