Design Document

1. Overview

这是一个 **命令行个人所得税计算器**，支持：

- 计算个人所得税
- 修改起征点
- 更新税率表
- 交互式 CLI 操作
- 重置默认配置 系统采用面向对象的设计思想，将功能模块化，便于扩展和维护。

运行环境

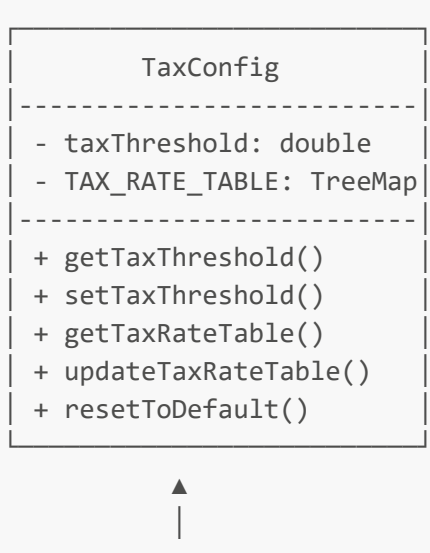
- **JDK:** 17+
- **JUnit:** 5.8.1 (用于单元测试)
- **开发工具:** 任意IDE (本项目在Eclipse上实现)

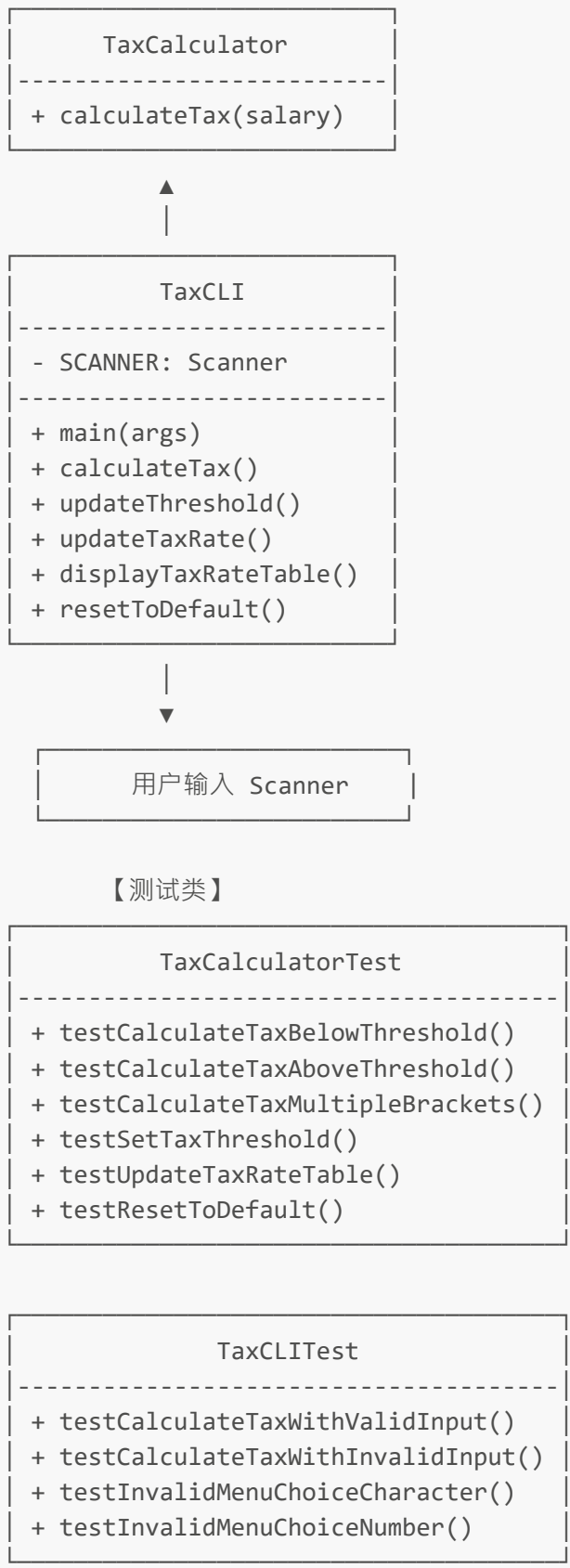
2. 模块设计

2.1 主要组件

类名	职责
`TaxCalculator`	计算应缴税款，根据税率表逐级计算税额
`TaxConfig`	维护**起征点** 及 **税率表**，支持修改和重置
`TaxCLI`	提供**命令行交互界面**，处理用户输入和操作
`TaxCalculatorTest`	单元测试 `TaxCalculator` 计算逻辑
`TaxCLITest`	测试 CLI 交互逻辑，模拟用户输入

2.2 类关系图





2.3 数据流

1. 用户通过 `TaxCLI`输入操作。
2. `TaxCLI`调用 `TaxConfig`获取或修改配置。
3. `TaxCLI`调用 `TaxCalculator`计算税款。

