**НАДЁЖНОСТЬ ПРОГРАММНОГО ОБЕМПЕЧЕНИЯ**

Лабораторная работа № 4

Тестирование программного обеспечения в **SilkTest 8.0** (4 часа)

1. Краткие сведения

SilkTest – инструмент автоматизированного тестирования прикладных программ через графический интерфейс пользователя. Программа предназначена для регрессионного, кросс-платформенного и локализационного тестирования для широкого диапазона технологий разработки приложений, в том числе Web, Java, .NET и клиент-серверных технологий.

SilkTest предоставляет уровень абстракции GUI, так называемые карты отображения GUI (GUI maps), сопоставляющие графические элементы тестируемого приложения с объектами более низкого уровня, генерируемого SilkTest;

Поддерживаемые операционные системы: Microsoft Windows.

Язык скриптов:  4Test.

В SilkTest среди файлов выделяют файлы двух типов:

–Фреймы (frame) – это описания окон, с которыми работает SilkTest. Фреймы хранятся в файлах с расширением **\*.inc** (рисунок 1);

– Скрипты или тест-кейсы (testcase) – код программы, в котором описываются действия с тестируемой системой. Скрипты хранятся в файлах с расширением **\*.t**.

1.1 Создание фреймов

Существует 2 способа записи окна:

1. File - New - Test Frame - Ok - Выбор открытого тестируемого приложения - Ок.

2) Record -Window Declarations. Открывается окно Record Window Declarations, где отображаются объекты окна приложения, на котором в данный момент находится курсор мыши. Чтобы остановить процесс сканирования и поместить информацию о текущем окне в документ SilkTest, необходимо нажать комбинацию клавиш Ctrl-Alt и нажать кнопку Paste to Editor.

1.2 Создание тест-кейсов

Файл скрипта содержит следующие компоненты:

– Заголовок. Комментарий, характеризующий содержимое данного файла.

– Подключение файлов. Набор Use-директив, которые подключают необходимые для данного скрипта файлы;

– Блок декларации внешних переменных. Область для объявления переменных, констант, которые используются различными тест-кейсами;

– Тест-кейсы – модули со сценариями автоматического тестирования. Каждый сценарий помещается в модули с ключевым словом testcase;

– Функция main – точка входа в программу.

1.3 Реализация технологии Record/PlayBack в среде SilkTest

Особенности реализация технологии Record/PlayBack в SilkTest рассмотрены для функций создания библиотек объектов и автоматических тестовых скриптов, для процедуры запуска скриптов и использования средств опознавания объектов.

1.3.1. Создание библиотеки объектов

Для создания библиотеки объектов (рисунок 1.1):

1. Запустить SilkTest.

2. Открыть тест-приложение, например «WordPad» .

3. Выбрать в меню SilkTest команду File-New-4Test Include file.

4. Сохранить его как MyFile1.inc.

5. Выбрать в меню SilkTest команду Record-Window Declarations.

6. Навести курсор мыши на тест-приложение.

7. Нажать комбинацию клавиш «Ctrl+Alt».

8. Нажать кнопку «Paste to Editor» на форме «Record Window Declarations».

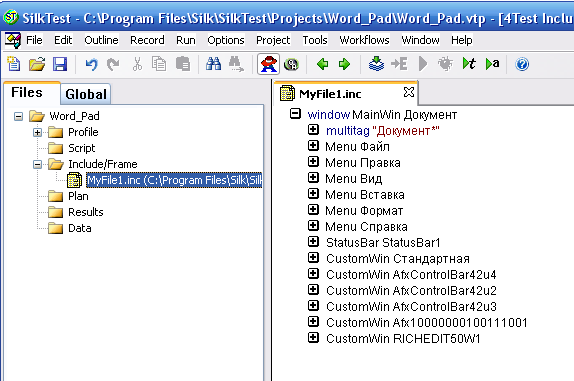


Рисунок 1.1 – Результат создания библиотеки объектов

Для использования объектов библиотеки, в скрипт необходимо включить следующий код: Use "MyFile1.inc".

1.3.2 Создание тестового скрипта

Для создания тестового скрипта следует (рисунок 2):

1. Запустить SilkTest.

2. Открыть тест-приложение, например «WordPad».

3. Выбрать в меню SilkTest команду File-New-4Test script.

4. Сохранить его как MyFile2.t.

5. Выбрать в меню SilkTest команду Record-Testcase.

6. Нажать кнопку Start Recording.

7. Выполнить заданные действия над тест-приложением.

8. Нажать кнопку «Done».

9. Нажать кнопку «Paste to Editor».

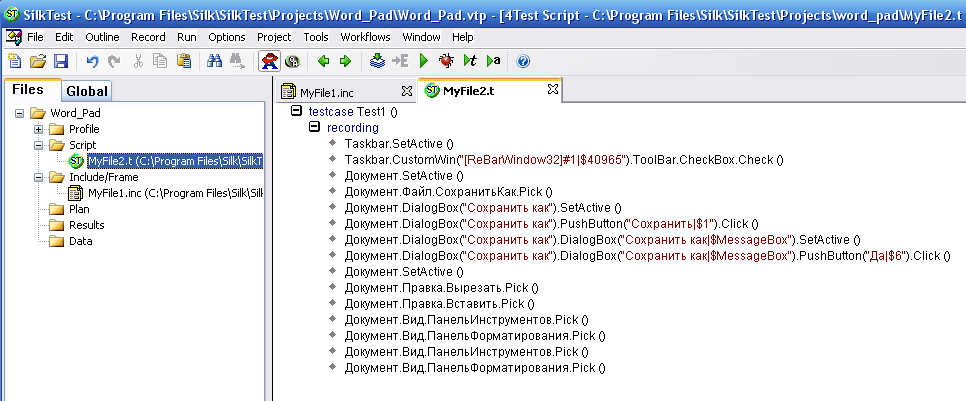


Рисунок 1.2 – Результат создания тестового скрипта

На рисунке 1.2 представлен фрагмент программного кода, выражающего предпринятые действия. Это готовый для исполнения код.

Все выполняемые строки кода находятся внутри блока testcase. Один скрипт может иметь множество тест-кейсов, исполняемых по очереди. Для того чтобы выполнить лишь один тест-кейс, необходимо нажать кнопку  и выбрать нужное значение из списка доступных тест-кейсов.

1.3.3 Средство распознавания объектов

Когда необходимо просмотреть описание конкретного объекта, следует использовать средство распознавания объектов SilkTest. Его можно вызвать следующим образом: Record-Window Identifiers. Чтобы получить информацию об объекте, необходимо навести на него курсор и нажать Ctrt+Alt. Описание можно скопировать в файл, нажав на кнопку Paste to Editor.

