Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

**Отчет**

**К лабораторной работе по дисциплине:**

|  |
| --- |
| «Технологии разработки программных комплексов» |
| «Тестирование» |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-19-1 |  |  |  | Путинцев О.Р. |
|  | шифр группы |  | подпись |  | Ф.И.О. |
| Проверил: |  |  |  |  | Бахвалова З.А. |
|  | должность |  | подпись |  | Ф.И.О. |

г. Иркутск 2022 г.

**Оглавление**

[1 Постановка задачи 3](#_Toc102698012)

[2 Требования 3](#_Toc102698013)

[2.1 Функциональные требования 3](#_Toc102698014)

[2.2 Требования к входным и выходным данным 4](#_Toc102698015)

[2.3 Нефункциональные требования 4](#_Toc102698016)

[2.4 Требования к пользовательскому интерфейсу 4](#_Toc102698017)

[3 Структурный контроль 4](#_Toc102698018)

[4 Стратегия «белого ящика» 7](#_Toc102698019)

[5 Стратегия «чёрного ящика» 16](#_Toc102698020)

[5. 1 Эквивалентное разбиение 16](#_Toc102698021)

[5. 2 Анализ граничных условий 18](#_Toc102698022)

[5. 3 Предупреждение об ошибке 18](#_Toc102698023)

[6 Оценочное тестирование 19](#_Toc102698024)

[7 Тестирование интерфейса 20](#_Toc102698025)

[8 Отчет по тестированию программы 22](#_Toc102698026)

[9 Листинг 24](#_Toc102698027)

[Класс Main 24](#_Toc102698028)

[Класс Controller 24](#_Toc102698029)

1 Постановка задачи

Дан правильный пятиугольник, в который вписана 5-конечная звезда. Известен радиус звезды и пятиугольника. Требуется рассчитать площадь закрашенной области (рис. 1).

Дан прямоугольник, в который вписаны окружность и четыре четверти окружностей в углах этой фигуры. У четвертей окружности и окружности в центре разные радиусы. Известны радиусы и длины сторон прямоугольника. Необходимо найти площадь не закрашенной фигуры

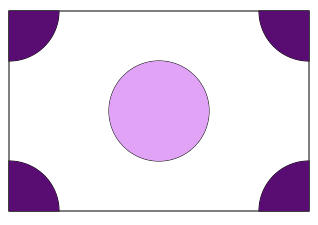


Рисунок 1 – Заданная фигура.

2 Требования

Требуется создать приложения на языке высокого уровня Java, которое будет рассчитывать площадь закрашенной области.

2.1 Функциональные требования

1. Программа должна строить геометрические фигуры и выводить их на экран. Для построения фигуры пользователю необходимо ввести длины сторон прямоугольника и радиусы окружности и четвертей окружности. Фигура будет располагаться по центру холста и соответствовать введенным размерам. Для построения прямоугольника и окружностей дается их радиус. Чтобы окружность была вписана в прямоугольник её радиус не должен быть больше радиуса прямоугольника.

2. Для расчета площади не закрашенной фигуры сначала рассчитывается площадь прямоугольника затем рассчитывается площадь окружностей, далее из площади прямоугольника вычитается сумма площадей окружностей.

3. Для изменения размеров построенных фигур в пределах окна необходимо выбрать масштаб фигуры.

4. Размеры окна с фигурами должно быть фиксированным, без возможности изменения.

5. При изменении размеров фигура будет автоматически перерисовываться.

6. Площадь должна рассчитываться автоматически и изменятся от веденных параметров.

2.2 Требования к входным и выходным данным

Входные данные:

* Радиус окружности.
* Радиус четвертей окружности.
* Ширина прямоугольника.
* Высота прямоугольника.

Выходные данные:

* Рисунок;
* Площадь заштрихованной фигуры;
* Площадь не может быть отрицательной.

2.3 Нефункциональные требования

Программа должна быть реализована на платформе Java с использованием графической библиотеки JavaFX.

2.4 Требования к пользовательскому интерфейсу

Программа представляет собой окно, состоящее из 7 элементов: область для построения фигур, двух полей для ввода, трех ползунков для изменения радиусов и масштаба и текстового элемента для вывода площади фигуры.

Результат подсчитанной площади будет выводиться под текстовыми полями для ввода.

3 Структурный контроль

Таблица 1 – Тестирование программы с использованием структурного контроля.

