# 测试计划

[测试计划](#测试计划)  
[GUI模块](#gui模块)  
 [1. 引言](#X9e0e48bd65310e5622b496d9e969f57a0000021)  
 [2. 测试目标](#Xbdea01da7f44006dd5b400cc6b299fe0f74f5a5)  
 [3. 测试范围](#Xe91ab2bca0a5781cb4c42e97a96bf2687dd66cf)  
 [4. 测试策略](#Xb01153f11571daf6cce7f3d5b065d0e4e0d40d6)  
 [4.1 功能测试](#X97e95ce5120260ae45e297c715f7df3f99cea43)  
 [4.2 性能测试](#X01881f6bcbccee76a5a806af3bf5ba997872735)  
 [4.3 兼容性测试](#Xc9b2f39b1b28faf73c536a995996826f93e9d6f)  
[数据IO测试](#数据io测试)  
 [1. 引言](#X3c91d5c57be95a5bc2f46dd7e555d2660eb5bc1)  
 [2. 测试目标](#X42484e81cd715fd3a0176cf594c725e7b63ccf5)  
 [3. 测试范围](#X80a0c451ac1dd0b54453b85efb885cd91acc9df)  
 [4. 测试策略](#Xe5500c73b523b267ffb7b1d0bb8c8a117467333)  
 [4.1 黑盒测试](#X1d21086577288cd8c53303d25017fd13d207cfe)  
 [4.2 白盒测试](#X7b6095e66bbcaa177c7777b5340fbda5667fde6)  
 [5. 测试用例设计](#Xc36693a07bb62adb92c925fd28385df99807575)  
 [用例1：文本框输入合法性测试](#用例1文本框输入合法性测试)  
 [用例2：数值输入范围测试](#用例2数值输入范围测试)  
 [用例3：必填项校验测试](#用例3必填项校验测试)  
[算法功能性测试](#算法功能性测试)  
 [1. 引言](#X4de72aa4476bf57514c1eba40b9db07182045dc)  
 [2. 测试目标](#Xcb9c3982e5e7f92abaed08a30d6dfb856191275)  
 [3. 测试范围](#X186eaae8b38fe4ccfe7b115dbe94fb213156edd)  
 [4. 测试策略](#X5c0e996f527636ad37a93bae94115246c0e6ddc)  
 [4.1 对拍测试](#X558bf791e3b2083bf6f8d84a06249aee9fafda2)  
 [4.2 暴力测试](#Xe71c5680593fac16639f18415d4585a5d6fd442)  
 [5. 测试用例设计](#X29a9768acb158d80039b2294cff47316aba27ae)  
 [用例1：排序算法对拍测试](#用例1排序算法对拍测试)  
 [用例2：查找算法暴力测试](#用例2查找算法暴力测试)  
 [用例3：计算逻辑边界测试](#用例3计算逻辑边界测试)  
 [6. 测试环境](#Xcec56cfd1a3916079c488574f1c217c97196652)  
[测试环境](#测试环境)

# GUI模块

## 1. 引言

本测试计划的目的是确保JAVAGUI实现的多窗口程序的功能性、稳定性和用户体验达到预期标准。测试将涵盖不同窗口之间的交互、窗口的打开和关闭、窗口内容的正确显示、响应速度及错误处理等方面。

### 2. 测试目标

* 验证多窗口程序的各项功能是否正确实现。
* 确保窗口之间的交互逻辑正确且无异常。
* 测试窗口在不同操作系统和屏幕分辨率下的兼容性。
* 检查程序在异常情况（如网络断开、系统资源不足等）下的表现。
* 评估用户界面的响应速度和易用性。

### Processes（进程表）

| 字段名称 |
| --- |
| process\_id |
| process\_name |
| arrival\_time |
| burst\_time |
| created\_at |
| updated\_at |

### Scheduling（调度表）

| 字段名称 |
| --- |
| scheduling\_id |
| process\_id |
| start\_time |
| end\_time |
| created\_at |
| updated\_at |

## 3. 测试范围

* 主窗口及子窗口的功能测试。
* 窗口之间的数据传递和状态保持。
* 窗口的打开、关闭及切换操作。
* 窗口内容的刷新和显示。
* 窗口事件的处理，包括鼠标和键盘事件。
* 错误处理及异常情况下的恢复能力。

## 4. 测试策略

### 4.1 功能测试

* **窗口打开和关闭**：测试所有窗口能否正常打开和关闭，检查是否存在资源泄漏。
* **窗口切换**：测试窗口切换的流畅度和正确性，确保前后台切换时状态保持一致。
* **窗口内容**：检查每个窗口的内容是否正确加载和显示，包括文本、图像和其他控件。
* **事件处理**：测试各窗口的事件响应能力，包括按钮点击、菜单选择、文本输入等。
* **数据传递**：验证窗口之间的数据传递是否准确无误。

### 4.2 性能测试

* **响应速度**：测试各窗口的响应速度，包括打开、关闭和切换的时间。
* **资源使用**：监控程序的内存和CPU使用情况，确保资源使用在合理范围内。

### 4.3 兼容性测试

* **操作系统**：在不同操作系统（如Windows, macOS, Linux）上进行测试，确保程序的跨平台兼容性。
* **分辨率**：在不同屏幕分辨率下进行测试，确保窗口布局和内容显示正常。

# 数据IO测试

## 1. 引言

本测试计划旨在确保JAVAGUI实现的多窗口程序中的输入输出（I/O）数据合法性，包括用户输入和输出的处理。通过结合黑盒测试和白盒测试的方法，确保程序在处理用户输入数据时的正确性和稳定性。