1.3.4 Recovery-система

При разработке тест-кейсов часто возникает необходимость написания кода, который готовит тестовую среду к запуску скриптов или выполняет некоторые предварительные действия. Инициализацию данных перед запуском скриптов можно осуществлять разными способами. Одним из вариантов является разработка функции или тест-кейса, который осуществит необходимую инициализацию и будет запускаться перед запуском других тест-кейсов. Этот вариант имеет недостаток, поскольку каждый тест-кейс может запускаться как в группе, так и поодиночке и инициализацию данных придется интегрировать в каждый тест-кейс. Более удобным вариантом будет помещение инициализации данных в AppState. Для таких целей в SilkTest существует Recovery-система.

Данная система реализована в виде набора функций:

– ScriptEnter – выполняется в самом начале выполнения файла скрипта

– ScriptExit – выполняется сразу по завершении выполнения файла скрипта

– TestCaseEnter – выполняется сразу перед началом выполнения отдельного тесткейса (до AppState)

– TestCaseExit – выполняется сразу после завершения тесткейса (после AppState)

– TestPlanEnter – выполняется сразу перед началом выполнения тестплана

– TestPlanExit – выполняется сразу после выполнения тестплана

Это набор функций, который можно переопределить. В противном случае будут задействованы функции по-умолчанию (DefaultScriptEnter, DefaultScriptExit, DefaultTestCaseEnter, DefaultTestCaseExit, DefaultTestPlanEnter, DefaultTestPlanExit).

1.3.5 Создание тестплана (TestPlan)

Тестплан в SilkTest – это последовательный набор инструкций для запуска тест-кейсов пакетом. Большие тестпланы могут быть поделены как master test plan и один или более subplans. Файлы, содержащие тестпланы имеют расширение .pln, например find.pln.

Для создания плана тестирования нужно выбрать пункт меню Меню File-New-Testplan.

Для того, чтобы добавить в план тесткейс, необходимо:

– задать групповое описание тесткейсов и нажать Enter;

– сделать новый абзац;

– ввести скрипт и/или тест-кейс используя нижеописанный синтаксис:

***Script****: scriptname.t, //* имя файла, содержащего тест-кейсы;

***Testcase****: testcasename, //* название тест-кейса;

***Testdata****: testdata //* передаются дополнительные тестовые данные

***Optionset****: filename.opt, //* подключаемый файл свойств

Для соединение master plan с subplan необходимо добавить include выражение в ***master plan***:

***include***: myinclude.pln // файл плана, который является subplan

Для того, чтобы добавить комментарии к тесткейсу следует использовать нижеописанный синтаксис:

***comment***: Your\_comment\_text; // комментарии в тестплан

Пример подключения используемых файлов и фреймов:

[-] *header: // Please do not modify.*

[ ] *usepath:*

[ ] *usefiles: D:\Energizer\Declarations\BaseStateWB.inc, extend\explorer.inc*

[ ] *framefile: D:\Energizer\Declarations\BaseStateWB.inc*

Для запуска тестплана на выполнение необходимо выбрать пункт меню Run->Run All Testcases или нажать F9. Существует возможность выборочного запуска тест-кейсов из тестплана. Для пометки тест-кейса нужно установить курсор на тест-кейсе и выбрать пункт меню Testplan -> Mark. Для запуска помеченного множества тест-кейсов на выполнение необходимо выбрать пункт меню Run->Marked Tests.

1.3.6 Использование внешних данных (Data Driven Test)

SilkTest позволяет, помимо объявления переменных в тест-кейсах, использовать данные, которые считываются из какого-либо внешнего источника (базы данных, текстовых файлов в определенном формате, электронной таблицы). Тест-кейсы, считывающие данные из внешнего источника, называются Data Driven Testcases. Подобное хранение данных в одном месте упрощает их просмотр и редактирование.

Рассмотрим *пример подключения внешних данных из Excel-файла*.

**Шаг 1**: Создадим файл с именем Search.

**Шаг 2**: Заполним Excel-таблицу данными для поиска (рисунок 1.3).

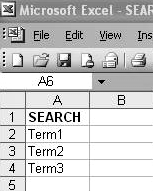


Рисунок 1.3 – Шаблон подключаемых внешних данных

**Шаг 3**: зайти в SilkTest Window и выбрать Tools.

**Шаг 4**: выбрать Data Driven Testcase (тест-кейс должен быть открыт в SilkTest).

**Шаг 5**: открыть окно Select Testcase. Выбрать тест и нажать Ok.

**Шаг 6:** открыть окно Specify Data Driven Script. Выбрать «Create a new file/Overwrite an existing file» и нажать Ok.

**Шаг 7:** открыть окно Select Data Source. Выбрать Silk DDA Excel. Выбрать Excel-файл через диалог Browse и нажать OK после выбора.

**Шаг 8:** открыть Specify Data Driven Testсase окно. Выбрать Add a new Data Driven Testcase и нажать OK.

**Шаг 9:** открыть окно Find/Replace Values. Нажать Cancel в Find/Replace Values.

**Шаг 10**: войти в файл с именем скрипта, но с расширением .g.t.

**Шаг 11:** в файле выбрать тип данных (Find values of type). Если данные представляют собой текст, выбрать Text. В Replace value with для Table выбрать Sheet1$, в Column выбрать столбец с именем, который присвоен столбцу в Excel-файле. После выбора таблицы и столбца становится доступной кнопка Replace.

**Шаг 12:** указанные значения будут вставлены в DD-скрипт.

**Шаг 13:** нажать кнопку Replace.

**Шаг 14:** нажать Save и выбрать RunTestCase. В окне Run TestCase выбрать DD-скрипт. Нажать Run.

1.3.7 Просмотр результатов исполнения тестовых сценариев

После запуска тестовых сценариев SilkTest выводит окно Results с результатами отработки тест-кейсов. В нем отображается статистика по пройденным тест-кейсам, время, затраченное на прохождение всех тест-кейсов (рисунок 1.4)



Рисунок 1.4 – Результат выполнения тестового сценария

Также в этом окне отображается информация, выводимая в процессе работы скрипта функциями Print, ListPrint, LogError, LogWarning. Для просмотра информации, выводимой в процессе работы тест-кейса, необходимо нажать на значок [+] слева от названия тест-кейса.

2. Практическая часть (порядок выполнения лабораторной работы)

2.1. Получите у преподавателя готовый тестплан проверки выполнения операции «Сложение» в двоичной системе счисления над 2-байтными операндами для калькулятора вида «Программист».