|  |  |  |  |
| --- | --- | --- | --- |
| № вопроса | Строки, подлежащие проверке | Результат проверки | Вывод |
| 1.1 Все ли переменные инициализированы? | 21-42, 47-48, 121-124, 214-215, 249-296, 300-303 | Все переменные инициализированы | Все переменные инициализированы |
| 1.2 Не превышены ли максимальные (или реальные) размеры массивов и строк? | 88-89, 103-104, 119-120, 136-137, 142-143 | Максимально допустимые размеры элементов превышают тип double | Размеры переменных превышаются |
| 1.3 Не перепутаны ли строки со столбцами при работе с матрицами? | - | - | Матриц в программе нет |
| 1.4 Присутствуют ли переменные со сходными именами? | 120-200, 277-295,297-305 | rS – rB, x1 – x4, y1-y4, | Все переменные имеют разные имена |
| 1.5 Используются ли файлы?  Если да, то:  При вводе из файла проверяется ли завершение файла? Соответствуют ли типы записываемых и читаемых значений? | - | - | Файлы не используются |
| 1.6 Использованы ли не типизированные переменные, открытые массивы, динамическая память? Если да, то Соответствуют ли типы переменных при "наложении" формата? Не выходят ли индексы за границы массивов? | - | - | Не используются |
| 2.1 Правильно ли записаны выражения (порядок следования операторов)? | 1-304 | Компилятор не пропускает программу, если в коде присутствуют ошибки | Выражения записаны правильно, порядок следования операторов верен |
| 2.2 Корректно ли выполнены вычисления с переменными различных типов (в том числе с использованием целочисленной арифметики)? | 47-57, 120-132, 109-111, 261,270, 304 | Для задания переменных используются значения, введённые пользователем, либо инициализируются объекты классов графических компонент | Все вычисления выполняются корректно |
| 2.3 Возможно ли переполнение разрядной сетки или ситуация машинного нуля? | - | - | Не возможно |
| 2.4 Соответствуют ли вычисления заданным требованиям точности? | - | - | Вычисления соответствуют заданным требованиям |
| 2.5 Присутствуют ли сравнения переменных различных типов? | - | - | Сравнения отсутствуют |
| 3.1 Будут ли корректно завершены циклы? | - | - | Циклы используются корректно |
| 3.2 Будет ли завершена программа? | - | - | Программа будет завершена |
| 3.3 Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа? Корректно ли продолжатся вычисления? | - | - | Циклов, которые не будут выполняться из-за некорректных условий входа в программе нет |
| 3.4 Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации "элемент найден" и "элемент не найден"? | - | - | Циклов поиска в программе нет |
| 4.1 Соответствуют ли списки параметров и аргументов по порядку, типу, единицам измерения? | - | Программа принимает данные, водимые пользователем в сантиметрах, а выводимые данные конвертирует из пикселей в сантиметры | Данные представлены в нужных единицах измерения и требуемой точностью |
| 4.2 Не изменяет ли подпрограмма аргументов, которые не должны изменяться? | - | - | Программа не изменяет значения, которые не должны изменяться |
| 4.3 Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами? | - | - | Нарушения области действия переменных не происходит |

4 Стратегия «белого ящика»

Стратегия тестирования по принципу «белого ящика», или стратегия тестирования, управляемая логикой программы (с учетом алгоритма), позволяет проверить логику и внутреннюю структуру программы.

**4.1 Покрытие операторов**

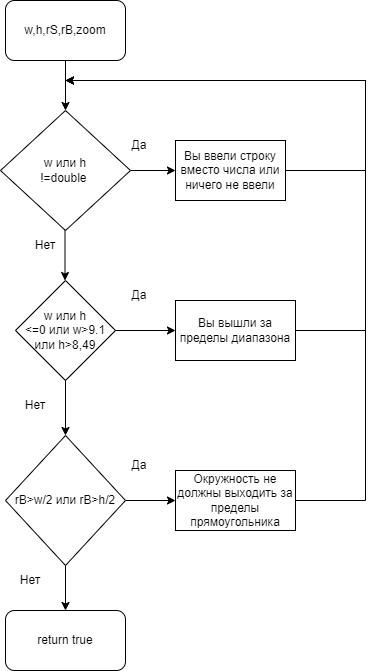


Рисунок 2 – Схема алгоритма обработки входящих данных

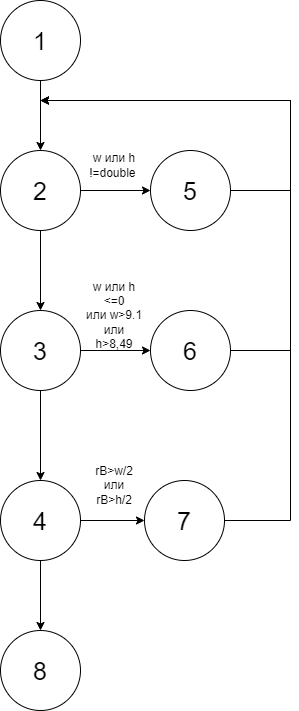


Рисунок 3 – Граф алгоритма обработки входящих данных

Таблица 2 – Результат тестирования по принципу покрытие операторов

|  |  |  |  |
| --- | --- | --- | --- |
| Назначение теста | Входные данные | Ожидаемый путь | Результат тестирования |
| Проверка операторов | w=«ab», h=«3» | 1-2-5-1 | 1-2-5-1 |
| Проверка операторов | w=«1», h=«2b» | 1-2-5-1 | 1-2-5-1 |
| Проверка операторов | w=«0», h=«2» | 1-2-3-6-1 | 1-2-3-4-1 |
| Проверка операторов | w=«3», h=«0» | 1-2-3-6-1 | 1-2-3-6-1 |
| Проверка операторов | w=«2», h=«3» | 1-2-3-4-5-6-7-8 | 1-2-3-4-5-6-7-8 |

Вывод: так как данный метод требует написания такого количества тестов, чтобы при выполнении их всех каждый оператор был выполнен хотя бы один раз. Очевидно, что все 5 рассмотренных нами теста, покрывают все операторы, хотя бы один раз. При w=ab и h=2, R1=1 и w=2b, h=0 и w=2, h=3 и w=0 в программе будет выявлена ошибка, а при w =2, h=3.

