STORY 2 定义并解析多个 FLAGS

在上一个 Story 中，我们实现了对单个 Flags 的定义和解析。那么今天我们要实现对多个 Flags 的定义和解析。

AC 1 定义 Flags

我们可以通过多次调用 AddFlagOption 方法来添加支持的 Flags。例如

```
ArgsParser parser = new ArgsParserBuilder()
    .AddFlagOption("flag-a", 'a', "description")
    .AddFlagOption("flag-b", 'b', "description")
    ...
```

但是我们需要详细的定义这个方法的细节。假定我们的方法签名为：

```
ArgsParserBuilder ArgsParserBuilder.AddFlagOption(
    string fullForm,
    char? abbreviationForm,
    string description);
```

该方法每一次都将在创建的解析器中添加一个新的 Flag 定义。其参数的意义如下：

- `fullForm`：该参数代表了这个 flag 的全称。如果当前的 flag 定义并不包含全称，则请传递 `null`。
- `abbreviationForm`：该参数代表了这个 flag 的简称。如果当前 flag 定义并不包含简称，则请传递 `null`。
- `description`：该参数代表了这个 flag 的简短描述。如果不包含描述信息，则请传递 `null`。

每一个 Flag 定义至少应当包含全称或者简称。因此如果 `fullForm` 和 `abbreviationForm` 均为 `null`，则抛出 `ArgumentException`。

Flag 的 Full Form 只能够由字母数字下划线和短划线构成。且第一个字符不能为短划线。而 Abbreviation Form 只能是英文字母。如果参数不符合该规定则抛出 `ArgumentException`。

每一个 Flag 的定义的关键组成部分就是他们的 Full Form 与 Abbreviation Form。并且它可以只包含其中一种形式。因此我们在添加 Flag 定义的时候要小心的避免二义性。我们定义，如果两个 Flag 定义中，Full Form 和 Abbreviation Form 只要有一个重复（注意，大小写是不敏感的）的话就算是冲突的。

因此如果当前添加的 Flag 和之前定义的相冲突，则会抛出 `ArgumentException`。

AC 2 解析 Flags

我们使用如下的语句解析并获得 Flag 的值：

```
ArgsParser parser = builder.Build();

ArgsParsingResult result = parser.Parse(new [] { "--no-edit" "--
amend" });

bool isNoEdit = result.GetFlagValue("--no-edit");
bool isAmended = result.GetFlagValue("--amend");
```

需要注意的是，Flag 的定义并没有规定 Flag 必须出现的位置。因此 `--amend --no-edit` 和 `--no-edit --amend` 是等价的。

首先我们定义 `ArgsParsingResult ArgsParser.Parse(string [] args)` 方法：

参数定义如下：

- `args`：代表需要被解析的字符串数组。一般来说我们会将应用程序启动时的命令行参数直接传递给该方法。

我们不允许 `args` 为 `null`，否则将会抛出 `ArgumentNullException`。我们也不允许 `args` 中有任何的为 `null` 的元素。这在实际的应用程序参数中也是不可能的，如果出现这种情况则应当抛出 `ArgumentException`。

在解析过程中的任何解析错误都将随 `ArgsParsingResult` 返回。

目前已知的 `ArgsParsingResult` 的错误代码如下：

`FreeValueNotSupported`：当前的参数表中含有未定义的或者不能识别的元素。例如，

```
parser.Parse(new [] { "--no-edit", "what_is_this" });
```

```
parser.Parse(new [] { "--not-defined-flags" });
parser.Parse(new [] { "what_is_this" });
parser.Parse(new [] { "-rf" });
```

`DuplicateFlagsInArgs`：当前的参数表中含有重复的 `flag` 元素。例如：

```
parser.Parse(new [] { "-f", "-f" });
parser.Parse(new [] { "-f", "--flag" }); // 假定 --flag 是 -f 的 Full Form
```

AC 3 获得 Flags 的值

我们将详细定义 `bool ArgsParsingResult.GetFlagValue(string flag)` 函数。该函数的参数表如下：

- `flag`：flag 的 Full Form 或者 Abbreviation Form（包含前导的短划线）。

该函数的返回值是相应的 `flag` 的值。由于 `flag` 是一个布尔变量，因此如果它没有出现在参数表中，则其值为 `false`，否则为 `true`。

我们不支持 `flag` 参数为 `null`。将抛出 `ArgumentNullException`。

由于只有在 `ArgsParsingResult.IsSuccess` 为 `true` 的情况下调用这个方法才有意义，因此若其值为 `false`。调用该方法会抛出 `InvalidOperationException`。

如果用户输入的 `flag` 没有定义或者不符合命名规则，则抛出 `ArgumentException`。