2.2. Изучите созданный тест план. Описание примера разработки тест см. Приложение 1.

2.3. Согласно выданному преподавателем варианту № 1-12 (таблица 1) выполните детализацию тестов, разработав 3 тест-кейса. Входные данные читать из Excel-файла. Для разработанных тест-кейсов создать 3 тестплана. Общие шаги инструкций поместить в Recovery-файл.

Таблица 1– Варианты заданий тестирования

|  |  |
| --- | --- |
| **№** | **Описание задания, шаги инструкции** |
| **1** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: инженерные расчеты в 8-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Ave", "Sta", "7", "8".  – проверить отмечены ли чекбоксы "Inv" и "Hyp".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать выполнение 2-байтных операций сложения в 8-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **2** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: инженерные расчеты в 8-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "7", "9", "Backspase", "Sum" .  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать выполнение 1-байтных операций вычитания в 8-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **3** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: инженерные расчеты в 2-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "2", "3", "4", "5", "0", "1" .  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать выполнение 4-байтных операций умножения в 2-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **4** | тест-приложение: «Калькулятор», вид «Программист»;  тестирование функции: инженерные расчеты в 16-ой системе счисления; |
| – выбрать пункт меню «Вид – Инженерный».  – проверить доступность кнопок "Ave", "Sta", "CE", "C", "D", "E".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать выполнение 1-байтных операций деления в 16-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть главное окно «Калькулятор». |
| **5** | тест-приложение: «Калькулятор», вид «Программист»;  тестирование функции: инженерные расчеты в 2-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "2", "3", "4", "5", "0", "1", "MC".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать выполнение 2-байтных операций деления в 2-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **6** | тест-приложение: «Калькулятор», вид «Программист»;  тестирование функции: статистические расчеты в 2-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "2", "3", "4", "5", "0", "1", "M+".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать статистические операции: "Dat", "Ave", "Sum" для 1-байтных операций в 2-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **7** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: статистические расчеты в 10-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "2", "3", "4", "5", "A", "C" .  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать статистические операции: "Dat", "S", "Sum" для 10-ой системы счисления.  – протестировать функции: "RET", "LOAD".  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **8** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: статистические расчеты в 16-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "3", "4", "5", "A", "C", "Exp".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать статистические операции: "Dat", "S", "Ave" для 1-байтной операции в 16-ой системе счисления.  – протестировать функции: "CD", "CAD".  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **9** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: работа с памятью в 16-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "Ave", "Dat", "4", "A", "C", "Exp".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать работу с памятью: "MS", "MR", "MC", "M+" для 2-байтных операций в 16-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **10** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: работа с памятью в 2-ой системе счисления; |
|  | – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "Ave", "Cos", "MC", "MR", "MS", "M+".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать работу с памятью: "MS", "MR", "MC", "M+" для 1-байтных операций в 2-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **11** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: работа с памятью в 2-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "Ave", "Cos", "MC", "MR", "MS", "M+".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать работу с памятью: "MS", "MR", "MC", "M+" для 1-байтных операций в 2-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |
| **12** | тест-приложение: «Калькулятор», вид «Классический»;  тестирование функции: работа с памятью в 2-ой системе счисления; |
| – выбрать пункт меню «Вид-Инженерный».  – проверить доступность кнопок "Sta", "Ave", "Cos", "MC", "MR", "MS", "M+".  – получить значения координат из внешних данных (X и Y).  – переместить калькулятор по полученным координатам.  – текущее значение в поле ввода сбросить в 0.  – протестировать работу с памятью: "MS", "MR", "MC", "M+" для 1-байтных операций в 2-ой системе счисления.  – вернуть вид приложения «Вид-Обычный».  – закрыть окно «Калькулятор». |

**ПРИЛОЖЕНИЕ А**

#### Пример разработки теста

**Задание:** Разработать тесты для проверки выполнения операции «Сложение» в двоичной системе счисления над 2-байтными операндами для калькулятора вида «Программист».

Тест-кейс №1 – Проверка выполнения операции сложения в случае переполнения:

– Запустить приложение;

– Проверить запущено ли приложение.

– Переключиться на вид «Инженерный».

– Проверить переключение на вид «Инженерный».

– Переключиться на вид «Программист»;

– Проверить переключение на вид «Программист».

– Переключиться на операцию «Bin - 2 байта»;

– Проверить переключение на операцию «Bin-2 байта».

– Проверить доступность кнопок для двоичной системы счисления (доступны кнопки: 0, 1; недоступны кнопки: 2, 3, 4, 5, 6, 7, 8, 9, А, В, С, D, Е, F);

– Ввести первое слагаемое (1111 1111 1111 1111);

– Набрать знак операции «+»;

– Ввести второе слагаемое (1);

– Нажать клавишу «=»;

– Сравнить результаты. **Ожидаемый результат: 0**.

Тест-кейс №2 – Проверка выполнения операции сложения над отрицательными числами:

– Запустить приложение;

– Проверить запущено ли приложение;

– Переключиться на вид «Инженерный»;

– Проверить переключение на вид «Инженерный»:

– Переключиться на вид «Программист»;

– Проверить переключение на вид «Программист»;

– Переключиться на операцию «Bin - 2 байта»;

– Проверить переключение на операцию «Bin-2 байта»;

– Проверить доступность кнопок для двоичной системы счисления (доступны – кнопки: 0, 1; недоступны кнопки: 2, 3, 4, 5, 6, 7, 8, 9, А, В, С, D, Е, F);

– Набрать первое слагаемое (1010);

– Нажать клавишу +/- для смены знака;

– Проверить результат (-1010);

– Набрать знак операции «+»;

– Проверить перевод первого слагаемого в дополнительный код (1111 1111 1111 0110);

– Набрать второе слагаемое (1011);

– Нажать клавишу +/- для смены знака;

– Проверить результат (-1011);

– Нажать клавишу «=»;

– Сравнить результаты. **Ожидаемый результат: 1111 1111 1110 1011**.

Тест-кейс №3 – Проверка выполнения операции сложения с использованием Data Driven Testcase:

– Запустить приложение;

– Проверить запущено ли приложение;

– Переключиться на вид «Инженерный»;

– Проверить переключение на вид «Инженерный»;

– Переключиться на вид «Программист»;

– Проверить переключение на вид «Программист»;

– Переключиться на операцию «Bin - 2 байта»;

– Проверить переключение на операцию «Bin-2 байта»;

– Проверить доступность кнопок для двоичной системы счисления (доступны кнопки: 0, 1; недоступны кнопки: 2, 3, 4, 5, 6, 7, 8, 9, А, В, С, D, Е, F);

– Ввод первого операнда (Х) (из Excel-файла);

– Набрать знак операции «+»;

– Ввод второго операнда (Y) (из Excel-файла);

– Нажать клавишу «=»;

– Проверить результат (Z) (из Excel-файла);

Дополнительные требования к созданию тестов

1) Разработать следующие тестовые сценарии или тестпланы (Test Plan):

– запуск только Тест-кейса №1;

– запуск только Тест-кейса №2;

– запуск только Тест-кейса №3;

– запуск Тест-кейсов №1, №2, №3.