**4.2 Покрытие решений (переходов)**

Для реализации этого критерия необходимо достаточное число тестов, такое, что каждое решение на этих тестах принимает значение «истина» или «ложь», по крайней мере, один раз.

Таблица 3 - Результат тестирования по принципу покрытие решений

|  |  |  |  |
| --- | --- | --- | --- |
| Назначение теста | Входные данные | Ожидаемый путь | Результат тестирования |
| «Окружность выходит за пределы прямоугольника (радиус больше ширины)» | w=«2», rB=«3», h= «4» | 1-2-3-4-7 | 1-2-3-4-7 |
| «Окружность выходит за пределы прямоугольника (радиус больше высоты)» | w=«4», rB=«3», h= «2» | 1-2-3-4-7 | 1-2-3-4-7 |
| «Введенная ширина прямоугольника не является числом» | w=«ab», h=«2» | 1-2-7-1 | 1-2-5-1 |
| «Введенная высота звезды не является числом» | w=«2», h=«ab» | 1-2-7-1 | 1-2-5-1 |
| «Введенная ширина прямоугольника отрицательная» | w=«-5», h=«4» | 1-2-3-6-1 | 1-2-3-6-1 |
| «Введенная высота прямоугольника отрицательная» | w=«4», h=«-5» | 1-2-3-6-1 | 1-2-3-6-1 |
| «Введенная ширина прямоугольника равен нулю» | w=«0», h=«4» | 1-2-3-6-1 | 1-2-3-6-1 |
| «Введенная высота прямоугольника равна нулю» | w=«5», h=«0» | 1-2-3-6-1 | 1-2-3-6-1 |
| «Корректный ввод параметров» | w=«5», h=«3», rB= «2» | 1-2-3-4-8 | 1-2-3-4-8 |

Вывод: метод покрытия решений требует такого количества тестов, чтобы при выполнении их всех, по каждой траектории, соединяющей соседние элементы блок-схемы, вычисление прошло хотя бы один раз. Это означает, что каждое решение должно принимать как истинные, так и ложные значения. Именно это обеспечивает использование всех путей, выходящих из точек ветвления. Из таблицы видно, что каждое действие принимает либо истину, либо ложь

**4.3 Покрытие условий**

Этот метод требует составить тесты так, чтобы все возможные результаты каждого условия выполнились, по крайней мере, один раз, все результаты каждого решения выполнились, по крайней мере, один раз и каждой точке входа управление передается, по крайней мере, один раз.

Таблица 4 - Результат тестирования по принципу покрытие условий

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение теста | Входные данные | Ожидаемый путь | | Результат тестирования | |
| Проверка условия «Окружность выходит за пределы прямоугольника» (радиус больше ширины)». | w=«2», rB=«3», h= «4» | 1-2-3-4-7 | | 1-2-3-4-7 | |
| Проверка условия «Окружность выходит за пределы прямоугольника (радиус больше высоты).» | w=«4», rB=«3», h= «2» | 1-2-3-4-7 | | 1-2-3-4-7 | |
| Условие «Окружность выходит за пределы прямоугольника (радиус больше ширины)». Проверка решения когда радиус меня ширины и высоты | w=«4», h= «4», rB=«3» | 1-2-3-4-8 | 1-2-5-1 | | |
| Проверка условия «Введенная ширина прямоугольника не является числом» | w=«ab», h=«2» | 1-2-5-1 | | 1-2-5-1 | |
| Проверка условия «Введенная высота звезды не является числом» | w=«2», h=«ab» | 1-2-5-1 | | | 1-2-5-1 |
| Проверка условия «Введенная высота звезды не является числом» | w=«2», h=«ab» | 1-2-7-1 | | 1-2-7-1 | |
| Проверка условия «Введенная ширина прямоугольника отрицательная» | w=«-5», h=«4» | 1-2-3-6-1 | | | 1-2-3-6-1 |
| Проверка условия  «Введенная высота прямоугольника отрицательная» | w=«4», h=«-5» | 1-2-3-6-1 | | | 1-2-3-6-1 |
| Проверка условия «Введенная высота прямоугольника равна нулю» | w=«5», h=«0» | 1-2-3-6-1 | | | 1-2-3-6-1 |
| Проверка условия  «Введенная ширина прямоугольника равна нулю» | w=«0», h=«4» | 1-2-3-6-1 | | | 1-2-3-6-1 |
| Проверка условия «Введенный радиус звезды равен нулю» | R1=«0», R2=«4» | 1-2-3-4-8-1 | | 1-2-3-4-8-1 | |
| Проверка условия «Введенный радиус пятиугольника равен нулю» | R1=«5», R2=«0» | 1-2-3-4-8-1 | | 1-2-3-4-8-1 | |
| Проверка условия «Число меньше или равно нулю» | Zoom=«0» | 1-2-3-4-8-1 | | 1-2-3-4-8-1 | |
| Проверка условия «Введенный зум меньше нуля» | Zoom=«-50» | 1-2-3-4-5-9-1 | | 1-2-3-4-5-9-1 | |
| Проверка условия «Корректный ввод параметров» | w=«5», h=«3», rB= «2» | 1-2-3-4-8 | | | 1-2-3-4-8 |

Вывод: метод покрытия условий состоит в таком подборе тестов, когда каждое условие (элементарное суждения в условных операторах) принимает как истинное, так и ложное значение. В нашем примере имеется четыре условия: w < rB,h<rB, w или h не число, w или h <= нулю. Тест: w=«5», h= «3», rB=«2» покрывает все эти условия.

