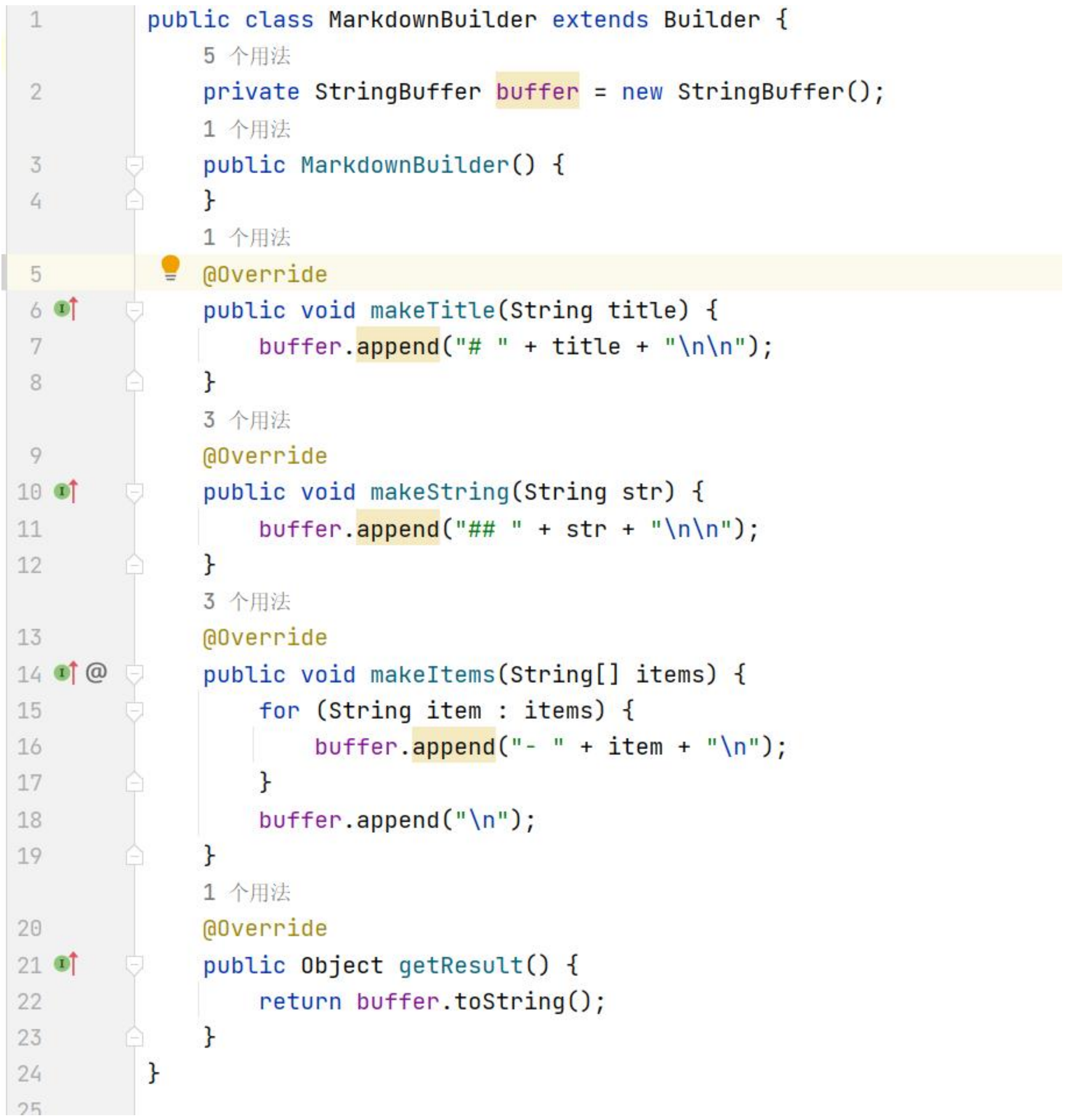
Homework：

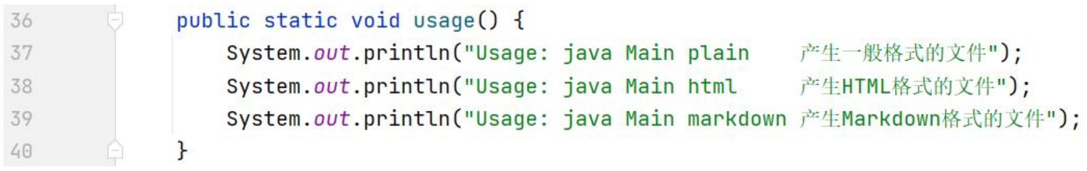
1、　　修改本例，增加一个新的 concrete 的 Builder。

答：增加一个用于生成 Markdown 格式文档的 Builder，MarkdownBuilder 类遵循原有的 Builder 模式结构，并实现了相应的 Builder 抽象方法。

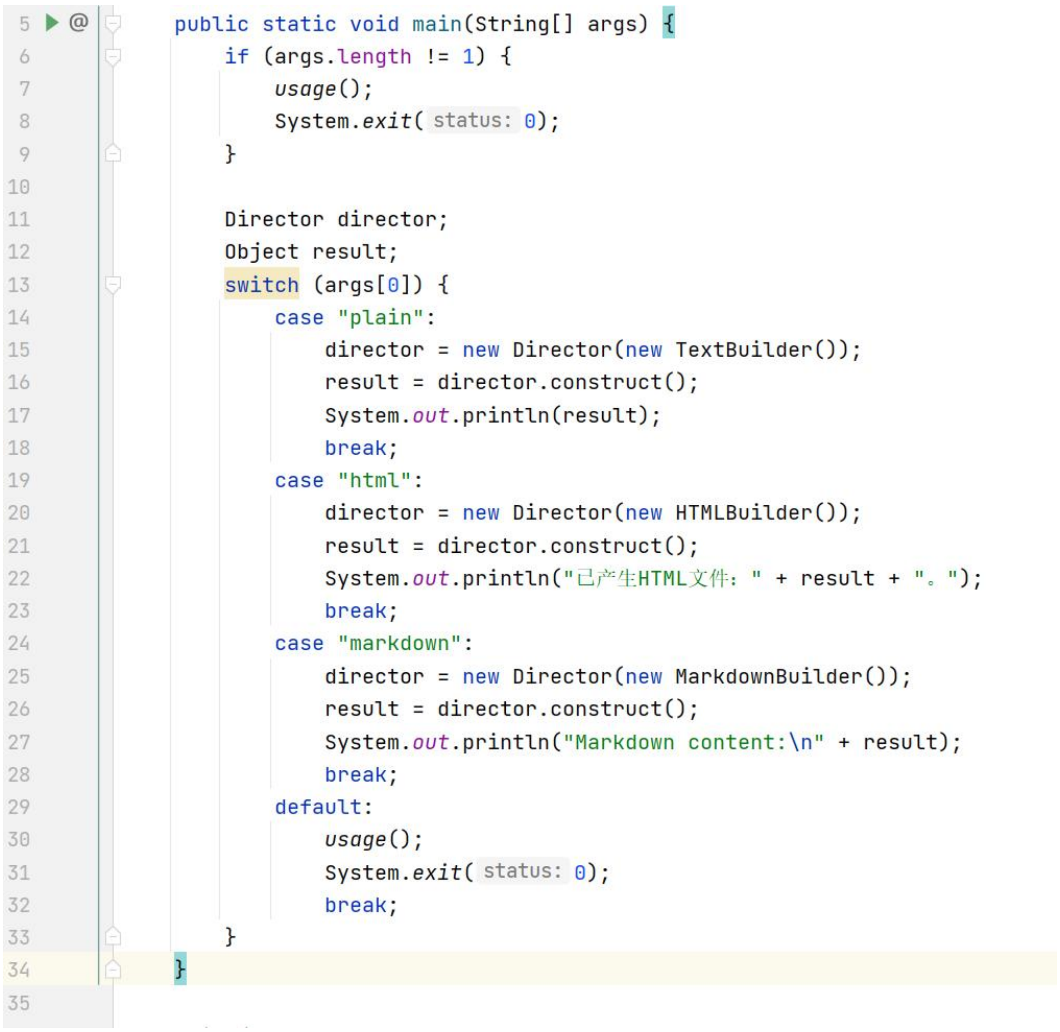


```java
public class MarkdownBuilder extends Builder {
    5 个用法
    private StringBuffer buffer = new StringBuffer();
    1 个用法
    public MarkdownBuilder() {
    }
    1 个用法
    @Override
    public void makeTitle(String title) {
        buffer.append("# " + title + "\n\n");
    }
    3 个用法
    @Override
    public void makeString(String str) {
        buffer.append("## " + str + "\n\n");
    }
    3 个用法
    @Override
    public void makeItems(String[] items) {
        for (String item : items) {
            buffer.append("- " + item + "\n");
        }
        buffer.append("\n");
    }
    1 个用法
    @Override
    public Object getResult() {
        return buffer.toString();
    }
}
```