2) Тест-кейс №3 создать как Data-Driven Testcase, используя Excel-файл для выбора тестовых данных.

Формат Excel-файла:

|  |  |  |
| --- | --- | --- |
| Первый операнд Х | Второй операнд Y | Ожидаемый результат Z |
| 1000101 | 0111010 | 1111111 |

3) Общую функциональность тест-кейсов поместить в Recovery-файл.

Выполнение задания

Прежде всего создадим новый проект, куда будут помещены разработанные скрипты и тестпланы (File-New Project). В открывшемся окне указать пункт Classic Agent. Нажать Оk (рисунок 1.5). После задания имени проекта (рисунок 1.6), откроется окно рисунок 1.7, в которое можно помещать описания окон и скрипты.

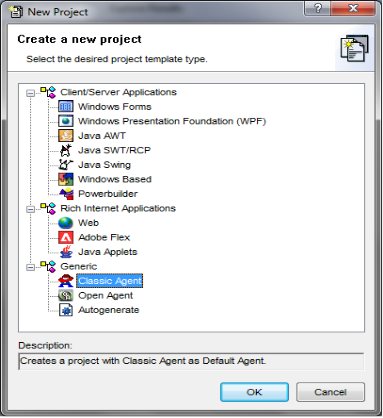


Рисунок 1.5 – Создание нового проекта

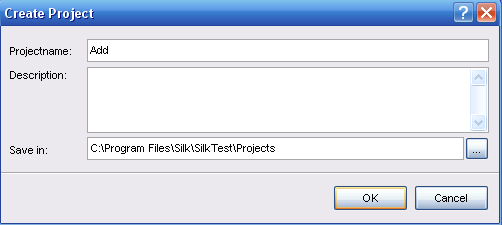


Рисунок 1.6 – Параметры нового проекта

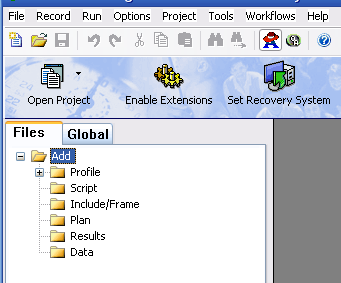


Рисунок 1.7 – Структура проекта

Теперь необходимо подготовить тестовое приложение. Его можно открыть из главного меню Пуск-Программы-Стандартные-Калькулятор либо запустить командой calc.exe в командной строке.

Приступим к записи описаний окон или оконных деклараций. Для этого в SilkTest следует выбрать пункт меню Record-Window Declarations. После появления окна Record Window Declaration, наведите курсор на заголовок окна Calculator и нажмите комбинацию клавиш Ctrl+Alt. В результате окно Record Window Declaration примет вид, показанный на рисунок 1.8.

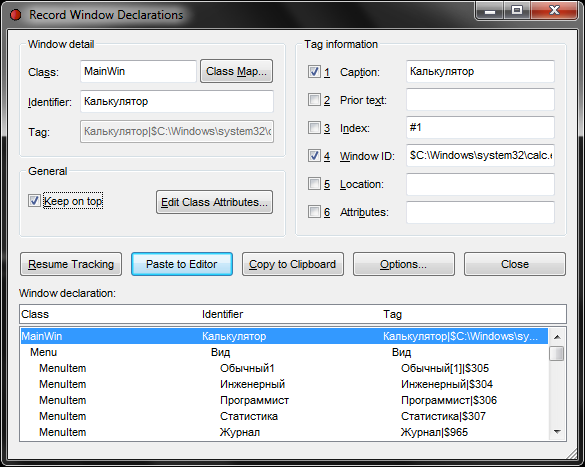


Рисунок 1.8 – Окно Record Window Declaration

Как видно среди описания окон выделено собственно главное окно приложения. В разделе Window detail видно, что окно является окном класса MainWin (в зависимости от того, к какому классу принадлежит окно или элемент управления, с ним можно выполнять разные действия). Идентификатор окна «Калькулятор» – это, фактически, имя этого окна. Далее представлено поле Tag. Тег (Tag) – это уникальная характеристика окна, по которому SilkTest распознает элементы управления в приложениях. В правой части окна Record Window Declaration можно просмотреть более подробную информацию о теге. В нашем случае тег состоит из заголовка (Caption) и идентификатора (Window ID). Кроме того, у окна есть индекс (Index), в нашем случае он равен "#1", однако соответствующая галочка возле него выключена, следовательно, он не будет присутствовать в теге.

Собственно сейчас уже можно вставлять декларацию окна в документ SilkTest, однако прежде сделаем дополнительные настройки в системе распознавания окон (рисунок 1.9). Нажав кнопку Options, отключим опцию Record Multiple Tags в нижней части окна Options. В разделе Default tag выберем опцию Caption. В разделе Window declaration identifiers выберем опцию Use the Caption. Нажмем Оk.

Теперь в окне Record Window Declaration нажмем кнопку Resume Tracking (для возобновления записи). Наведем курсор мыши на приложение Калькулятор и нажмем Ctrl+Alt. Как видно, окно Record Window Declaration изменилось. Теперь в качестве тега может выступать только один идентификатор (например, Caption), а не несколько, как было в прошлый раз.

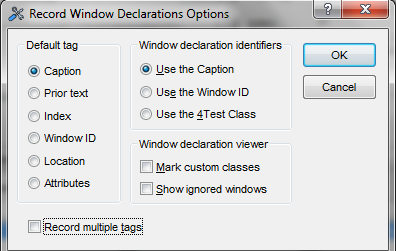


Рисунок 1.9 – Настройка системы распознования окон

Нажав кнопку Paste to Editor, вставим описание окна Калькулятор в документ SilkTest.

В полученном примере (рисунок 1.10) MainWin, Menu и MenuItem – это классы, по которым SilkTest различает, как работать с тем или иным элементом; Калькулятор, Правка, Вид, Справка – это имена объектов приложения. Имена объектов можно (а иногда даже нужно) менять. Например, имя Калькулятор слишком длинно, его можно заменить на Calc.

