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

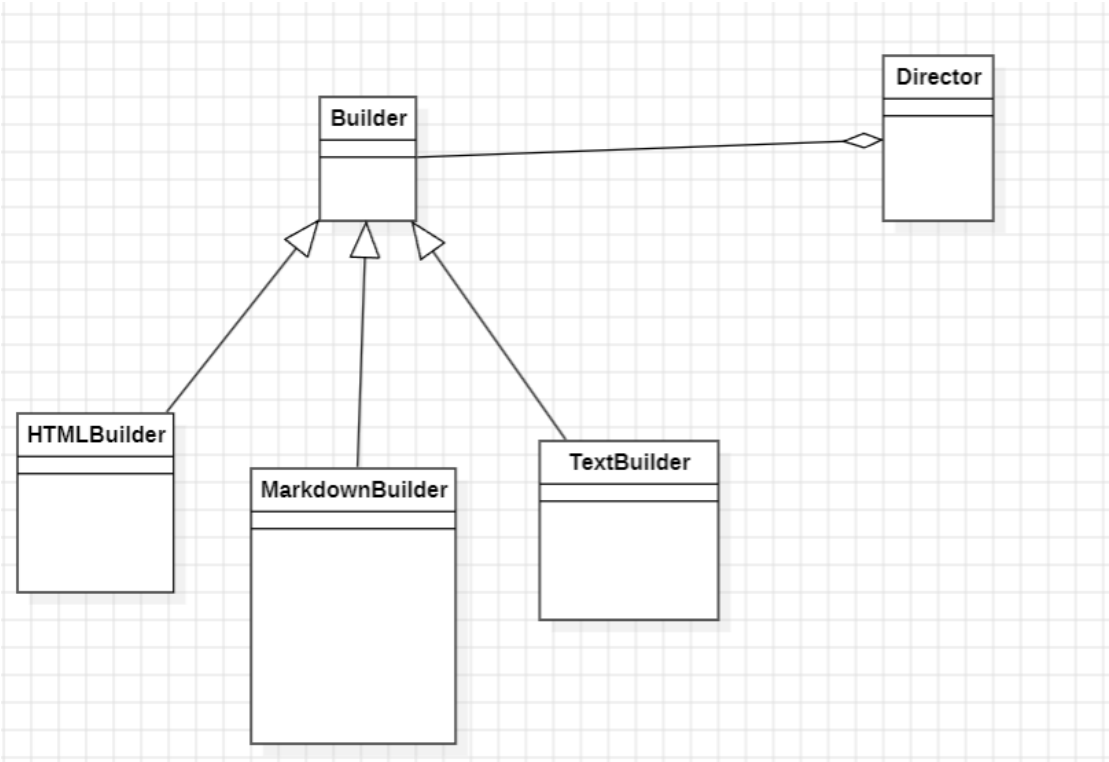
2025 年 4 月 20 日

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:

```java
import java.io.*;

public class MarkdownBuilder extends Builder {
    private String filename;
    private PrintWriter writer;

    public void makeTitle(String title) {
        filename = title + ".md";
        try {
            writer = new PrintWriter(new FileWriter(filename));
        } catch (IOException e) {
            e.printStackTrace();
        }
        writer.println("# " + title);
        writer.println();
    }

    public void makeString(String str) {
        writer.println("## " + str);
        writer.println();
    }

    public void makeItems(String[] items) {
        for (int i = 0; i < items.length; i++) {
            writer.println("* " + items[i]);
        }
        writer.println();
    }

    public Object getResult() {
        writer.println(x:"---");
        writer.close();
        return filename;
    }
}
```

```java
public class Main {
    public Main() {
    }

    public static void main(String[] var0) {
        if (var0.length != 1) {
            usage();
            System.exit(0);
        }

        Director var1;
        String var2;
        if (var0[0].equals("plain")) {
            var1 = new Director(new TextBuilder());
            var2 = (String)var1.construct();
            System.out.println(var2);
        } else if (var0[0].equals("html")) {
            var1 = new Director(new HTMLBuilder());
            var2 = (String)var1.construct();
            System.out.println("已产生" + var2 + "。");
        } else if (var0[0].equals("markdown")) {
            var1 = new Director(new MarkdownBuilder());
            var2 = (String)var1.construct();
            System.out.println("已产生" + var2 + "。");
        } else {
            usage();
            System.exit(0);
        }

    }

    public static void usage() {
        System.out.println("Usage: java Main plain    产生一般格式的文件");
        System.out.println("Usage: java Main html     产生HTML格式的文件");
        System.out.println("Usage: java Main markdown 产生Markdown格式的文件");
    }
}
```

运行截图：

```
PS E:\大三资料\大三下课程资料\体系结构\PPT\Code\Builder> java Main markdown
已产生Greeting.md。
```