## 2. 测试目标

* 验证用户输入的数据合法性。
* 确保程序对非法输入的处理正确。
* 测试不同类型用户输入的数据处理能力。
* 检查程序输出的数据是否符合预期。

## 3. 测试范围

* 用户输入：包括文本框、按钮、菜单选择等用户交互输入。
* 数据合法性检查：包括数值范围、格式校验、必填项等。
* 程序输出：包括窗口中显示的数据和反馈信息。

## 4. 测试策略

### 4.1 黑盒测试

* 等价类划分
* ：将输入数据划分为若干等价类，对每个等价类进行测试，以确保覆盖不同类型的输入数据。
  + 有效等价类：输入数据符合要求的情况。
  + 无效等价类：输入数据不符合要求的情况。
* 边界值分析
* ：针对输入数据的边界值进行测试，确保程序在边界条件下的正确性。
  + 例如：输入数值的最小值、最大值及其邻近值。
* **错误猜测**：基于经验和直觉，对可能出现错误的输入数据进行测试。

### 4.2 白盒测试

* **语句覆盖**：确保所有代码语句至少执行一次。
* **分支覆盖**：确保所有代码分支（if, else等）至少执行一次。
* **路径覆盖**：确保所有可能的代码路径均被执行，重点测试复杂逻辑的路径。

## 5. 测试用例设计

### 用例1：文本框输入合法性测试

* **测试目标**：验证文本框中输入数据的合法性。
* **前置条件**：程序已启动并显示文本框。
* 测试步骤：
  1. 输入有效文本数据（如："Hello World"）。
  2. 输入无效文本数据（如：空字符串、超长字符串、特殊字符）。
* 预期结果：
  + 对于有效文本数据，程序应正常处理并显示。
* 对于无效文本数据，程序应提示错误信息或拒绝输入。

### 用例2：数值输入范围测试

* **测试目标**：验证数值输入的数据范围。
* **前置条件**：程序已启动并显示数值输入框。
* 测试步骤：
  1. 输入最小边界值及其邻近值。
  2. 输入最大边界值及其邻近值。
  3. 输入超出范围的值（如：负数、过大数值）。
* 预期结果：
  + 对于边界值，程序应正常处理并显示。
* 对于超出范围的值，程序应提示错误信息或拒绝输入。

### 用例3：必填项校验测试

* **测试目标**：验证必填项的输入合法性。
* **前置条件**：程序已启动并显示必填项。
* 测试步骤：
  1. 尝试提交空的必填项。
  2. 输入合法数据后提交。
* 预期结果：
  + 对于空的必填项，程序应提示错误信息并阻止提交。
  + 对于合法数据，程序应正常提交并处理。

# 算法功能性测试

## 1. 引言

本测试计划旨在确保JAVAGUI实现的多窗口程序中的各项算法功能正常，包括排序算法、查找算法、计算逻辑等。通过对拍测试和暴力测试的方法，验证算法的正确性和效率。

## 2. 测试目标

* 验证程序中实现的各类算法功能的正确性。
* 确保算法在不同输入条件下的稳定性和性能。
* 通过对拍测试和暴力测试的方法，检测潜在的算法错误。

## 3. 测试范围

* 各类排序算法的正确性和性能测试。
* 查找算法的正确性和边界条件测试。
* 计算逻辑（如数学公式、数据处理等）的正确性验证。

## 4. 测试策略

### 4.1 对拍测试

对拍测试是一种通过生成大量随机输入数据，并分别使用待测算法和已知正确的参考算法进行对比的方法。通过比对两者输出的结果，检测待测算法的正确性。

### 4.2 暴力测试

暴力测试是指通过枚举所有可能的输入，或者使用简易但低效的算法计算出正确结果，然后将其与待测算法的结果进行对比，验证算法的正确性。

## 5. 测试用例设计

### 用例1：排序算法对拍测试

* **测试目标**：验证排序算法的正确性。
* **前置条件**：待测排序算法和参考排序算法已实现。
* 测试步骤
* ：
  1. 生成随机数组。
  2. 使用待测排序算法对数组排序。
  3. 使用参考排序算法对同一数组排序。
  4. 比较两个排序结果。
* **预期结果**：两种排序结果应完全相同。

### 用例2：查找算法暴力测试

* **测试目标**：验证查找算法的正确性。
* **前置条件**：待测查找算法和暴力查找算法已实现。
* 测试步骤
* ：
  1. 生成随机数组和随机目标值。
  2. 使用待测查找算法查找目标值在数组中的位置。
  3. 使用暴力查找算法查找目标值在数组中的位置。
  4. 比较两个查找结果。
* **预期结果**：两种查找结果应完全相同。

### 用例3：计算逻辑边界测试

* **测试目标**：验证计算逻辑在边界条件下的正确性。
* **前置条件**：待测计算逻辑已实现。
* 测试步骤
* ：
  1. 输入边界值（如最小值、最大值）。
  2. 计算期望的正确结果。
  3. 使用待测计算逻辑计算结果。
  4. 比较计算结果与期望结果。
* **预期结果**：计算结果应与期望结果一致。

## 6. 测试环境

# 测试环境

* 操作系统：Windows 10
* Java版本：Java SE 8及以上
* 测试工具：Eclipse或IntelliJ IDEA（用于代码调试），JIRA（用于缺陷管理）