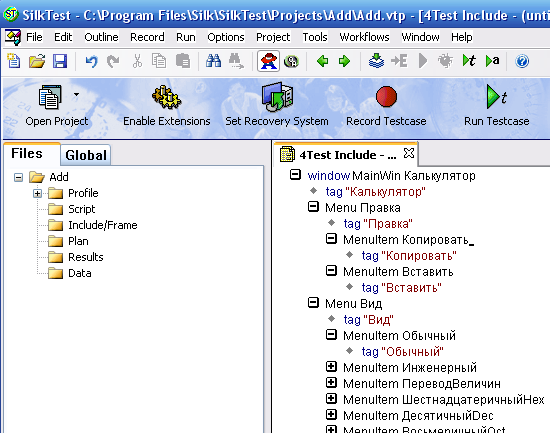


Рисунок 1.10 – Описание элементов тестируемого приложения

Для настройки Recovery System необходимо нажать кнопку Set Recovery System, в результате появится окно (рисунок 1.11).

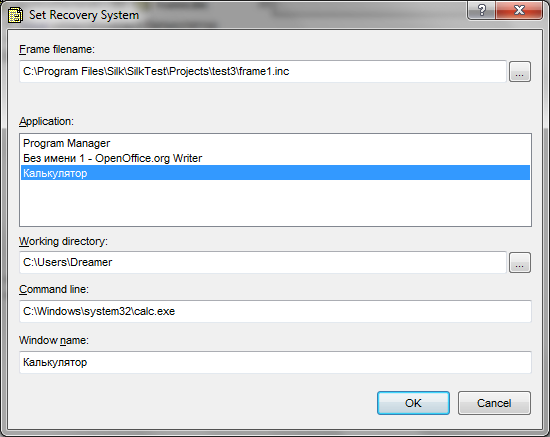


Рисунок 1.11 – Recovery System

В окне выберем тестируемое приложение «Калькулятор» и нажмем кнопку Ok.

В результате чего в проект будет добавлен Recovery-файл, содержащий информацию о тестируемом приложении, в частности вводимые константы инициализируемые для дальнейшего использования в тесткейсах (рисунок 1.12).

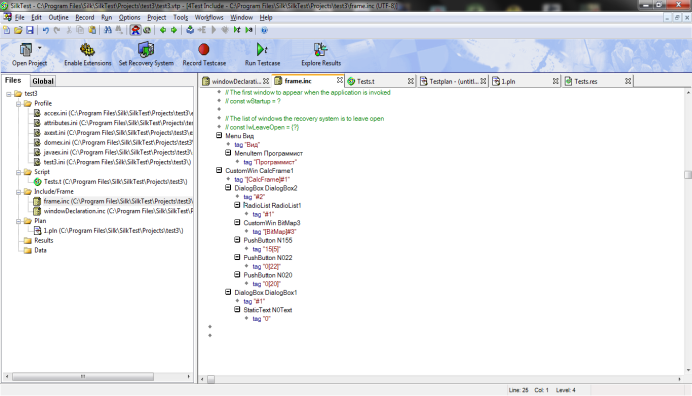


Рисунок 1.12 – Добавление Recovery-файла

После проведения данных операций, мы избавляемся от необходимости запуска приложения в каждом тесткейсе, за это будет отвечать Recovery System.

Когда оконные декларации созданы, можно приступить к записи и воспроизведению скрипта.

Для начала создадим тесткейс №1. Для этого выберем пункт меню Record-Testcase. На экране появится окно Record Testcase. В поле Testcase name введем имя теста OverflowTest (Tест-кейс №1 проверяет приложение на переполнение) или оставим существующее. Нажмем кнопку Start Recording (рисунок 3.25).

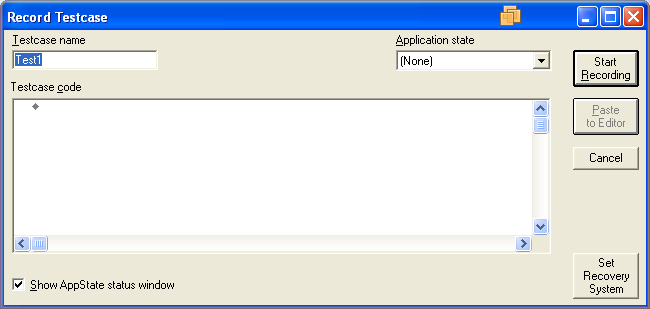


Рисунок 3.25.

На экране в правом нижнем углу появится окошко Record Status, в котором отображается статус записи и текущее окно, находящееся под курсором мыши (рисунок 3.26).

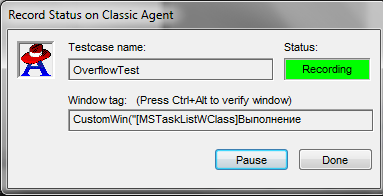


Рисунок3.26.

Кнопка Pause позволяет приостановить запись теста, а кнопка Done останавливает процесс записи тест-кейса.

Необходимо записать следующие шаги:

1. Переключить приложение на вид «Программист»;
2. Переключиться на операцию «Bin - 2 байта»;
3. Проверить доступность кнопок ввода для двоичной системы счисления (доступны кнопки: 0, 1; недоступны кнопки: 2, 3, 4, 5, 6, 7, 8, 9, А, В, С, D, Е, F).

Для чего на главном окне приложения «Калькулятор» нажать сочетание клавиш Ctrl+Alt (Ctrl+Shift в зависимости от настроек) на клавиатуре, после чего отркроется Verify Properties приложения, где можно указать контролируемые свойства. Например, активность у приложения «Калькулятор» вида «Программист» (рисунок 3.27).

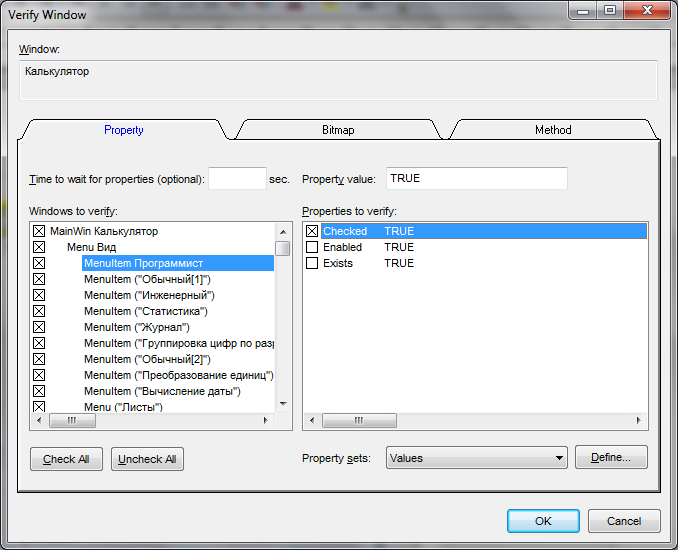


Рисунок 3.27.