修改 Main 函数以调用新增加的 MarkdownBuilder 类。



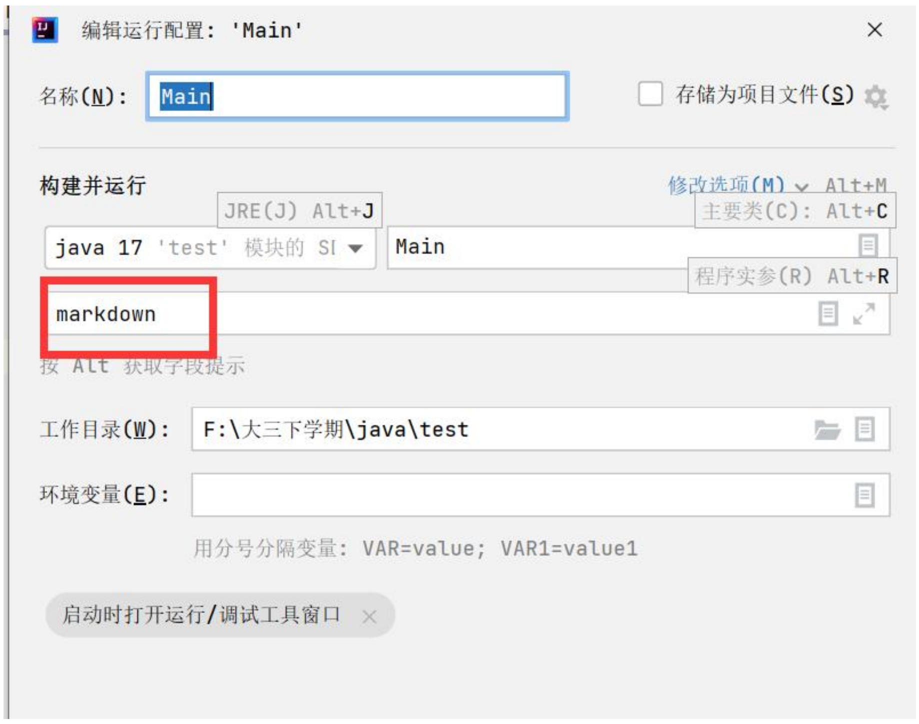
```java
    public static void usage() {
        System.out.println("Usage: java Main plain    产生一般格式的文件");
        System.out.println("Usage: java Main html     产生HTML格式的文件");
        System.out.println("Usage: java Main markdown 产生Markdown格式的文件");
    }
```



```java
    public static void main(String[] args) {
        if (args.length != 1) {
            usage();
            System.exit( status: 0);
        }

        Director director;
        Object result;
        switch (args[0]) {
            case "plain":
                director = new Director(new TextBuilder());
                result = director.construct();
                System.out.println(result);
                break;
            case "html":
                director = new Director(new HTMLBuilder());
                result = director.construct();
                System.out.println("已产生HTML文件: " + result + "。");
                break;
            case "markdown":
                director = new Director(new MarkdownBuilder());
                result = director.construct();
                System.out.println("Markdown content:\n" + result);
                break;
            default:
                usage();
                System.exit( status: 0);
                break;
        }
    }
```