**4.4 Покрытие решений/условий**

Этот метод требует составить тесты так, чтобы все возможные результаты каждого условия выполнились, по крайней мере, один раз, все результаты каждого решения выполнились, по крайней мере, один раз и каждой точке входа управление передается, по крайней мере, один раз.

Таблица 5 - Результат тестирования по принципу покрытие решений/условий

|  |  |  |  |
| --- | --- | --- | --- |
| Назначение теста | Входные данные | Ожидаемый путь | Результат тестирования |
| Условие «Окружность выходит за пределы прямоугольника (радиус больше ширины)». Проверка решения | w=«2», h= «4», rB=«3» | 1-2-5-1 | 1-2-5-1 |
| Условие «Окружность выходит за пределы прямоугольника (радиус больше высоты)». Проверка решения | w=«4», h= «2», rB=«3» | 1-2-5-1 | 1-2-5-1 |
| Условие «Окружность выходит за пределы прямоугольника (радиус больше ширины)». Проверка решения когда радиус меня ширины и высоты | w=«4», h= «4», rB=«3» | 1-2-5-1 | 1-2-5-1 |
| Условие «Введенная ширина прямоугольника не является числом»  Проверка решения когда введено не число | w=«ab», h=«2» | 1-2-5-1 | 1-2-5-1 |
| Условие «Введенная высота прямоугольника не является числом». Проверка решения когда введено не число | w=«2», h=«ab» | 1-2-5-1 | 1-2-5-1 |
| Условие «Введенная высота прямоугольника не является числом». Проверка решения когда введено число | w=«2», h=«ab» | 1-2-3-4-8 | 1-2-3-4-8 |
| Условие «Введенная ширина прямоугольника не является числом». Проверка решения когда не введено число | w=«ab», h=«2» | 1-2-5-1 | 1-2-5-1 |
| Условие «Введенная ширина прямоугольника не является числом». Проверка решения когда не введено число | w=«ab», h=«2» | 1-2-3-4-8 | 1-2-3-4-8 |
| Условие «Введенная высота прямоугольника отрицательная». Проверка решения когда w < 0 | w=«-5», h=«4» | 1-2-3-6-1 | 1-2-3-6-1 |
| Условие «Введенная высота прямоугольника отрицательная». Проверка решения когда w > 0 | w=«5», h=«4» | 1-2-3-4-8 | 1-2-3-4-8 |
| Условие «Введенная высота прямоугольника отрицательная». Проверка решения когда h < 0 | w=«4», h=«-5» | 1-2-3-6-1 | 1-2-3-6-1 |
| Условие «Введенная высота прямоугольника отрицательная». Проверка решения когда h > 0 | w=«4», h=«2» | 1-2-3-4-8 | 1-2-3-4-8 |
| Условие «Введенная ширина прямоугольника равна нулю». Проверка решения когда w = 0 | w=«0», h=«4» | 1-2-3-6-1 | 1-2-3-6-1 |
| Условие «Введенная ширина прямоугольника равна нулю». Проверка решения когда w > 0 | w=«3», h=«4» | 1-2-3-4-8 | 1-2-3-4-8 |
| Условие «Введенная высота прямоугольника равна нулю». Проверка решения когда h = 0 | w=«5», h=«0» | 1-2-3-6-1 | 1-2-3-6-1 |
| Условие «Введенная высота прямоугольника равна нулю». Проверка решения когда h > 0 | w=«5», h=«3» | 1-2-3-4-8 | 1-2-3-4-8 |
| Условие «Корректный ввод параметров». Проверка условия когда w=3, h=2 | w=«3», h=«2» | 1-2-3-4-8 | 1-2-3-4-8 |

Вывод: это такой набор тестов, который покрывает и условия, и решения одновременно.

**4.5 Комбинаторное покрытие условий**

Этот критерий требует создания такого числа тестов, чтобы все возможные комбинации результатов условий в каждом решении и все точки входа выполнялись, по крайней мере, один раз.

Комбинации

1)rB>w

2)rB>h 5)w > 0

2)w !=String 6)h > 0

3)h != String

Таблица 6 - Результат тестирования по принципу комбинаторное покрытие условий

|  |  |  |  |
| --- | --- | --- | --- |
| Назначение теста | Входные данные | Ожидаемый путь | Результат тестирования |
| Проверка комбинированного условия “Корректный ввод параметров” | w=4, h=6, rB = 3 | 1-2-3-4-8 | 1-2-3-4-8 |

Вывод: для программ, содержащих решение, каждое из которых имеет более одного условия, минимальный критерий состоит из набора тестов, вызывающих выполнение всех возможных комбинаций результатов условий в каждом решении и передающих управление каждой точки входа, по крайней мере, один раз. Термин “возможных” употреблён здесь потому, что некоторые комбинации условий могут быть не реализуемы.