Таким образом, устанавливаются для проверки все требуемые состояния кнопок и полей.

После чего можно приступить к проверке операции сложения на переполнение:

1. Набрать первый операнд «1111 1111 1111 1111»;
2. Выбрать операцию сложения «+»;
3. Набрать второй операнд 1;
4. Нажать кнопку получения результата «=»;
5. Проверить результат на равенство 0.

Для этого над полем результата снова необходимо нажать сочетание Ctrl+Alt (Ctrl+Shift) и ввести проверку на 0 как показано на рисунок 3.28.

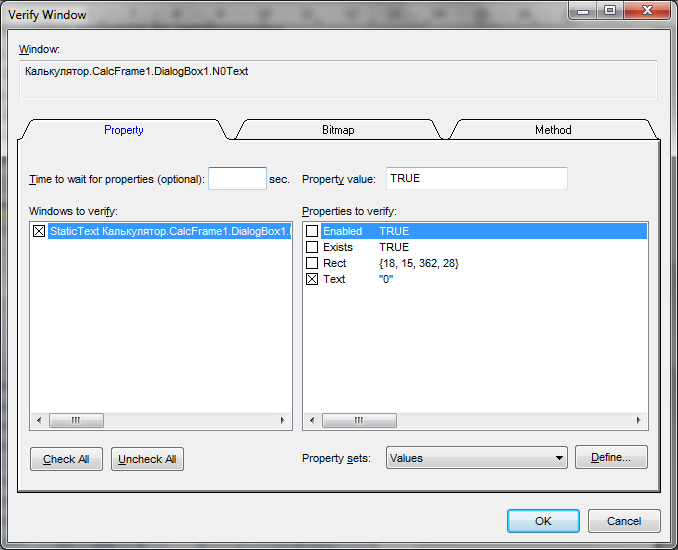


Рисунок 3.28.

Далее необходимо завершить запись тест-кейса путем нажатия кнопки Done и сгенерировать код скрипта нажатием кнопки Paste To Editor. В результате будет сгенерирован следующий код:

*[-] testcase Test1 ()*

*[-] recording*

*[ ] Калькулятор.SetActive ()*

*[ ] Калькулятор.Вид.Программист.Pick ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.RadioList1.Select ("Bin")*

*[ ] Калькулятор.CalcFrame1.DialogBox2.BitMap3.Click (1, 16, 50)*

*[ ] Калькулятор.SetActive ()*

*[+] Калькулятор.VerifyProperties ({...})*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N020.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N020.Click ()*

*[-] Калькулятор.CalcFrame1.DialogBox1.N0Text.VerifyProperties ({...})*

*[ ] ""*

*[-] {...}*

*[ ] {"Text", "0"}*

Проанализируем текст кода. Скрипт начинается с ключевого слова testcase. Далее следует его имя (Test1), в скобках можно указать передаваемые для этого тест-кейса параметры. Далее следует ключевое слово recording, показывающее, что код скрипта был записан автоматически. Это ключевое слово является лишь информативным и его можно удалить. Калькулятор.SetActive () – активация окна приложения. Хотя SilkTest автоматически активирует окно при первом обращении к нему (или к любому элементу внутри этого окна), все же желательно использовать метод SetActive(), чтобы быть точно уверенным, что окно активировано. Калькулятор.Программист.Pick() – выбор пункта меню Вид-Инженерный. Метод Pick() используется для выбора пунктов меню (для других элементов управления обычно используется метод Click(), например, для нажатия цифр калькулятора). Калькулятор.Move(x, y) – метод Move() используется для передвижения окон по экрану. В качестве параметров в этот метод передаются новые координаты по горизонтали и вертикали. Калькулятор.Close() – метод Close() служит для закрытия окон.

Следует обратить внимание на то, как SilkTest обращается к объектам. Если нужно обратиться к пункту меню Инженерный, который является потомком меню Вид, нельзя написать Калькулятор.Инженерный. В этом случае SilkTest выдаст ошибку. В этом примере немного элементов управления и нет большой вложенности объектов (иерархии). Однако в больших приложениях, когда иерархия может достигать 10-20 уровней вложенности, возникает необходимость модификации фреймов с группировкой элементов управления, отличной от того, как это делает SilkTest по умолчанию.

Заметим, что в данном коде одна и таже операция прописана несколько раз:

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click()*

Модифицируем данный код следующим образом:

*[-] testcase Test1 ()*

*integer i*

*[-] recording*

*[ ] Калькулятор.SetActive ()*

*[ ] Калькулятор.Вид.Программист.Pick ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.RadioList1.Select ("Bin")*

*[ ] Калькулятор.CalcFrame1.DialogBox2.BitMap3.Click (1, 16, 50)*

*[ ] Калькулятор.SetActive ()*

*[+] Калькулятор.VerifyProperties ({…})*

*for(i=0;i<16;i++)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N020.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N020.Click ()*

*[-] Калькулятор.CalcFrame1.DialogBox1.N0Text.VerifyProperties ({...})*

*[ ] ""*

*[-] {...}*

*[ ] {"Text", "0"}*

Теперь запустим сгенерированный и отредактированный скрипт на выполнение. Для воспроизведения скрипта, следует выбрать пункт меню Run-Testcase. В появившемся окне указать тест-кейс (в нашем случае это Test1) и нажать кнопку Run.

Скрипт отрабатывает примерно за 1 секунду и в итоге SilkTest выдаст отчет о проделанной работе (во сколько времени скрипт начал работать, как долго работал, сколько произошло ошибок и предупреждений (рисунок 3.29)

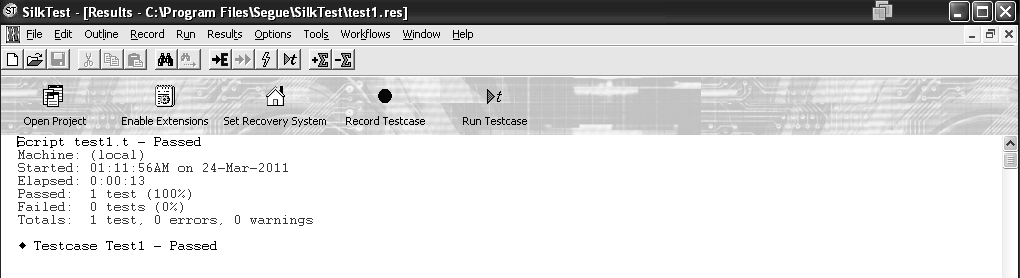


Рисунок 3.29

Создадим подобным образом реализацию тест-кейса №2, выполняющего проверку операции сложения над отрицательными числами.