修改运行配置以测试新增的 Builder 类。



编辑运行配置: 'Main'

名称(N): Main

存储为项目文件(S)

构建并运行

修改选项(M) Alt+M

JRE(J) Alt+J

主要类(C): Alt+C

java 17 'test' 模块的 SI

Main

程序实参(R) Alt+R

markdown

按 Alt 获取字段提示

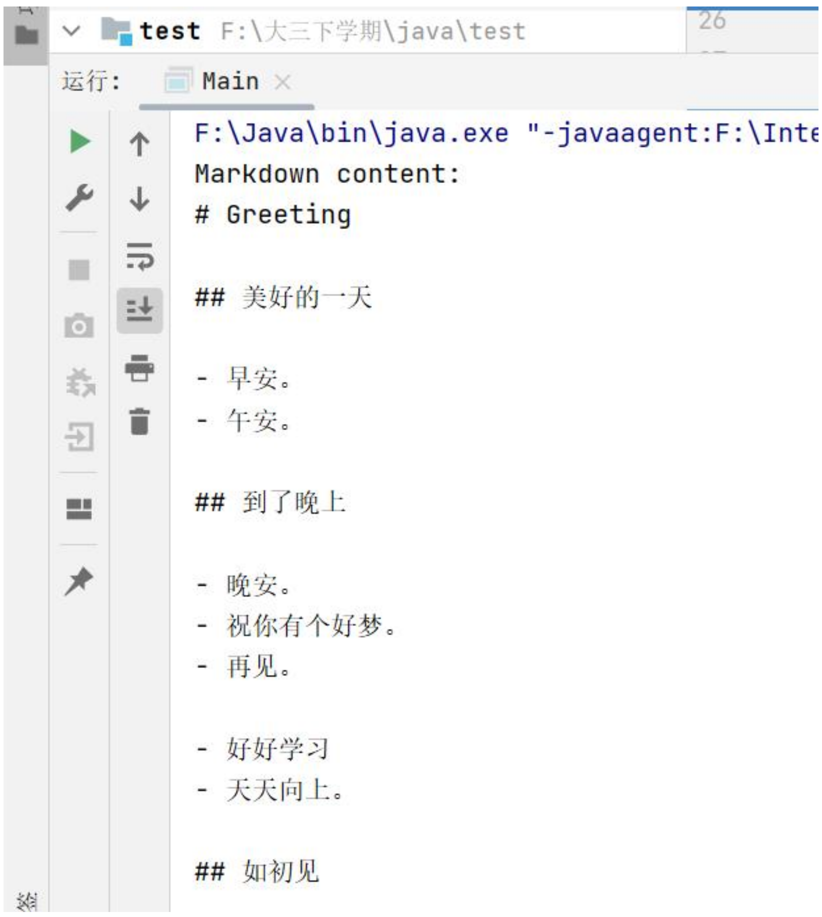
工作目录(W): F:\大三下学期\java\test

环境变量(E):

用分号分隔变量: VAR=value; VAR1=value1

启动时打开运行/调试工具窗口

查看运行结果。



```
F:\Java\bin\java.exe "-javaagent:F:\Inte
Markdown content:
# Greeting

## 美好的一天

- 早安。
- 午安。

## 到了晚上

- 晚安。
- 祝你有个好梦。
- 再见。

- 好好学习
- 天天向上。

## 如初见
```