5 Стратегия «чёрного ящика»

5. 1 Эквивалентное разбиение

Таблица 7 – Классы эквивалентности

|  |  |  |
| --- | --- | --- |
| Входное условие | Правильные классы эквивалентности | Неправильные классы эквивалентности |
| Ширина прямоугольника w | 1) 0<число  2) разделитель «.» | 3) Числа <=0  4) Буквы  5) Специальные символы  6) Разделитель «,» |
| Высота прямоугольника h | 1) 0<число  2) разделитель «.» | 3) Числа <=0  4) Буквы  5) Специальные символы  6) Разделитель «,» |
| радиус окружности rB | 1) 0<число  2) разделитель «.» | 3) Числа <=0  4) Буквы  5) Специальные символы  6) Разделитель «,» |

Так как высота и ширина — это не отрицательные величины, площадь не может быть отрицательной.

Вывод: правильные классы включают правильные данные, неправильные классы - неправильные данные.

Таблица 8 – Тестирование программы с использованием стратегии чёрного ящика

|  |  |  |
| --- | --- | --- |
| Назначение теста | Входные данные | Результат тестирования |
| Проверка значения правильного класса эквивалентности | w=3.5, h=3, rB=2 | Построится фигура |
| Проверка значения неправильного класса эквивалентности | w=3,5 h=2,rB=4 | Вывод сообщения:  «Окружность не может быть за пределами прямоугольника!» |
| Проверка значения правильного класса эквивалентности | w=3, h=2.5, rB=2 | Построится фигура |
| Проверка значения неправильного класса эквивалентности | R1=3 R2=2,5,rB=4 | Вывод сообщения:  «Окружность не может быть за пределами прямоугольника!» |
| Проверка значения правильного класса эквивалентности | w=3, h=2 | Построится фигура |
| Проверка значения неправильного класса эквивалентности | w=«ab», h=«2» | Вывод сообщения:  «Вы ввели не число или вообще ничего не ввели!» |
| Проверка значения неправильного класса эквивалентности | w=«2», h=«ab» | Вывод сообщения:  «Вы ввели не число или вообще ничего не ввели!» |
| Проверка значения неправильного класса эквивалентности | w=«af», h=«ab» | Вывод сообщения:  «Вы ввели не число или вообще ничего не ввели!» |
| Проверка значения неправильного класса эквивалентности | w=«0», h=«4» | Вывод сообщения:  «Вы ввели число вне диапазона!» |
| Проверка значения неправильного класса эквивалентности | w=«4», h=«0» | Вывод сообщения:  «Вы ввели число вне диапазона!» |
| Проверка значения неправильного класса эквивалентности | w=«-3», h=«2» | Вывод сообщения:  «Вы ввели число вне диапазона!» |
| Проверка значения неправильного класса эквивалентности | w=«3», h=«-2» | Вывод сообщения:  «Вы ввели число вне диапазона!» |

**Вывод:** Проведя тестирование программы с использованием стратегии чёрного ящика, можно сделать вывод, что метод позволяет выявить поведение программы, которое не соответствует спецификации. Замечаний по тестированию выявлено не было.

5. 2 Анализ граничных условий

Граничные условия - это ситуации, возникающие выше или ниже границ входных классов эквивалентности.

Таблица 9 – Результат тестирования по анализу граничных значений

|  |  |  |
| --- | --- | --- |
| Назначение теста | Входные данные | Результат тестирования |
| Проверка нижнего граничного  значения правильного класса  эквивалентности | w=1, h=1,rB=0.5 | Фигура построится |
| Проверка значения меньше  нижнего граничного значения  правильного класса эквивалентности | w=0, h=-1, rB=0.5 | Вывод сообщения:  «Вы ввели число вне диапазона!» |

Вывод:при данной разработке тестов рассматриваются не только входные значения, но и выходные. В данном случае мы проверяем тестом каждую границу этого класса.

5. 3 Предупреждение об ошибке

Процедура метода предположения об ошибке в значительной степени основана на интуиции. Основная идея метода состоит в том, чтобы перечислить в некотором списке возможные ошибки или ситуации, в которых они могут появиться, а затем на основе этого списка составить тесты. Другими словами, требуется перечислить те специальные случаи, которые могут быть не учтены при проектировании.

Таблица 10 – Результат тестирования ошибок

|  |  |  |
| --- | --- | --- |
| Назначение теста | Входные данные | Результат тестирования |
| Радиус звезды больше пятиугольника | w=«2», h=«3», rB= «4» | Вывод сообщения:  «Окружность выходит за пределы прямоугольника» |
| Ввод строки | w=«ab», h=«2» | Вывод сообщения:  «Вы ввели не число или вообще ничего не ввели!» |
| Ввод строки | w=«2», h=«ab» | Вывод сообщения:  «Вы ввели не число или вообще ничего не ввели!» |
| Ввод отрицательного числа | w=«-5», h=«4» | Вывод сообщения:  «Вы ввели число вне диапазона!» |
| Ввод отрицательного числа | w=«4», h=«-5» | Вывод сообщения:  «Вы ввели число вне диапазона!» |
| Ввод нуля | w=«0», h=«4» | Вывод сообщения:  «Вы ввели число вне диапазона!» |
| Ввод нуля | w=«5», h=«0» | Вывод сообщения:  «Вы ввели число вне диапазона!» |

6 Оценочное тестирование

После завершения функционального и структурного тестирования приступают к оценочному тестированию, целью которого является тестирование программы на соответствие основным требованиям. Эта стадия тестирования особенно важна для программных продуктов, предназначенных для продажи на рынке.