После выполнения шагов теста получим следующий код:

*[-] testcase Test2 ()*

*[ ] integer i*

*[-] recording*

*[ ] Калькулятор.SetActive ()*

*[ ] Калькулятор.Вид.Программист.Pick ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.RadioList1.Select ("Bin")*

*[ ] Калькулятор.CalcFrame1.DialogBox2.BitMap3.Click (1, 16, 50)*

*[ ] Калькулятор.SetActive ()*

*[+] Калькулятор.VerifyProperties ({...})*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N115.Click () //-*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()//1*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()//0*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()//1*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()//0*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N115.Click ()*

*[-] Калькулятор.CalcFrame1.DialogBox1.N0Text.VerifyProperties ({...})*

*[ ] ""*

*[-] {...}*

*[ ] {"Text", "1111 1111 1111 0110"}*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()//1*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()//0*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()//1*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()//1*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N022.Click ()*

*[-] Калькулятор.CalcFrame1.DialogBox1.N0Text.VerifyProperties ({...})*

*[ ] ""*

*[-] {...}*

*[ ] {"Text", "1111 1111 1110 1011"}*

3.7.5. Создание Testcase с использованием Data Driven Test

Для создания тест-кейса №3 будем использовать Data Driven. Но сначала создадим тесткейс без использования Data Driven. Выполнив шаги инструкции, получим следующий код:

*[-] testcase Test3 ()*

*[ ] recording*

*[ ] integer val1=1000101*

*[ ] integer val2=111010*

*[ ] integer result=1111111*

*[ ] integer temp=10*

*[-] integer value*

*[ ] Калькулятор.SetActive ()*

*[ ] Калькулятор.Вид.Программист.Pick ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.RadioList1.Select ("Bin")*

*[ ] Калькулятор.CalcFrame1.DialogBox2.BitMap3.Click (1, 16, 50)*

*[ ] Калькулятор.SetActive ()*

*[ ] //наберем число 1*

*[-] while(val1!=0)*

*[ ] value= val1 % temp*

*[-] if(value==1)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[-] if(value==0)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()*

*[ ] val1=(val1-value)/temp*

*[ ]*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N022.Click () //+*

*[-] while(val2!=0)*

*[ ] value= val2 % temp*

*[-] if(value==1)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[-] if(value==0)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()*

*[ ] val2=(val2-value)/temp*

*[ ]*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N023.Click () //=*

*[-] Калькулятор.CalcFrame1.DialogBox1.N0Text.VerifyProperties ({...})*

*[ ] ""*

*[-] {...}*

*[ ] {"Text", Str(Result) }*

Для создания DDT-файла выполним следующие шаги:

**Шаг 1**: создать файл с именем Numbers.

**Шаг 2**: заполнить Excel-таблицу данными для тесткейса (рисунок 3.30).

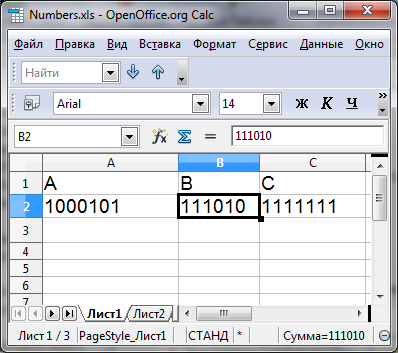


Рисунок3.30.

**Шаг 3**: войти в SilkTest Window и выбрать Tools.

**Шаг 4**: выбрать Data Driven Testcase (тест-кейс должен быть открыт в SilkTest).

**Шаг 5**: открыть окно Select Testcase. Выбрать тест и нажать OK (рисунок 3.31):

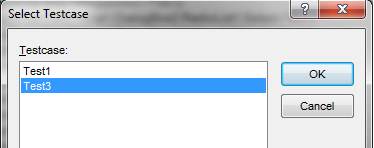


Рисунок3.31.

**Шаг 6**: открыть окно Specify Data Driven Script. Выбрать «Create a new file/Overwrite an existing file» и нажать Ok (рисунок3.32):

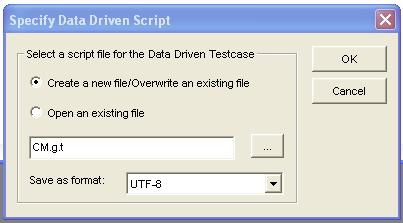


Рисунок 3.32.

**Шаг 7**: открыть окно Select Data Source. Выбрать Silk DDA Excel. Выбрать Excel-файл через диалог Browse и нажать Ok после выбора (рисунок 3.33):

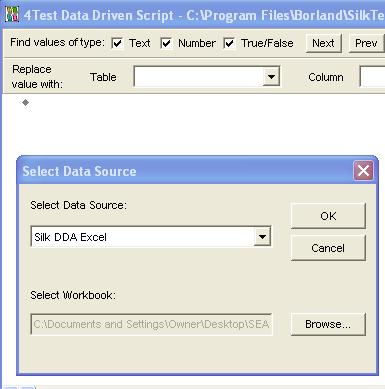


Рисунок3.33.

**Шаг 8**: открыть Specify Data Driven TestCase окно. Выбрать Add a new Data Driven Testcase и нажать Ok (рисунок 3.34):

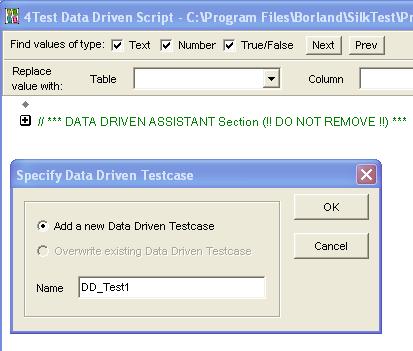


Рисунок 3.34

**Шаг 9**: открыть окно Find/Replace Values. Нажать Cancel в Find/Replace Values (рисунок3.35):

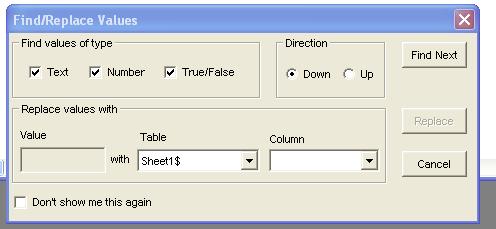


Рисунок 3.35

**Шаг 10**: войти в файл, который сохранен с именем вашего скрипта, но имеет расширение .g.t.

**Шаг 11**: в файле выбрать тип данных (Find values of type). Если тестовые данные представляют собой текст, выбрать Text. В Replace value with для Table выбрать Sheet1$, в Column выбрать столбец с именем, который присвоен столбцу в Excel-файле. После выбора таблицы и столбца станет доступна кнопка Replace.

