Part2 核心流程设计分析

组件设计

Javassist 的核心流程由各个组件共同参与组成，并最终完成对字节码的操作。查阅核心代码和文件目录后发现以下几个类提供核心操作，它们的功能如下：

- `CtField.java`：它可以用来创建新的字段或者修改现有字段的属性，如类型等，并可以将字段添加到 `CtClass` 中。
- `CtMethod.java`：它允许使用者创建新的方法或者修改现有方法的返回类型、访问修饰符以及方法体内容。
- `CtNewMethod.java`：它提供了一些静态方法来快速创建新的方法，如生成 `getter` 和 `setter` 方法，或者根据给定的源代码创建新的方法。
- `CtConstructor.java`：它可以用来创建新的构造函数或者修改现有构造函数的参数和方法体。

获取类文件

Javassist 需要从字节码的层面编辑类文件，并且这个类可能正在运行，即存在内存中，因此需要将类抽象为一个对象，这一过程由 `ClassFile` 类提供。而在核心模块中，类文件可能来自输入流，也可能由 `CtClass` 提供。分析后发现 `ClassFile` 类与 `ConstInfo`、`FieldInfo`、`MethodInfo`、`AttributeInfo` 相互存在依赖关系。在输入流提供类文件时，`ClassFile` 创建抽象的类文件对象并初始化生成各信息供上面四个类使用；反之，若 `CtClass` 提供类文件，则由这四个类的 `get` 方法来构建抽象对象。

```
ClassPool pool = ClassPool.getDefault();
```

编辑类信息

该过程通过 `CtField`、`CtNewMethod`、`CtConstructor` 完成，先由 `CtField` 添加新字段：

```
CtField param = new CtField(pool.get("java.lang.String"), "name", c);
c.addField(param, CtField.Initializer.constant("con"));
```

然后由 `CtNewMethod` 创建新方法 `getter` 和 `setter`：

```
c.addMethod(CtNewMethod.setter("setName", param));
c.addMethod(CtNewMethod.getter("getName", param));
```

再由 `CtConstructor` 添加新的构造函数：

```
CtConstructor cons = new CtConstructor(new CtClass[]
{pool.get("java.lang.String")}, c);
c.addConstructor(cons);
```

修改方法体

该过程通过 `CtMethod` 完成，使用 `CtMethod` 获取后直接进行修改：

```
CtMethod method = c.getDeclaredMethod("myMethod");
method.setBody("{ System.out.println(\"Modified Method\"); }");
```

保存修改

将修改后的类写入文件系统：

```
c.writeFile();
```

总结

Javassist 的核心流程通过 `ClassPool`、`CtClass`、`CtField`、`CtMethod`、`ClassFile` 等若干个关键组件实现，它们分别提供不同的功能又相互依存，共同构成了 Javassist 动态字节码操作的基础架构。