Таблица 11 - Результат оценочного тестирования

|  |  |  |
| --- | --- | --- |
| Назначение теста | Ожидаемый результат | Результат тестирования |
| Тестирование удобства использования | Программный продукт удобен в использовании. | Программный продукт удобен в использовании. |
| Тестирование на предельных объемах | Программа работает при вводе большого числа. | Программа работает при вводе большого числа, но нет сообщения, что фигуры не видно из-за того, что она выходит за размер области экрана. |
| Тестирование удобства эксплуатации | Интерфейс удобен, цветовое сопровождение и расположение элементов управления не раздражает. | Интерфейс вполне удобен, за счет расположения элементов управления, цветовое сопровождение не раздражает. |

7 Тестирование интерфейса

На рисунке 4 изображен пользовательский интерфейс программы.

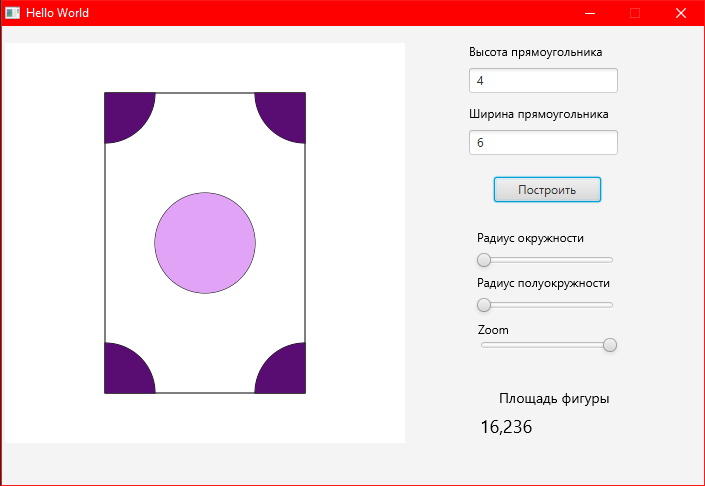


Рисунок 4 – Пользовательский интерфейс

Функциональные характеристики интерфейса:

* Взаимодействие с пользователем с помощью элементов управления;
* Вычисление площади фигуры;
* Локализация на0дписей экранных форм, а также сообщений, выдаваемых пользователю;

Таблица 11 – Тестирование пользовательского интерфейса программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Ожидаемый результат | Результат тестирования |
| 1 | Взаимодействие с пользователем с помощью элементов управления | Достаточное количество элементов управления, позволяющие выполнить необходимые действия пользователю. | Присутствуют элементы: область построения, поля для ввода радиуса звезды и размера фигуры, с лейблами которые показывают площадь |
| 2 | Вычисление площади фигуры | Чтобы получить площадь фигуры, нужно ввести радиус в поля радиусов и фигура сразу же построится. При любом изменения полей фигура буте манятся, при вводе некорректных значений будет выведено сообщение об ошибке. | При некорректном вводе значения радиуса выводится сообщение об ошибке.  При корректном вводе радиуса, рисуется фигура и в поле для площади выводится значение площади фигуры. |
| 3 | Построение фигуры | Чтобы построить фигуру, нужно ввести радиус звезды и пятиугольника, если введено допустимое значение радиуса, то будет построена фигура, иначе выведется ошибка | При корректном вводе значения радиуса окружности строится фигура.  При некорректном выводится ошибка. |
| 4 | Масштабирование фигуры относительно размера экрана | Чтобы масштабировать фигуру, нужно в поле с зумом ввести желаемый масштаб фигуры  Фигура должна масштабироваться относительно размера окна. При уменьшении/увеличении размера окна фигура также должна уменьшаться/ увеличиваться относительно новому размеру окна. | Фигура масштабируется при выборе масштаба в текстовом поле. Изменить размер окна пользователь не может. |

8 Отчет по тестированию программы

Каждое отклонение от спецификации обязательно документируют, заполняя специальный протокол.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название компании ИРНИТУ | | | | |  |
| Отчёт о проблеме № 2 | | | | |  |
| Программа «Тестирование» | Версия 0.1 | |  | | |
| Тип проблемы(1-6) 2 | Степень важности(1-3) 2 | | | Приложения(Д/Н) | |
| 1 - Ошибка компиляции  2 - Ошибка проектирования  3 - Предположение  4 - Расхождение с документацией  5 - Взаимодействие с аппаратурой  6 - Вопрос | | 1 - Фатальная  2 - Серьёзная  3 -Незначительная |  | | |
| ПРОБЛЕМА | | | | | |
| Можете ли вы воспроизвести проблемную ситуацию? (Д/Н) | | | | | |
| Подробное описание проблемы и способа её воспроизведения: | | | | | |
|  | | | | | |
| Не ограничен ввод чисел по диапазону. Если ввести очень большое число, программа будет строить не корректную фигуру или вообще её не строить. | | | | | |
|  | | | | | |
|  | | | | | |
| Предлагаемое исправление (необязательно): | | |  | | |
| Добавить разумное ограничение на диапазон вводимых чисел. | | | | | |
|  | | | | | |
|  | | | | | |
| Отчёт предоставлен | Путинцев О.Р. | | Дата 15.04.2022 | | |
|  |  | |  | | |