**Шаг 12**: значения необходимо вставить в скрипт. Последовательно выбирая данные для замены в скрипте и соответственно изменяя на соответствующий Column в таблице Excel, путем нажатия клавиши Replace вставляем данные из DDТ-файла. Получим следующий код:

*[-] testcase Test3 (REC\_DATALIST\_DD\_Test3 rData)*

*[ ] integer val1=rData.recSheet1.A*

*[ ] integer val2=rData.recSheet1.B*

*[ ] integer result=rData.recSheet1.C*

*[ ] integer temp=10*

*[-] integer value*

*[ ] Калькулятор.SetActive ()*

*[ ] Калькулятор.Вид.Программист.Pick ()*

*[ ] Калькулятор.CalcFrame1.DialogBox2.RadioList1.Select ("Bin")*

*[ ] Калькулятор.CalcFrame1.DialogBox2.BitMap3.Click (1, 16, 50)*

*[ ] Калькулятор.SetActive ()*

*[ ] //наберем число 1*

*[-] while(val1!=0)*

*[ ] value= val1 % temp*

*[-] if(value==1)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[-] if(value==0)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()*

*[ ] val1=(val1-value)/temp*

*[ ]*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N022.Click () //+*

*[-] while(val2!=0)*

*[ ] value= val2 % temp*

*[-] if(value==1)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N155.Click ()*

*[-] if(value==0)*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N156.Click ()*

*[ ] val2=(val2-value)/temp*

*[ ]*

*[ ] Калькулятор.CalcFrame1.DialogBox2.N022.Click () //=*

*[-] Калькулятор.CalcFrame1.DialogBox1.N0Text.VerifyProperties ({...})*

*[ ] ""*

*[-] {...}*

*[ ] {"Text", Str(result) }*

**Шаг 13**: Нажать Save и выбрать RunTestCase. В окне RunTestCase выбрать DD-скрипт. Нажать Run.

3.7.6. Создание ТестПлана

Так как в проекте создано несколько тесткейсов, организуем их выполнение в тестпланах по следующим сценариям:

1. Запуск только Тест-кейс 1.
2. Запуск только Тест-кейс 2.
3. Запуск только Тест-кейс 3.
4. Запуск Тест-кейсов 1, 2, 3.

Для создания тестплана необходимо выбрать пункт File-New, в результате появится окно New File (рисунок 3.36), в котором необходимо выбрать пункт Test Plan. В результате будет создан пустой тестплан в виде файла с расширением .pln. Для добавления тест-кейсов в тестовый сценарий следует выбрать пункт TestPlan-Detail (рисунок3.37).

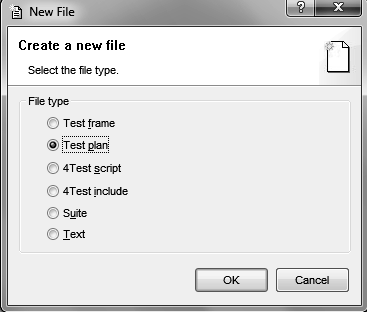


Рисунок 3.36.

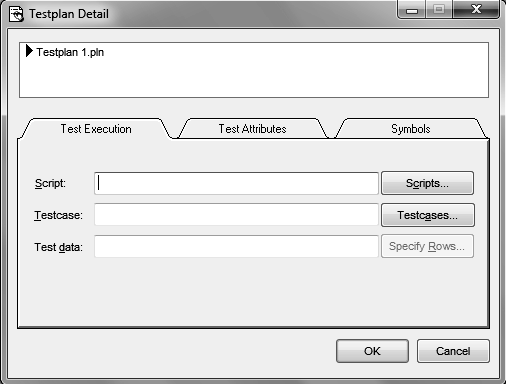


Рисунок 3.37.

В поле Script выбрать файл-скрипт содержащий необходимые тесткейсы, а в поле Testcase выбрать из списка содержащихся в нем тесткейсов необходимый. После чего нажать Ok.

В результате получим тестплан запуска тестов (рисунок 3.38)

Для запуска тестплана необходимо выбрать пункт Run-All Tests либо кнопку на панели управления SilkTest – Run the Script, suite or Testplan in the Active Window.

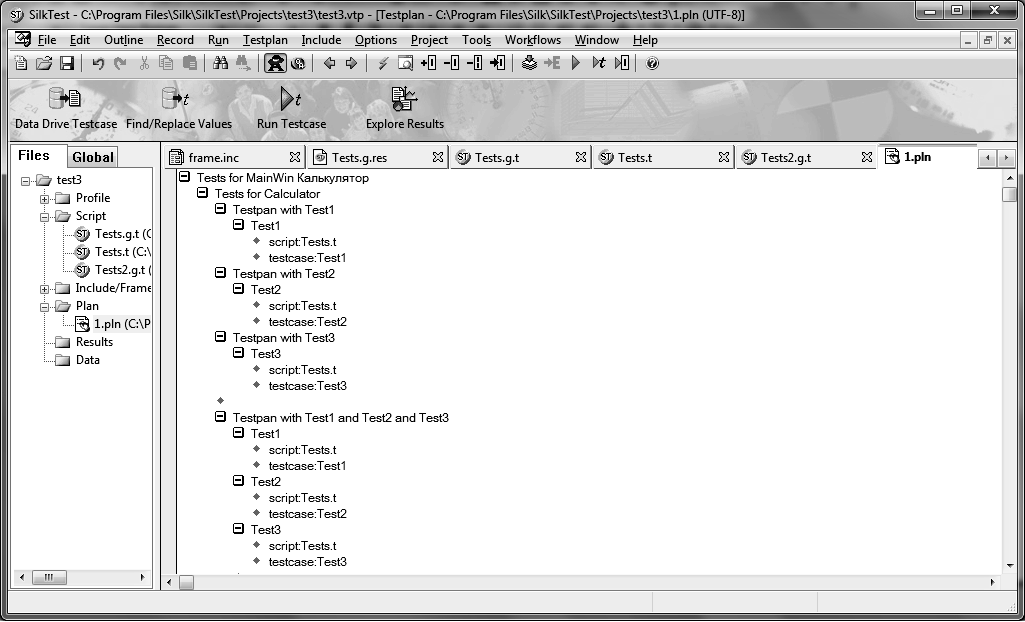


Рисунок3.38.