4. 计算结果通过 `TaxCLI`输出给用户。

3. 测试方案

3.1 测试类型

类型	功能
单元测试	测试 `TaxCalculator`计算是否正确
配置测试	测试 `TaxConfig`设置是否正确
边界测试	确保税率边界值计算正确
异常测试	处理错误输入，防止崩溃
CLI 测试	测试交互逻辑，模拟用户输入

3.2 测试工具

- JUnit 5：用于编写和运行单元测试。
- Assertions：用于验证测试结果是否符合预期。

3.3 测试用例设计

测试用例名称	描述	输入数据	预期输出
testCalculateTaxBelowThreshold	测试工资低于起征点的情况	工资 = 3000	税款 = 0
testCalculateTaxAboveThreshold	测试工资高于起征点的情况	工资 = 8000	税款 = 90
testCalculateTax40000	测试高收入的税款计算	工资 = 40000	税款 = 5390
testCalculateTaxEdge8000	测试边界值（8000 元）的税款计算	工资 = 8000	税款 = 90
testCalculateTax_NegativeSalary	测试负工资的情况	工资 = -5000	税款 = 0
testCalculateTax_MaxDoubleValue	测试极大值的情况	工资 = Double.MAX_VALUE	税款 > 0
testSetTaxThreshold	测试设置新的起征点	新起征点 = 6000	起征点 = 6000
testUpdateTaxRateTable	测试更新税率表	收入 = 5000, 税率 = 0.05	税率表更新成功
testResetToDefault	测试重置为默认配置	无	起征点和税率表恢复为默认值
testCalculateTaxWithValidInput	测试正常输入下的税款计算	1\n8000\n6\n	输出包含 "应缴个人所得税: 90.0 元"

测试用例名称	描述	输入数据	预期输出
testCalculateTaxWithInvalidInput	测试非数字输入的处理	1\n*\n6\n	输出包含 "无效输入，请输入数字"
testModifyTaxThreshold	测试修改起征点的功能	2\n6000\n6\n	输出包含 "起征点已更新为: 6000"
testModifyTaxRate	测试修改税率表的功能	3\n5000\n0.05\n6\n	输出包含 "税率已更新！"
testInvalidMenuChoiceNumber	测试无效菜单选项（数字）的处理	7\n6\n	输出包含 "请输入 1-6 之间的选项"
testInvalidMenuChoiceCharacter	测试无效菜单选项（字符）的处理	a\n6\n	输出包含 "请输入有效的数字"

3.4 回归测试

自动化回归测试：编写 `test.bat`批处理脚本：

```
@echo off
echo ===== Regression Test Lauching =====

:: 1. 编译代码
call compile.bat
if %ERRORLEVEL% NEQ 0 (
    echo Compilation FAILED
    exit /b 1
)

:: 2. 运行 JUnit 测试
echo >> Unit Testing...
java -cp "bin;lib/*" org.junit.platform.console.ConsoleLauncher ^
    --select-class TaxCalculatorSystem.TaxCalculatorTest ^
    --select-class TaxCalculatorSystem.TaxCLITest
if %ERRORLEVEL% NEQ 0 (
    echo Some tests FAILED!
    exit /b 1
)

echo All PASSED!
exit /b 0
```

3.5 测试覆盖率

覆盖了：

- 核心功能：税款计算、起征点修改、税率表更新。
- 边界情况：工资为 0、负工资、极大值等情况。

- 异常处理：非数字输入、无效菜单选项等情况。

3.6 测试结果

所有测试用例均通过，系统功能符合预期。

```
? Thanks for using JUnit! Support its development at https://junit.org/sponsoring

.
+-- JUnit Jupiter [OK]
| +-- TaxCalculatorTest [OK]
| | +-- testResetToDefault() [OK]
| | +-- testCalculateTaxAboveThreshold() [OK]
| | +-- testCalculateTax_MaxDoubleValue() [OK]
| | +-- testSetTaxThreshold() [OK]
| | +-- testCalculateTax15000() [OK]
| | +-- testCalculateTax40000() [OK]
| | +-- testCalculateTaxEdge8000() [OK]
| | +-- testCalculateTax_NegativeSalary() [OK]
| | +-- testUpdateTaxRateTable() [OK]
| | '-- testCalculateTaxBelowThreshold() [OK]
| '-- TaxCLITest [OK]
|   +-- testModifyTaxThreshold() [OK]
|   +-- testInvalidMenuChoiceCharacter() [OK]
|   +-- testCalculateTaxWithValidInput() [OK]
|   +-- testCalculateTaxWithInvalidInput() [OK]
|   +-- testInvalidMenuChoiceNumber() [OK]
|   '-- testModifyTaxRate() [OK]
+-- JUnit Vintage [OK]
'-- JUnit Platform Suite [OK]

Test run finished after 80 ms
[         5 containers found      ]
[         0 containers skipped    ]
[         5 containers started    ]
[         0 containers aborted    ]
[         5 containers successful ]
[         0 containers failed     ]
[        16 tests found           ]
[         0 tests skipped         ]
[        16 tests started         ]
[         0 tests aborted         ]
[        16 tests successful      ]
[         0 tests failed          ]


WARNING: Delegated to the 'execute' command.
```