2、 附录

1) Builder

```
public abstract class Builder {
    public Builder() {
    }

    public abstract void makeTitle(String var1);

    public abstract void makeString(String var1);

    public abstract void makeItems(String[] var1);

    public abstract Object getResult();
}
```

2) HTMLBuilder

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
```

```java
public class HTMLBuilder extends Builder {
    private String filename;
    private PrintWriter writer;

    public HTMLBuilder() {
    }

    public void makeTitle(String title) {
        this.filename = title + ".html";

        try {
            this.writer = new PrintWriter(new
FileWriter(this.filename));
        } catch (IOException var3) {
            IOException e = var3;
            e.printStackTrace();
        }

        this.writer.println("<html><head><title>" + title +
"</title></head><body>");
        this.writer.println("<h1>" + title + "</h1>");
    }

    public void makeString(String str) {
        this.writer.println("<p>" + str + "</p>");
    }

    public void makeItems(String[] items) {
        this.writer.println("<ul>");

        for(int i = 0; i < items.length; ++i) {
            this.writer.println("<li>" + items[i] + "</li>");
        }

        this.writer.println("</ul>");
    }

    public Object getResult() {
        this.writer.println("</body></html>");
        this.writer.close();
        return this.filename;
    }
}
```

3) MarkdownBuilder

```
public class MarkdownBuilder extends Builder {
    private StringBuffer buffer = new StringBuffer();
    public MarkdownBuilder() {
    }
    @Override
    public void makeTitle(String title) {
        buffer.append("# " + title + "\n\n");
    }
    @Override
    public void makeString(String str) {
        buffer.append("## " + str + "\n\n");
    }
    @Override
    public void makeItems(String[] items) {
        for (String item : items) {
            buffer.append("- " + item + "\n");
        }
        buffer.append("\n");
    }
    @Override
    public Object getResult() {
        return buffer.toString();
    }
}
```

4) TextBuilder

```
public class TextBuilder extends Builder {
    private StringBuffer buffer = new StringBuffer();

    public TextBuilder() {
    }

    public void makeTitle(String title) {
        this.buffer.append("==============================\n");
        this.buffer.append("『" + title + "』\n");
        this.buffer.append("\n");
    }

    public void makeString(String str) {
        this.buffer.append( str + "\n");
        this.buffer.append("\n");
    }
    public void makeItems(String[] items) {
        for(int i = 0; i < items.length; ++i) {
            this.buffer.append("#" + items[i] + "\n");
        }

        this.buffer.append("\n");
    }

    public Object getResult() {
        this.buffer.append("==============================\n");
        return this.buffer.toString();
    }
}
```

5) Director

```
public class Director {
    private Builder builder;

    public Director(Builder builder) {
        this.builder = builder;
    }

    public Object construct() {
        this.builder.makeTitle("Greeting");
        this.builder.makeString("美好的一天");
        this.builder.makeItems(new String[]{"早安。", "午安。"});
        this.builder.makeString("到了晚上");
        this.builder.makeItems(new String[]{"晚安。", "祝你有个好梦。",
"再见。"});
        this.builder.makeItems(new String[]{"好好学习", "天天向上。"});
        this.builder.makeString("如初见");
        return this.builder.getResult();
    }
}
```

6) Main

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

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

        Director director;
        Object result;
        switch (args[0]) {
            case "plain":
                director = new Director(new TextBuilder());
                result = director.construct();
                System.out.println(result);
                break;
            case "html":
                director = new Director(new HTMLBuilder());
                result = director.construct();
                System.out.println("已产生 HTML 文件：" + result + "。
");
                break;
            case "markdown":
                director = new Director(new MarkdownBuilder());
                result = director.construct();
                System.out.println("Markdown content:\n" + result);
                break;
            default:
                usage();
                System.exit(0);
                break;
        }
    }

    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");
    }
}
```