9 Листинг

Класс Main

public class Main extends Application {  
 public Main() {  
 }  
  
 public void start(Stage primaryStage) throws Exception {  
 Parent root = (Parent)FXMLLoader.*load*(this.getClass().getResource("sample.fxml"));  
 primaryStage.setTitle("Hello World");  
 primaryStage.setScene(new Scene(root));  
 primaryStage.setResizable(false);  
 primaryStage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

Класс Controller

public class Controller implements Initializable {  
 @FXML  
 private Slider sld1;  
 @FXML  
 private Slider zoom1;  
 @FXML  
 private Slider sld2;  
 @FXML  
 private Canvas mainCanvas;  
 @FXML  
 private TextField widthTxt;  
 @FXML  
 private TextField heightTxt;  
 @FXML  
 private Button button;  
 @FXML  
 private Text resultField;  
  
 public Controller() {  
 }  
  
 public void initialize(URL location, ResourceBundle resources) {  
 GraphicsContext ctx = this.mainCanvas.getGraphicsContext2D();  
 ctx.setFill(Color.*WHITE*);  
 ctx.fillRect(0.0D, 0.0D, this.mainCanvas.getWidth(), this.mainCanvas.getHeight());  
 this.sld1.valueProperty().addListener((observable, oldValue, newValue) -> {  
 ctx.fillRect(0.0D, 0.0D, this.mainCanvas.getWidth(), this.mainCanvas.getHeight());  
 this.draw(Double.*parseDouble*(this.widthTxt.getText()), Double.*parseDouble*(this.heightTxt.getText()), this.sld1.getValue(), this.sld2.getValue(), ctx);  
 this.area(Double.*parseDouble*(this.widthTxt.getText()), Double.*parseDouble*(this.heightTxt.getText()), this.sld1.getValue(), this.sld2.getValue());  
 });  
 this.sld2.valueProperty().addListener((observable, oldValue, newValue) -> {  
 ctx.fillRect(0.0D, 0.0D, this.mainCanvas.getWidth(), this.mainCanvas.getHeight());  
 this.draw(Double.*parseDouble*(this.widthTxt.getText()), Double.*parseDouble*(this.heightTxt.getText()), this.sld1.getValue(), this.sld2.getValue(), ctx);  
 this.area(Double.*parseDouble*(this.widthTxt.getText()), Double.*parseDouble*(this.heightTxt.getText()), this.sld1.getValue(), this.sld2.getValue());  
 });  
 this.button.setOnAction((event) -> {  
 if (this.isDouble(this.widthTxt.getText()) && this.isDouble(this.heightTxt.getText()) && this.inRangeHeight(this.heightTxt.getText()) && this.inRangeWidth(this.widthTxt.getText())) {  
 ctx.fillRect(0.0D, 0.0D, this.mainCanvas.getWidth(), this.mainCanvas.getHeight());  
 this.area(Double.*parseDouble*(this.widthTxt.getText()), Double.*parseDouble*(this.heightTxt.getText()), this.sld1.getValue(), this.sld2.getValue());  
 this.draw(Double.*parseDouble*(this.widthTxt.getText()), Double.*parseDouble*(this.heightTxt.getText()), this.sld1.getValue(), this.sld2.getValue(), ctx);  
 this.zoom1.setMax(400.0D);  
 this.zoom1.setMin(50.0D);  
 this.zoom1.setValue(this.zoom1.getMax());  
 } else if (!this.isDouble(this.widthTxt.getText()) || !this.isDouble(this.heightTxt.getText())) {  
 Alert alert = new Alert(AlertType.*INFORMATION*);  
 alert.setTitle("Ошибка ввода");  
 alert.setHeaderText("Вы ввели строку вместо числа или ничего не ввели!");  
 alert.showAndWait();  
 this.widthTxt.clear();  
 this.heightTxt.clear();  
 }  
  
 });  
 this.zoom1.valueProperty().addListener((observable, oldValue, newValue) -> {  
 ctx.fillRect(0.0D, 0.0D, this.mainCanvas.getWidth(), this.mainCanvas.getHeight());  
 this.mainCanvas.setHeight(this.zoom1.getValue());  
 this.mainCanvas.setWidth(this.zoom1.getValue());  
 ctx.fillRect(0.0D, 0.0D, this.mainCanvas.getWidth(), this.mainCanvas.getHeight());  
 double k = 400.0D / this.zoom1.getValue();  
 this.draw(Double.*parseDouble*(this.widthTxt.getText()) / k, Double.*parseDouble*(this.heightTxt.getText()) / k, this.sld1.getValue() / k, this.sld2.getValue() / k, ctx);  
 this.area(Double.*parseDouble*(this.widthTxt.getText()) / k, Double.*parseDouble*(this.heightTxt.getText()) / k, this.sld1.getValue() / k, this.sld2.getValue() / k);  
 });  
 }  
  
 private void draw(Double h, Double w, Double rB, Double rS, GraphicsContext ctx) {  
 double zoom = 50.0D;  
 ctx.save();  
 Affine transform = ctx.getTransform();  
 transform.appendTranslation(this.mainCanvas.getWidth() / 2.0D, this.mainCanvas.getHeight() / 2.0D);  
 transform.appendScale(zoom, zoom);  
 ctx.setTransform(transform);  
 ctx.setLineWidth(1.0D / zoom);  
 ctx.strokePolygon(new double[]{w / 2.0D, w / 2.0D, -w / 2.0D, -w / 2.0D}, new double[]{h / 2.0D, -h / 2.0D, -h / 2.0D, h / 2.0D}, 4);  
 ctx.strokeArc(w / 2.0D - rS, -h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, -90.0D, -90.0D, ArcType.*ROUND*);  
 ctx.setFill(Color.*web*("#590d73"));  
 ctx.fillArc(w / 2.0D - rS, -h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, -90.0D, -90.0D, ArcType.*ROUND*);  
 ctx.strokeArc(-w / 2.0D - rS, h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, 90.0D, -90.0D, ArcType.*ROUND*);  
 ctx.setFill(Color.*web*("#590d73"));  
 ctx.fillArc(-w / 2.0D - rS, h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, 90.0D, -90.0D, ArcType.*ROUND*);  
 ctx.strokeArc(-w / 2.0D - rS, -h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, -90.0D, 90.0D, ArcType.*ROUND*);  
 ctx.setFill(Color.*web*("#590d73"));  
 ctx.fillArc(-w / 2.0D - rS, -h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, -90.0D, 90.0D, ArcType.*ROUND*);  
 ctx.strokeArc(w / 2.0D - rS, h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, 90.0D, 90.0D, ArcType.*ROUND*);  
 ctx.setFill(Color.*web*("#590d73"));  
 ctx.fillArc(w / 2.0D - rS, h / 2.0D - rS, 2.0D \* rS, 2.0D \* rS, 90.0D, 90.0D, ArcType.*ROUND*);  
 double[] coord = new double[2];  
 this.cross(coord, w / 2.0D, h / 2.0D, -w / 2.0D, -h / 2.0D, w / 2.0D, -h / 2.0D, -w / 2.0D, h / 2.0D);  
 ctx.strokeOval(coord[0] - rB, coord[1] - rB, 2.0D \* rB, 2.0D \* rB);  
 ctx.setFill(Color.*web*("#e0a3f5"));  
 ctx.fillOval(coord[0] - rB, coord[1] - rB, 2.0D \* rB, 2.0D \* rB);  
 ctx.restore();  
 Alert alert;  
 if (rB > h / 2.0D) {  
 alert = new Alert(AlertType.*INFORMATION*);  
 alert.setTitle("Ошибка ввода");  
 alert.setHeaderText("Вы вышли за пределы диапазона!");  
 alert.showAndWait();  
 this.sld1.setValue(h / 2.0D - 1.0D);  
 } else if (rB > w / 2.0D) {  
 alert = new Alert(AlertType.*INFORMATION*);  
 alert.setTitle("Ошибка ввода");  
 alert.setHeaderText("Вы вышли за пределы диапазона!");  
 alert.showAndWait();  
 this.sld1.setValue(w / 2.0D - 1.0D);  
 }  
  
 }  
  
 boolean isDouble(String str) {  
 boolean result;  
 try {  
 Double.*parseDouble*(str);  
 result = true;  
 } catch (Exception var4) {  
 result = false;  
 }  
  
 return result;  
 }  
  
 boolean inRangeWidth(String str) {  
 boolean result;  
 if (Double.*parseDouble*(str) <= 9.1D && Double.*parseDouble*(str) >= 1.0D) {  
 result = true;  
 } else {  
 Alert alert = new Alert(AlertType.*INFORMATION*);  
 alert.setTitle("Ошибка ввода");  
 alert.setHeaderText("Вы вышли за пределы диапазона!");  
 alert.showAndWait();  
 result = false;  
 }  
  
 return result;  
 }  
  
 boolean inRangeHeight(String str) {  
 boolean result;  
 if (Double.*parseDouble*(str) <= 8.49D && Double.*parseDouble*(str) >= 1.0D) {  
 result = true;  
 } else {  
 result = false;  
 Alert alert = new Alert(AlertType.*INFORMATION*);  
 alert.setTitle("Ошибка ввода");  
 alert.setHeaderText("Вы вышли за пределы диапазона!");  
 alert.showAndWait();  
 }  
  
 return result;  
 }  
  
 boolean cross(double[] intersectionPoints, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4) {  
 double n;  
 if (y2 - y1 != 0.0D) {  
 double q = (x2 - x1) / (y1 - y2);  
 double sn = x3 - x4 + (y3 - y4) \* q;  
 if (sn == 0.0D) {  
 return false;  
 }  
  
 double fn = x3 - x1 + (y3 - y1) \* q;  
 n = fn / sn;  
 } else {  
 if (y3 - y4 == 0.0D) {  
 return false;  
 }  
  
 n = (y3 - y1) / (y3 - y4);  
 }  
  
 intersectionPoints[0] = x3 + (x4 - x3) \* n;  
 intersectionPoints[1] = y3 + (y4 - y3) \* n;  
 return true;  
 }  
  
 void area(double rB, double rS, Double h, Double w) {  
 double s1 = 3.141592653589793D \* rS \* rS;  
 double s2 = 3.141592653589793D \* rB \* rB;  
 double sr = Math.*sqrt*(Math.*pow*(h \* w - s1 - s2, 2.0D)) / 10.0D;  
 this.resultField.setText(String.*format*("%.3f", sr));  
 }  
}